Large Scale Image Deduplication

Tzay-Yeu Wen
Stanford University
tywen@stanford.edu

Abstract

With the rise of internet and personal digital camera, it becomes easy for researchers to get image data in mass quantity. With these large amount of image data, it is impossible for humans to examine each image and insure the quality of the dataset. Therefore it is crucial to develop algorithms that can process large amount of data.

This paper will focus on a particular problem related to image dataset, image deduplication. We propose an efficient and scalable method to find near-duplicate images in an image collection. Our method includes 3-steps, first, extract compact features from each image. Second, use a fast clustering algorithm to reduce possible image match. The clustering algorithm should be CPU/memory efficient and can scale through multiple machines easily. We will use the method proposed by [4] that uses map-reduce to implement approximate nearest neighbor.

Finally, apply a more accurate method on the clustered images to find duplicate images in each group. We will use method proposal by [9].

1. Introduction

Near-duplicate image detection is a special kind of image retrieval problem, which is relatively easy and well studied compared to other computer vision problems. Several image features, for example [5] and [1], have been proposed to calculate the similarity of two image or image parts. Those features are robust to noise and many image transforms; Therefore are more than enough for duplicate image detection. Using feature aggregated from SIFT, [7] has proposed a method which will find similar images in the image dataset. Their experiments showed that their method maintains high accuracy for up to 1M of images.

The challenge however, is to be able to handle massive amount of images using reasonable computation resources. The amount of data an image retrieval algorithms, like K-means or KNN, can processed are constrained by the amount of available memory. Therefore many methods that can reduce the memory footprint or scale to multiple machines had been proposed.

[3] proposed a method that reduce the feature representation of each image into less than 100 bytes. This increase the limit on a single machine but the result is less accurate. [9] proposed another features aggregate algorithm that can take advantage from both local and global features. Their algorithm uses visual words to represent an image and inverted index to search though the dataset.

Another approach proposed by [4] is to compute the approximate nearest neighbor on features using Map-Reduce, which can process large amount of data with less accuracy.

2. Image Clustering

3. Feature Extraction

3.1. Bundle Feature

For each incoming image we will build a discriminative feature representation of it. We want the feature to be invariant to some image transformations including rotation, illumination, scale and crop, because those are the most common methods used by people when processing images.

SIFT, which is invariant to rotation and scale, is one of the robustest point feature in computer vision. It can achieve very high precision on small dataset. However, when the dataset become larger, false positive increase rapidly. To counter this problem, an intuitive method is to increase the feature space by combing multiple SIFT features into a feature bundle. Two bundles are consider matched if matched SIFT features exceed certain threshold.

As one can imagine, how the features are grouped can greatly affect the performance of the algorithm. Commonly used clustering algorithms like K-means, which group features by some distance metrics, are not suitable for this problem due to the following reason. First, because of different image depths and object occlusion, nearby feature may not belong to the same objects. The resulting bundle will depend on multiple objects and will be harder to match when the image is cropped. Second, it is hard to determined how many clusters each image should have, the number may vary greatly between complex images and sim-

ple images. Finally, even the image are similar, depending on different initial condition feature might not be grouped consistently. Therefore, a better clustering algorithm that is consistent across different images and can consider image context will be preferable.

3.2. Construction

In this section we will describe how to build the bundle image features using SIFT and MSER. For each image I_i, extract the SIFT features $S_i = \{s_{ij}\}$, where $s_{ij} = (x, y, \vec{f}),\ \ x, y \in \mathbb{R},\ \ \vec{f} \in \mathbb{N}^{128}$, and the MSER features $M_i = \{m_{ij}\}$, where $m_{ij} = (x, y, c_{xx}, c_{xy}, c_{yy})$ and c_{xx}, c_{xy}, x_{yy} are the covariance of the region. Finally, define the bundles for image I_i as $B_i = \{b_{ij}\}$.

We define the bundle feature

$$b_{ij} = \{s | s \in S_i \text{ and } s \text{ is inside region } m_{ij}\} \qquad (1)$$

The inside of m_{ij} is define by the ellipse, which is an approximation of the actual region, calculated from the covariance terms. We discard those bundles that has it's ellipse width or height larger than half of the image width or height because it is very hard to reproduce the same region in two different images. We also discard those bundle that has no or only one SIFT feature because this indicates that the region is too small and can potentially match to many other regions. In order to save storage and computing power, we also discard bundles that has it's SIFT features overlap with another bundles by more than 97%.

4. Image Matching

4.1. Visual Word

To retrieve large amount of image from the database, we need an efficient image representation that can be easily processed and indexed, compare to SIFT feature. The state of the art in image retrieval is to model image as document and image feature as visual word. We use the method propose by [8] to convert SIFT feature into visual word. It calculates the center of each visual word using hierarchical K-means [6]. After the center is calculated, we assign each feature to the first k nearest visual words to reduce the quantization error.

4.2. Bundle Matching

In this retrieval framework, what we actually do is matching bundles rather than images because the similarity between two images is simply calculated by summing the similarity between each pair of bundles in these two images.

$$S(I_i, I_j) = \sum_{b_1 \in I_i} \sum_{b_2 \in I_j} S(b_1, b_2) \qquad (2)$$

Where b_1, b_2 are bundle from the first and second image, respectively. The bundle similarity is calculated by the multiplying the standard tf-idf term by the correlation between bundles.

$$S(b_1, b_2) = Corr(b_1, b_2) \sum_{s \in b_1 \cap b_2} \text{tf-idf}(s) \qquad (3)$$

The correlation term is composed by similarity and locality.

$$Corr(b_1, b_2) = Corr_s(b_1, b_2) + \lambda Corr_l(b_1, b_2) \qquad (4)$$

Where $Corr_s(b_1, b_2)$ is simply the number of visual word these two bundles have in common.

$$Corr_s(b_1, b_2) = |b_1 \cap b_2| \qquad (5)$$

Currently this number is not normalized but we are experimenting with different normalization schemes, such as $max(|b_1|, |b_2|)$ and $max(area(b_1), area(b_2))$, to see if there are improvement to make.

We assume that the extracted regions are the invariant part of an image that are stable to image translation. Therefore, the feature location inside two matching bundles should be consistent, which is taken into account by the second term $Coor_l(b_1, b_2)$. A high score means that the feature inside these two bundles are positioned on similar location. The term is defined by

$$Corr_l(b_1, b_2) = \sum_{i=1}^{|b_2|} O^D_{b_1}(s_{2,i}) < O^D_{b_1}(s_{2,i+1}) \qquad (6)$$

Where $O^D_{b_1}(x)$ is the order of feature x in bundle b_1 with respect to a defined geometric order D. To make the locality term invariant to rotation, we ordered the feature by θ, which is it's position (θ, r) on the polar coordinate.

5. Experiment

5.1. Dataset

We will evaluate our method using two different image dataset, one for accuracy and one for performance.

For accuracy measurement we will use a dataset produced by [7], which contains 2550 distinct images. For each image we will generated 3 images by randomly combine translation, cropping and rotation, which will result in a total 10200 images dataset. We will query the database using all the 10200 images and calculate the accuracy accordingly. Following [9] we will use mAP as our evaluation metric.

For performance measurement we will use a dataset provided by ILSVRC2010, which contains about 1.2M images that are unlabled. The accuracy of the result will be manually verified. The performance will be evaluate by the CPU and memory usage.

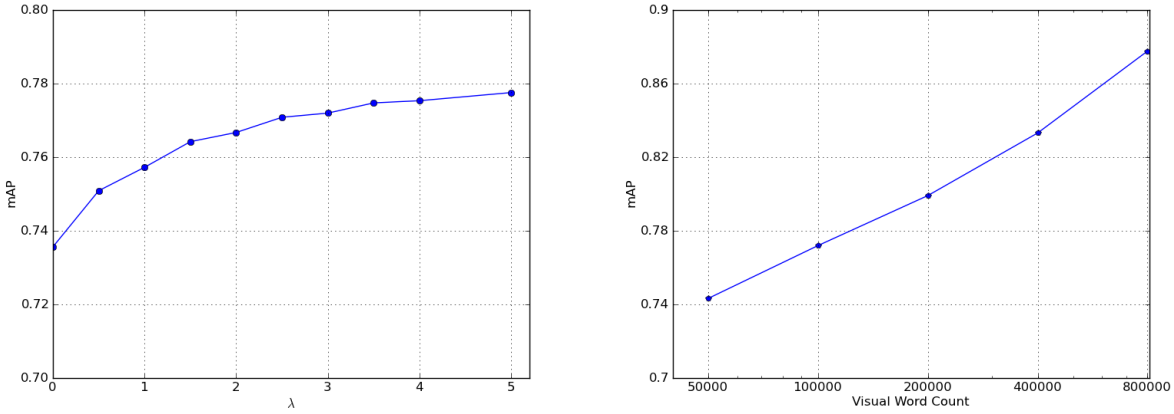

Figure 1. Left: Comparison of different λ term with visual words size 100K. $\lambda = 0$ means only the $Corr_s$ term is used. Right: Comparison of different visual words size with $\lambda = 3$. The original feature size is about 2.6M.

5.2. Result

As Figure 1 shows the locality term plays a important role on overall performance. Compare to the original paper [9], which define the geometric order on X-Y coordinate, our method have a much higher optimal λ. This suggests that in our experiments the locality term can more accurately represent the similarity between images, which confirms that rotation invariance can improve accuracy.

We also experiments with different visual words size. The original image data contain about 2.6M SIFT features, which are quantized into visual words. The size of visual word affect the performance greatly. If there are too few visual words, false match will increase. On the other hand, too many visual words will make matching impossible because most of the visual words will only map into one SIFT feature.

5.3. Runtime

The runtime of the system is quite slow. Currently, it takes about 3-5 sec to query a single image. The bottleneck is the calculation of the locality term which take about 50% of the time. We will try to decrease the possible relative images for each query by implementing hamming embedding [2] when building visual word.

References

[1] M. S. Extremal, J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from. In *In British Machine Vision Conference*, pages 384–393, 2002.

[2] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In A. Z. David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317. Springer, oct 2008.

[3] H. Jegou, M. Douze, C. Schmid, and P. Prez. Aggregating local descriptors into a compact image representation. In *CVPR'10*, pages 3304–3311, 2010.

[4] T. Liu, C. Rosenberg, and H. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, page 28, feb. 2007.

[5] D. Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[6] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[7] D. Nistr and H. Stewnius. Scalable recognition with a vocabulary tree. In *IN CVPR*, pages 2161–2168, 2006.

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[9] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 25 –32, june 2009.