Accurate Visual Features for Automatic Tag Correction in Videos

Hoang-Tung Tran, Elisa Fromont, François Jacquenet, Baptiste Jeudy

Accurate Visual Features for Automatic Tag Correction in Videos

Hoang-Tung Tran, Elisa Fromont,
Francois Jacquenet, and Baptiste Jeudy

Université de Lyon, F-42023, St-Etienne, France,
CNRS UMR 5516, Laboratoire Hubert-Curien, 42023 St-Etienne, Fr,
Université de Saint-Etienne, Jean Monnet, F-42023 St-Etienne, Fr

Abstract. We present a new system for video auto tagging which aims at correcting the tags provided by users for videos uploaded on the Internet. Unlike most existing systems, in our proposal, we do not use the questionable textual information nor any supervised learning system to perform a tag propagation. We propose to compare directly the visual content of the videos described by different sets of features such as Bag-Of-visual-Words or frequent patterns built from them. We then propose an original tag correction strategy based on the frequency of the tags in the visual neighborhood of the videos. Experiments on a Youtube corpus show that our method can effectively improve the existing tags and that frequent patterns are useful to construct accurate visual features.

1 Introduction

As a result of the recent explosion of online multimedia content, it is more and more important to index all forms of web content for various search and retrieval tasks [12]. Classic text-based search engines already offer a good access to multimedia contents in the online world. However, these search engines cannot accurately index the extensive resource of online videos unless these videos are carefully annotated (mostly by hand) before being put on the web. However, user-provided annotations are often incorrect (i.e. irrelevant to the video) and incomplete. The reason for the former is because uploaders might want to rapidly increase the video's number-of-view by tagging it with a popular tag such as "Harry Potter", even though that video has no relationship with this famous book series. Incompleteness means that a given list of correct tags might not be sufficient to describe the video. Because of these two issues, a lot of online videos are hidden to text-based search engines (i.e. to users).

To overcome these drawbacks, we will focus on the task of improving annotations of web video data. Our aim is to set up a system which would be able to handle the two above drawbacks. In addition, our system should be able to suggest the most suitable tags for a new uploaded video (for example on Youtube). There have been many efforts to automatically annotate videos (e.g [9], [12]). However, most of the current proposed systems use limited concepts (tags) and some supervised information to learn one or multiple classifiers to tag a video

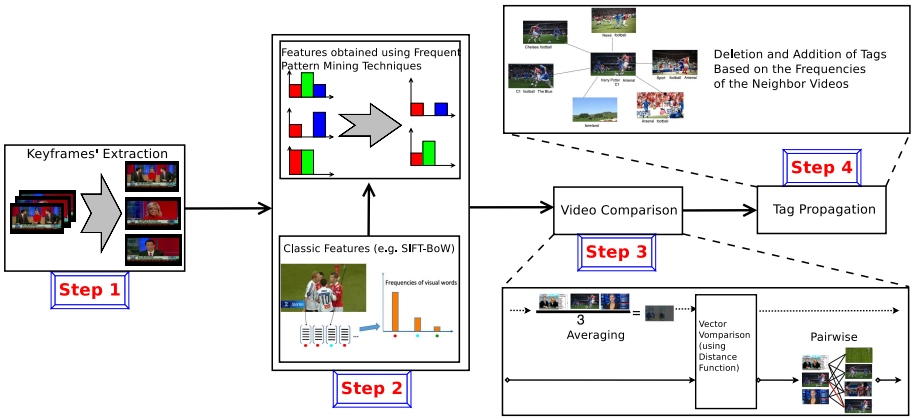

Fig. 1. Video Comparison Process

dataset. These approaches seems inappropriate to correct the tags of any video on a large website such as Youtube where the number of possible tags is infinite and where the ground truth (true labels) is inaccessible a priori. Thus, we would like to propose an unsupervised approach based on the comparison of the visual content of the videos to propagate the tags of the most similar videos (these tags are the only supervised information we use) based on their textual frequency. In this approach the major scientific issues lie in: i) the choice of the features are used to make relevant unsupervised comparisons, ii) the comparison method itself, iii) the propagation process and iv) the evaluation of the entire system.

The remainder of this paper is organized as follows. A review of related works concerning the above mentioned problems is given in Section 2. In Section 3, we describe in details how to apply data mining techniques as well as our proposed method to compare videos. Experiments on a real Youtube dataset converted into a huge transactional dataset are presented in Section 4 and show that our system can correct relevant tags. We conclude in Section 5.

2 Related Works on the General Framework

The first step of our system (described in Fig. 1) is to decompose a video into a sequence of keyframes (using for example [22]). In the following, the word frame is used for "keyframe". The related works concerning the subsequent steps taken in our tag correction approach are presented below.

2.1 Relevant Features (step 2)

Depending on the task you wish to perform on video, the best suited features can be different. So, the current trend in computer vision is to concatenate different kinds of low level features into a high dimensional vector that will be subsequently used for solving the vision tasks. For example, when dealing with

video comparison for automatic tagging, [10] uses edge distribution histograms, color moments or wavelet texture color autocorrelograms. [19] use both audio and visual features or Histograms of Oriented Gradient (HOG) from [3] as additional features. In [9], frame features include other kinds of global color histograms, and Haar and Gabor wavelets. Another very popular technique is to construct *Bag Of visual Words* [18] (BOW) from the original low-level feature vectors. These BOW are built by applying a clustering algorithm on the low-level descriptors (e.g. color RGB 3-D vectors). The number k of clusters is the number of different visual words. A frame can then be encoded by an histogram of the visual words it contains called a BOW.

However, when using only the visual content to compare videos, the above-mentioned features might not be accurate enough. Frequent pattern mining techniques are more and more often used in the computer vision community to provide better features (see e.g. [20] and [6]). Those approaches often rely on class information to select, in a post-processing step, a compact set of relevant features from the output of the mining algorithms. Without this selection step, this output would not be usable in practice to describe images.

2.2 Video Similarity (step 3)

Even though a video is considered as a sequence of images, variation in the videos duration or in the number of keyframes makes them more difficult to compare. We describe three categories of methods to compare videos. The first one consists in considering the average of the features of the keyframes. For example [19] consider the average of all frame histograms to produce a single histogram for the whole video. The histogram can be thresholded to remove some potential noise. Here classical distance functions such as $L1$ or $L2$ or histogram intersection can be used to estimate the similarity between videos. Even if this method is computationally efficient, one loses a lot of the available information by averaging all the frames. The second approach consists in comparing pairs of keyframes. For example in [10], the authors measure the similarity between two videos as the similarity between the two most similar frames of the videos. The comparison of the two videos is made using a unique pair of frames and no sequential information is taken into account. The last approach makes use of common identical frames called *near duplicate* to compare videos (see e.g. [21]). These frames are visually similar but different in terms of formatting, viewpoint, change in camera parameters, etc. but their common parts can still be used to compute a similarity score. Even though near-duplicate phenomenon appears quite often on video sharing sites, this approach can not be applied to all videos and especially not for the large set of videos found, e.g., on YouTube.

2.3 Tag Propagation (step 4)

The problem of automatic tag corrections of videos has often been tackled in the literature especially during the TRECViD [11] competition. However, it is often treated as a multi-label classification problem [14] or as a tag ranking problem

[7, 4]. The latter consists in finding a list of the most relevant tags for a new video given information about its neighborhood (this information can be visual as in our case or, for example, social in the context of social networks). Even if they are close to our problem, these two methods assume that the number of tags is fixed and known in advance and that they can have access to a perfectly tagged set of videos to learn a good model for each tag.

Since most video auto tagging systems use a supervised approach, the tag propagation step is not needed. However [21] uses such propagation procedure on which we base ours. For each video v, a list of possible-relevant tags is obtained from the k most similar videos (using a k-nearest neighbor algorithm). After that, a score function is applied for each tag to estimate its relevance according to the video v. This score function depends on the tag frequency (the higher the frequency, the higher the score), the number of tags associated to a video (the higher the number, the smaller the score), and the video similarity (the higher the similarity, the higher the score). Finally, all scores that are larger than a predefined threshold will be considered as suitable tags for the video v. Others tags (with smaller scores) will be deleted if they in Video v tag list.

3 Our Auto-tagging System

3.1 Proposed Features

As explained in the introduction, we can use many possible features to describe a video and this is a crucial point to work on to get a relevant tag propagation at the end of the process. As our low level features, we propose, as often suggested in the literature to describe images, to use 1000-dimensional (1000-D) BOW constructed from SIFT descriptors [8] (128-D descriptors) obtained regularly on a grid. However, as suggested in [6], we propose to extract those BOW locally from each frame to obtain more relevant frequent patterns later on. More precisely, for each point on the grid, we will create a BOW by counting the visual words which corresponds to that point and to its 18 nearest neighbors. This arbitrary number depends on the resolution of the video but roughly corresponds to a local description of half overlapping windows around each points. Each keyframe is thus described by a large number of BOW (in practice around 250 per frame).

Data Mining Techniques to Find More Discriminative Features. As mentioned in section 2, the data mining techniques used in the literature to obtain better features for video processing (for example APRIORI or LCM [16]) output a huge number of patterns (exponential in the number of dimension of the binary vectors). Those patterns can be filtered out using supervised information as, e.g, shown in [2]. However, in our case, no supervised information is available thus different criteria have to be proposed.

Both KRIMP [17] and SLIM [13] algorithms have been proposed to reduce the number of output patterns without relying on supervised information but by optimizing a criterion based on the Minimum Description Length principle. Both algorithms solve a minimal coding set problem but they differ in the way

they choose the collection of candidate patterns. KRIMP follows a straightforward two-phases approach: it first mines a collection of frequent itemsets, then it considers these candidates in static order, accepting a pattern if it improves a compression criterion. However, mining candidates is expensive and by considering candidates only once, and in a fixed order, KRIMP sometimes rejects candidates that could have been used latter on. SLIM greedily constructs pattern sets in a bottom-up fashion, iteratively joining co-occurring patterns such that compression is maximized. It employs a simple yet accurate heuristic to estimate the gain or cost of adding a candidate. For this reason, SLIM is faster and can handle larger datasets than KRIMP. In conclusion, SLIM seems a good candidate algorithm to filter out our patterns.

Converting Features into Binary Form. Most frequent pattern mining techniques use binary or transactional data. Therefore, the BOW must be converted into binary vectors. The most simple (and classical) method to do so is to transform all non-zero values into one. A lot of information is lost during this conversion if the original histogram is dense with many different values for each bin of the histogram. However, in our case, the 1000-D histogram contains at most 19 values (corresponding to the 18 neighbors + 1) and this is also the maximum value for a bin. This simple procedure thus seems appropriate to avoid unnecessary large binary vectors.

Encoding videos with BOW and FP. If F is the set of frequent patterns obtained using SLIM, we build a binary vector V of size $|F|$ for each keyframe. In this vector, $V(i)$ is set to 1 if the i^{th} pattern of F appears in this keyframe and 0 otherwise.

In our experiments, we encode our videos using BOW vectors, frequent pattern vectors (FP) built from them or with both of them (BOW+FP). For this last case, we need to normalize the feature vectors since both types of features have different distributions. Let N_{BOW}, σ^2_{BOW} be the number and the variance of the values in the BOW sub vector; and N_{FP}, σ^2_{FP} the number and variance of the values in the FP sub vector. We modified all the values of the FP vector as follows:

$$FP_{new}[i] = FP[i] * \frac{\sigma_{BOW}}{\sigma_{FP}} * \frac{N_{BOW}}{N_{FP}}$$

The new BOW+FP feature vector is the concatenation of BOW and FP_{new}.

3.2 Proposed Asymmetrical Video Similarity Measure

We propose an asymmetrical similarity measure inspired by the video pairwise comparison techniques to increase the relevance of the video comparisons. The first step consists in calculating all the pairwise similarities between all the keyframes of two videos. After that, instead of taking the optimum value of all the pairwise similarity scores (as in [21]), we propose to take the average of all maximum similarities corresponding to one video. In other words, for each

keyframe of a video A, we search in all the keyframes of video B for the highest pairwise matching score and we keep this value. Then, we take the average of all the computed values for all the keyframes of the video A to return the similarity score of video A towards video B. If we denote $A(i)$ the i^{th} keyframe of A and $|A|$ the number of keyframes in A, then

$$sim(A, B) = 1/|A| \sum_i \max_j sim(A(i), B(j)).$$

The similarity $sim(A(i), B(j))$ between frames is just the inverse of a distance between the vectors representing the frames (in the experimental section, we use the histogram intersection [15]). When the two frames are identical, this asymmetrical similarity is set to a maximal value.

3.3 Proposed Tag Propagation Algorithm

As explained in Section 2, to tag a given video $v \in V$, we rely on the tags $t \in T$ of the k most similar videos in its neighborhood. To propagate a given tag t to v, one need to set a threshold on the number of times t should appear in the neighbors. However, given the very different distribution of each tag, we decided to use two comparison statistical tests between the distribution of a tag in the entire dataset and its distribution in the k nearest neighbors. The first one states that the probability of a given tag should be significantly greater than 0 in the entire dataset to be propagated and the second one state that it should be significantly more present in the neighbors than in the entire dataset. Formally:

- (Global scale) A tag can be propagated if:

$$\hat{p} \geq u_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

  where $\hat{p}$ is the proportion of a tag over the whole dataset, N is the total number of videos, $u_{\alpha/2}$ is $\frac{1-\alpha}{2}$ percentile of a standard normal distribution.
- (Local scale) A tag is propagated if:

$$\hat{p}_1 \geq \hat{p} + u_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{k} + \frac{\hat{p}(1-\hat{p})}{N}}$$

  where $\hat{p}_1$ is the proportion of a tag in the k neighbors.

We arbitrarily decide to remove a tag from a video if it is never present in its neighbors. Note that the central limit theorem applies whenever $k \geq 30$.

4 Experiments

4.1 Protocol

Dataset. We pre-processed a Youtube dataset [1] with more than 10000 videos already decomposed into shots and keyframes. There are about 18 shots per

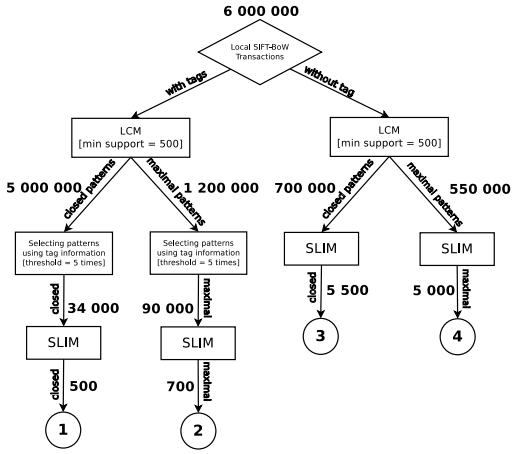

Fig. 2. Different strategies to obtain an accurate set of frequent patterns.

video and 1.5 keyframes per shot, i.e. about 27 keyframes for one video. We first decided to focus on videos with some common tags to obtain an interesting sample of the original dataset. For that, we focused on the 500 most frequent tags in the original set. We then removed the articles, pronouns, prepositions, words with less than two letters and the 50 most common tags in the remainder list (those, such as the word "video", were considered too frequent to be informative). This gave us a list of authorized 150 tags. We then kept the videos that contains at least 5 of those 150 tags and more than 1 keyframe. That led us to consider a corpus of 668 videos. Note that this is smaller than the 10000 initial videos but enough to illustrate our method. From this set of videos and from the local SIFT-BOW feature vectors computed from the keyframes we created a binary dataset of about 6000000 1000-D transactions.

Evaluation of the results. We randomly chose 50 videos from the 688 videos, and tagged them by hand using the 150 authorized tags to obtain a ground truth. We ran our tag propagation method on the 688 videos and reported the accuracy results for these particular 50 videos. This accuracy is measured in terms of "percentage of good corrections" (PGC). Let $T_{add,correct}$ be the correctly added tags out of $T_{add,total}$ added tags and $T_{remove,correct}$ be the correctly removed tags out of $T_{remove,total}$ removed tags.

$$PGC = \frac{T_{add,correct} + T_{remove,correct}}{T_{add,total} + T_{remove,total}}$$

If PGC is larger than 0.5, our system improves the tags in the videos. Note that since most existing tag correction systems use some supervised information, we do not compare our system to them. The following experiments stand for a proof of concept of our system.

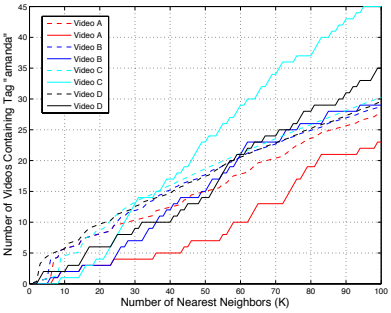

Fig. 3. Number of neighbors that contain the tag "amanda" according to the number of nearest neighbors for 4 different videos (plain lines) and propagation threshold for these 4 videos (dashed line).

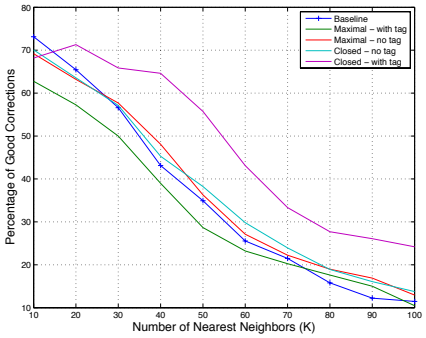

Fig. 4. Percentage of good corrections according to the number of neighbors in 50 videos represented with SIFT-BOW (baseline) and frequent patterns obtained LCM(closed and max)+SLIM and LCM(closed and max)+ SLIM+ tag_information.

Frequent Patterns Mining. As explained in Sect. 3, we first decided to use SLIM on the original transactional dataset. However, SLIM did not provide any fixed results after more than a week running. We then decided to use the well known and fast LCM algorithm [16] to mine closed or maximal patterns first and use SLIM as a post-processing step to select a less redundant number of patterns. The different strategies are shown in Figure 2.

Without tag information. We use LCM to find closed and maximal frequent patterns from the dataset made up of the 6000000 original 1000-D transactions. The support threshold was set as low as possible which corresponded for us to 500 (less than 1%). LCM produces $700k$ closed and $550k$ maximal patterns as shown in Fig. 2 (outputs 3 and 4). We then used SLIM to select a smaller set of around 5000 non redundant patterns (this is the final output of SLIM).

With tag information. For this experiment, we concatenate all these 1000-D vectors to 150-D vectors which describe for each transaction belonging to a frame of a given video, the list of tags that were associated to this video. Each of the 6000000 transactions is thus described by a sparse 1150-D binary vector. As shown in Fig. 2 (outputs 1 and 2), LCM is first used to produce around 5000000 closed and 1200000 maximal frequent patterns in a couple of hours. In this experiment, we make use of the existing tag information to filter out interesting patterns. A pattern is considered relevant for a certain tag if it is five times more frequent in videos that contain that tag than in videos that do not contain it. It is kept if it is relevant for at least one tag (note that this is similar to the concept of emerging patterns [5]). After going through this filtering process, the number of pattern is reduced to about 90000 for the maximal and

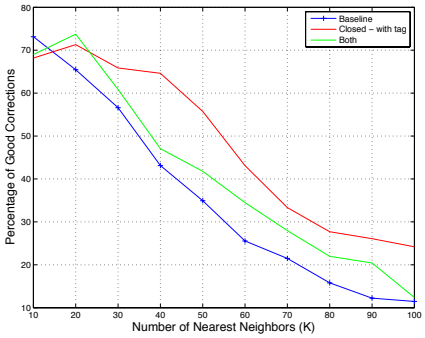

Fig. 5. Percentage of good corrections according to the number of neighbors for a video dataset represented with 3 different features: SIFT-BOW (baseline) and frequent patterns obtained with LCM(closed)+SLIM+tag_info, and the concatenation of both feature vectors.

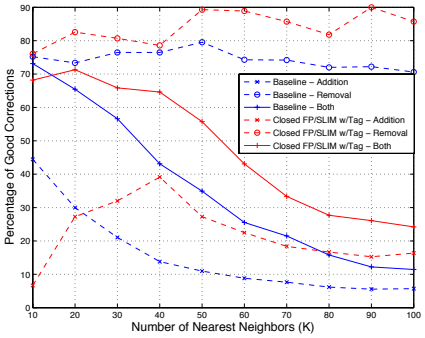

Fig. 6. Percentage of good corrections when counting only the addition (dashed line with +), the deletion (dashed line with ∘) and both according to the number of neighbors for a SIFT-BOW (baseline), LCM(closed)+SLIM+tag_info and the concatenation of both feature vectors.

34000 for the closed. Then, SLIM is used to produce (also in a couple of hours) around 600 non redundant patterns.

4.2 Tag Propagation Results

Test and Neighbors. As explained in Sect. 3, the number of neighbors considered for the tag propagation and the statistical tests are directly responsible for the propagation (or the removal) of a tag. To evaluate our choices experimentally, we selected a frequent tag ("amanda") in our video and 4 videos that should not be tagged with this particular tag. Fig. 3 shows the number of nearest neighbor videos that contain the tag "amanda" for the 4 different videos (plain lines). It also shows the propagation threshold (local scale in Sect. 3.3) according to the number of nearest neighbor videos that contains this tag (dashed line). Since the tag should not be propagated to these videos, the plain lines should stay below their corresponding dashed line. This is correct for 3 out of 4 videos for 30 nearest neighbors. It is not correct anymore when increasing the number of neighbors. This means that for this particular tag, increasing the number of neighbors will actually degrade the propagating system. Other similar experiments tend to confirm the relevance of choosing 30 neighbors.

Figure 4 shows the results for the 5 experimental settings described above. Note that, as explained in Sect. 3, using less than 30 neighbors questions our statistical tests. On the contrary, using 100 videos out of 668 clearly introduces a lot of noise. The best results are obtained using the combination of LCM(closed), SLIM and the tag information (case 1). In this case for 30 neighbors, the system is able to produce around 65% of good corrections which means that 54 correct

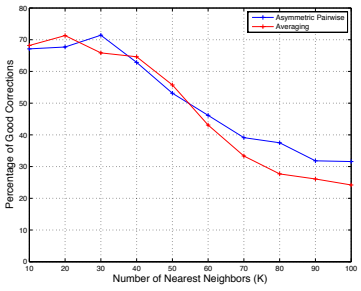

Fig. 7. Percentage of good corrections to the number of neighbors for a video dataset represented by frequent patterns obtained with LCM(closed)+SLIM+tag_information using the asymmetrical similarity measure and simply averaging the keyframes features and computing an histogram intersection.

tags were added or deleted in the process (28 were wrong propagations). For the same setting, the baseline only allows us to produce 57% good corrections. Figure 5 focuses on the baseline and the best case. It also shows the results when both vectors are concatenated. The concatenation produces worse results than the best case which means that the information given by the baseline is not complementary to the information given by the frequent patterns.

Figure 7 shows a comparison between our proposed asymmetrical similarity measure and the basic method which consists in simply averaging for one video all the keyframe features and computing an histogram intersection between the feature vectors representing two videos. Our proposed method does give better results but the difference is not significant using 30 neighbors. The frame average method may thus be preferred for efficiency reasons.

5 Conclusion

We have presented a complete tag correction system which corrects and completes original tags on videos without learning any model. We have proposed a new high level video feature vector to describe our videos based on frequent patterns and decided to compare the videos directly using an histogram intersection distance function. We have evaluated our method on a real Youtube dataset and shown that our system can effectively be used to correct tags. However, the new proposed feature vector and the pairwise video comparison procedure do not always make a significant improvement compared to naive methods which use simple BOW features (from SIFT) and averaging over the videos. As future work, we thus propose to take into account the sequential information in the video to create better high level features (such as frequent sub sequences). Besides, our proposed method should clearly be used off-line since the mining part takes a non negligible amount of time. Efficient algorithms (for example able to deal with streams) should also be designed to tackle a Youtube-scale dataset.

References

1. Cao, J., Zhang, Y., Song, Y., Chen, Z., Zhang, X., Li, J.: Mcg-webv: A benchmark dataset for web video analysis. Tech. rep., ICT-MCG-09-001, Institute of

Computing Technology (2009)
2. Cheng, H., Yan, X., Han, J., Yu, P.S.: Direct discriminative pattern mining for effective classification. In: 24th Int. Conf. on Data Engineering. pp. 169–178 (2008)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Conf. on Computer Vision and Pattern Recognition. vol. 1, pp. 886–893 (2005)
4. Denoyer, L., Gallinari, P.: A ranking based model for automatic image annotation in a social network. In: 4th Int. Conf. on Weblogs Social Media. pp. 231–234 (2010)
5. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: 5th Int. Conf. on Knowledge Discovery and Data Mining. pp. 43–52. ACM (1999)
6. Fernando, B., Fromont, E., Tuytelaars, T.: Effective use of frequent itemset mining for image classification. In: Europ. Conf. on Computer Vision. pp. 214–227 (2012)
7. Liu, D., Hua, X., Yang, L., Wang, M., Zhang, H.: Tag ranking. In: 18th Int. Conf. on World Wide Web. pp. 351–360. ACM (2009)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision 60(2), 91–110 (2004)
9. Morsillo, N., Mann, G., Pal, C.: Youtube scale, large vocabulary video annotation. Video Search and Mining pp. 357–386 (2010)
10. Moxley, E., Mei, T., Manjunath, B.: Video annotation through search and graph reinforcement mining. IEEE Transactions on Multimedia 12(3), 184–193 (2010)
11. Over, P., Awad, G., Michel, M., Fiscus, J., Sanders, G., Shaw, B., Kraaij, W., Smeaton, A.F., Quénot, G.: An overview of the goals, tasks, data, evaluation mechanisms and metrics. In: TRECVID 2012. NIST, USA (2012)
12. Shen, J., Wang, M., Yan, S., Hua, X.S.: Multimedia tagging: past, present and future. In: 19th ACM Int. Conf. on Multimedia. pp. 639–640 (2011)
13. Smets, K., Vreeken, J.: Slim: Directly mining descriptive patterns. In: SIAM Int. Conf. on Data Mining. pp. 236–247 (2012)
14. Sun, Y.Y., Zhang, Y., Zhou, Z.H.: Multi-label learning with weak label. In: 24th AAAI Conf. on Artificial Intelligence. pp. 593–598 (2010)
15. Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vision 7(1), 11–32 (1991)
16. Uno, T., Kiyomi, M., Arimura, H.: LCM ver. 3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In: 1st Int. Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations. pp. 77–86. ACM (2005)
17. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Mining and Knowledge Discovery pp. 1–46 (2011)
18. Yang, J., Jiang, Y., Hauptmann, A., Ngo, C.: Evaluating bag-of-visual-words representations in scene classification. In: Int. Workshop on Multimedia Information Retrieval. pp. 197–206. ACM (2007)
19. Yang, W., Toderici, G.: Discriminative tag learning on youtube videos with latent sub-tags. In: Computer Vision and Pattern Recognition. pp. 3217–3224 (2011)
20. Yuan, J., Yang, M., Wu, Y.: Mining discriminative co-occurrence patterns for visual recognition. In: Conf. on Computer Vision and Pattern Recognition. pp. 2777–2784 (2011)
21. Zhao, W., Wu, X., Ngo, C.: On the annotation of web videos by efficient near-duplicate search. IEEE Transactions on Multimedia 12(5), 448–461 (2010)
22. Zhuang, Y., Rui, Y., Huang, T., Mehrotra, S.: Adaptive key frame extraction using unsupervised clustering. In: Int. Conf. on Image Processing. vol. 1, pp. 866–870. IEEE (1998)