Web Scale Image Clustering

Foteini Tsavo

June 15, 2020

Contents

Abstract

The rapid growth of digital images on the Internet, due to the rise of technology, has created a new challenge in the field of Image Mining, which is the efficient processing and management of large-scale collections containing up to billions of images. In this paper we focus on the problem of image retrieval. Our goal is to propose ways to achieve a quick and efficient retrieval by Image Retrieval Systems, whether they accept a query as text, Text Based Image retrieval Systems, or as an image, Content Based Image Retrieval Systems. Clustering algorithms seem to be the key for this problem, therefore, we propose some k-means variants, which are appropriate for clustering millions and billions of images into hundreds of clusters. Such algorithms are Scalable k-means by Ranked Retrieval and Inverted Quantized k-means. In particular, we focus on the advanced data structures that they use, Inverted Indexes and Inverted-Multi Indexes, in order to achieve a fast search, as well as the intuition behind the inverse search process.

1 Introduction

An image retrieval system can be described as an engine that the user trusts to browse, search and retrieve images from a large database of digital images. In this work, we refer to the two most common types of image retrieval systems, Text Based Image Retrieval and Content Based Image retrieval Systems. The main difference between these two types, is the query that the user addresses to the system, as it is shown in *Figure 1* .

For Content Based Image Retrieval System, the query is an image that the user inserts and the images that are expected to be retrieved should have something in common, either an object or a color, with the initial image. On the other hand, Text Based Image retrieval Systems return lists of images based on a query that is inserted as text. This traditional method of image retrieval system utilize some method of adding metadata such as captioning, keywords, or descriptions to the images so that retrieval can be performed over the annotation words [4].

A simple experiment we have conducted using the Google Search Image engine, probably the only widely known way to search the world wide web for images, was enough to notice that there is room for improvements. Firstly, we consider the search based on textual information, and type the word *flat*. As a result, we are expecting a list of images of apartments, whereas the real outcome is a mixture of images of apartments, flat objects, as well as images that seem to be completely irrelevant to our search. There are two reasons for this outcome and the first one is that the list of imaged is returned based on matching keywords, therefore keyword matching algorithms are used and images are not clustered based on their visual content. The other reason is because the certain query is short and ambiguous. As a consequence, Google should make a guess that the word is related to the image.

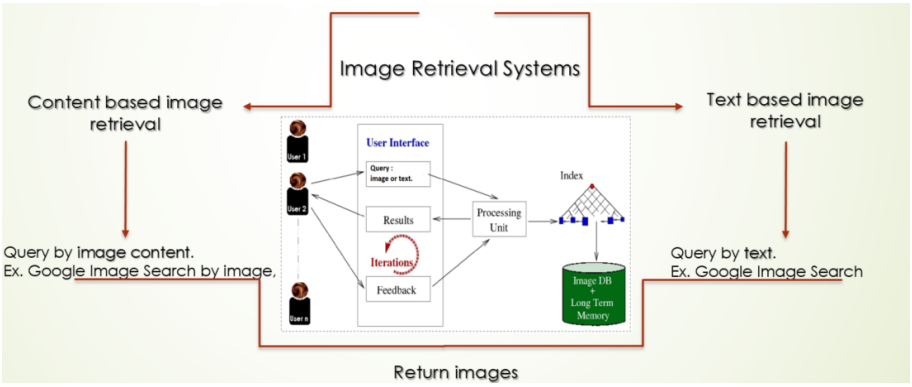


Figure 1: CBIR and TBIR

A great approach to achieve better search and more accurate results, is to combine image processing and image clustering algorithms that will organize digital images according to their visual contents. In *Section 2* we present some previews work that is related to our problem, in *Section 3* we explain the importance of feature vectors and state that the image processing stage, is able to help reduce the work and increase the efficiency in the clustering process, based on our experiments. We also, propose a method for the proper extraction of feature vectors, that will give as a better result if applied. Moreover, in *Section 4* we study some variants of K-Means and explain whether they are practical or not for our study. The goal of *Section 5* is to present and compare the data structures some of the most appropriate algorithms for web scale clustering. Furthermore, in *Section 6* we analyze two algorithms, proposing that their efficiency is what most of the Image Retrieval System Engines are looking for, today. Finally, we propose further research for the future in *Section 7* and conclude this work in *Section 8*.

2 Related Work

The recent years, a lot of research has been done in the field of Image Mining and there are a lot image techniques to mention, between them Image Retrieval, Image Indexing and Image Clustering, as they are described in [9].

Image mining requires that images be retrieved according to some requirement specifications. The requirement specifications can be classified into three levels of increasing complexity: comprises low level features of such as color, texture, shape or the spatial location of image elements, comprises image retrieval by derived or logical features like objects of a given type or individual objects or persons and Comprises high level features of image. Also, to further improve image retrieval rate, there is require of image data base using a fast and useful indexing scheme. A clever way to achieve better results of indexing, is with dimensionality reduction. Other pro-

posed indexing schemes concentrate on specific image features including color, texture and shape features.Finally, In unsupervised classification (or image clustering), the problem is always to group a given assortment of unlabeled images straight into Meaningful clusters based on the image content with not a priori knowledge. Clustering is often more advantage for minimizing the searching time period of images inside database. Specifically, there is a considerable number of algorithms that are used for clustering large number of images. Most of them are modifications of k-means algorithm, that are suitable for handling big data.

The problem of clustering is addressed in [7] paper, proposing to cluster binary hash codes of a large number of photos into binary clusters. A fast binary k-means algorithm, is presented, that works directly on the similarity-preserving hashes of images and clusters them into binary centers on which we can build hash indexes to speedup computation. The proposed method is capable of clustering millions of photos on a single machine in a few minutes.

Furthermore, a really fast algorithm for retrieval is the one presented in [2], the Scalable K-Means by Ranked retrieval, or else the wand-k-means. In this particular implementation, a different way of search is performed, that is the inverse process of the traditional's k-means search. Using centroids as queries into a nearest neighbor data structure built on the data points can be very effective in reducing the number of distance computations needed. Also, this algorithm is a modification of WAND algorithm [10] for similarity search to remember the top points that the centroid will be assigned to. By sparcifying the centroids, will improve performance further. The approach proposed in this paper is viable in practice, with a really considerable reduction in the computation time, especially for large values of k.

Another really fast algorithm for web scale image clustering is the Inverted Quantized k-means in [1]. The inverse process of search is used here as well, but with a significant difference, which makes this implementation even faster, using inverted multi-indexes as a data structure for searching and a proper quantization of datapoints, in the form of hashing. By using this two techniques, where input data are presented in the Euclidean space, an extremely fast variant of k-means is achieved, that can be used to any application. Also, dynamic estimation of k is achieved in the dynamic variant of IQ-Means, in a single run, which really important because the estimation k has been one of the disadvantages that was bothering us in k-means variants.This method of clustering was applied to 10_8 images, on a single machine, and the results were spectacular. It was really fast because the assignment step, which is the one where the most computations happen in k-means, is actually faster than the update step.

But before applying any clustering algorithm, Image Processing seems to be a really important step to take for a good result. As it is said in [11], interest in digital image processing methods stems from two principal application areas: improvement of pictorial information for human interpretation and processing of image data for storage, transmission, and representation for autonomous machine percep-

tion. The features of a digital image can play a big role for our clustering process, as we will present in our implementation.

3 Image Processing

In computer science, digital image processing is a method to perform some operations on an image, in order to extract some useful information from it. This information, is nothing else but the feature vector, we will later use for the clustering process. It is of a great importance to understand, before choosing the right clustering algorithm , based on what features the data should be grouped. For instance, if the user is more concerned about the color of the images that he or she searches, the color information of pixels should definitely be included in the feature vectors.

3.1 Experiments

Our study regarding feature vectors, is included in `https://github.com/TsFotini/Image_Retrieval-Clustering/blob/master/project.ipynb` . This work is implemented in python programming language and we also, have used a PostgreSQL Database, to store the paths of the images that we will be using for our experiments.

3.1.1 Raw Feature Vectors

Our first attempt to extract feature vectors from images that are stored in our local database, is to extract Raw Pixel Feature Vectors, which means that no processing has been applied to images. An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point.

After reading an image using OpenCV [5], matrixes that the elements of each of them represents the intensity of the pixel of each picture, are constructed. In fact, for one image that is translated in RGB form, three matrixes are created, one for each channel, *Figure 2*. The output that is shown above the image of Google, presents the way a machine can translate an image.

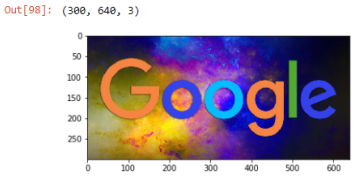


Figure 2: Image Representation with OpenCV

The first value, 300, is the height that represents the number of pixel rows in the image or the number of pixels in each column of the image array. The second value, 640, is the width that represents the number of pixel columns in the image or the number of pixels in each row of the image array. And finally, the third value, 3, is the number of channels that represents the number of components used to represent each pixel.

The challenge is to switch from feature matrixes to feature vectors, in order to perform later clustering with k-means written from scratch from us, as well as with the sklearn.kmeans

function, that already exists. This is achieved with the help of Principal Component Analysis. Principal Component Analysis is an unsupervised linear transformation technique that is widely used across different fields, most prominently for feature extraction and dimensionality reduction.

The groups that were formed after applying the sklearn.kmeans for k=3 are presented in the figures below. Each figure corresponds to a cluster.

Figure 3: Raw feature vectors: Cluster 0

Figure 4: Raw feature vectors: Cluster 1

Figure 5: Raw feature vectors: Cluster 2

3.1.2 Feature Vectors based on Mean Color

We have conducted experiments while extracting the feature vectors from the sample images we have stored, also, considering that color is an important ingredient of an image, and especially when we refer to an Image Retrieval System, color is a component that can not be ignored. The results are presented below.

Figure 6: Mean color feature vectors: Cluster 0

Figure 7: Mean color feature vectors: Cluster 1

Figure 8: Mean color feature vectors: Cluster 2

3.1.3 Gray Scale Image feature vectors

We have continued our research, this time eliminating the colors from all the images. In this way, we can distinguish better objects in our images.

Figure 9: Gray Scale Image feature vectors: Cluster 0

Figure 10: Gray Scale Image feature vectors: Cluster 1

Figure 11: Gray Scale Image feature vectors: Cluster 2

3.1.4 Mini - Content based Image retrieval System

After the extraction of different feature vectors from the same images, we have chosen mean color based feature vectors for our simple representation of a content based image retrieval system. We have inserted a query image, from which the feature vectors are extracted from, also based on mean color. After this action, the image is ready to be clustered and we are expecting a label. The label of the group that it will be cluster indicates the images that are most like the query image. The results are presented below.

Figure 12: Query Image

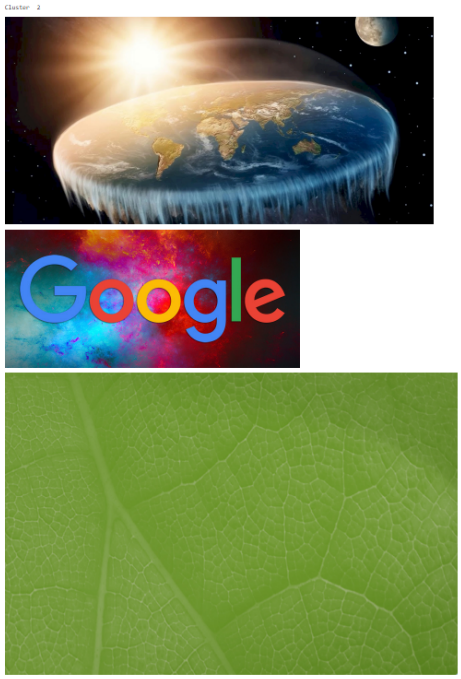


Figure 13: Retrieved Images based on mean color of query image

3.2 Proposal

We would like to propose a method that will help in clustering images based on their visual content and we refer to invariant features and invariant feature histograms [6]. The goal is to extract features invariant against translation and rotation. those particular features, will model the global appearance of the image, and not each object that is included in the image. These means that those features will remain the same, even if some transformation is applied to the same image. The integral approach of the features is presented: $\frac{1}{|G|}\int_{g\epsilon G} f(gX)dg$.

We will start the explanation of this formula with gX value, that represents a transformed image, and to be more precise g is the arbitrary transformation, whereas X is the matrix that contains the pixel values. If we apply this to a group of translations and rotation, the result we will get is formula with some integrals of an integral. Usually these complicated integral , is replaced by sums to achieve discretization. We replace one or more of the sums that are formed by histogramization. A histogram is an estimation of the distribution of a variable. The probability for data points falling into one of the regions of the histogram is determined by counting.

Summarizing, if the goal is to create a proper grouping of images for a Image retrieval System, the feature vectors play a great role. The features that we should be interested in are the color, shape of objects and texture of image. More importantly, we should

take into consideration the features that do not change when transformations are applied to the image.

4 K-Means and more variants

The reason why we have chosen to study k-means algorithm is because it one of the most popular unsupervised machine learning algorithm and many clustering algorithms that exist today have taken their main ingredients from K-Means Algorithm, but we can not ignore that their optimized. But firstly, the explanation of K-Means will help us in order to understand more complicated algorithms of this type.

4.1 K-Means

The ways that we can start initializing in K-means differ. One way is to generate random points, but this type of action will , probably, not be the best because we might have centroids that are really far from the input dataset, as it presented in our implementation. Another way of initializing centroids is to choose points that are already party of the dataset, so that we will achieve a better grouping. And thirdly, using K-means++ for initialization, is maybe one of the best choices we can make, because the centroids are not only part of the dataset, but they are as far as possible.

The comparison between the first method, that is the naive k-means, and the third are implemented from us and it is pretty obvious that K-means++ to approach this problem.

Below we will present the steps that we have followed to implement k-means algorithm. In general, k-means aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Step 1: Choose the number of clusters K.

Step 2: Select at random K points, the centroids(not necessarily from your dataset).

Step 3: Assign each data point to the closest centroid $\rightarrow$ that forms K clusters.

Step 4: Compute and place the new centroid of each cluster.

Step 5: Reassign each data point to the new closest centroid. If any reassignment took place, go to step 4,otherwise, the model is ready.

4.2 K-Means++ Initialization

This is an algorithm for choosing the initial values for the k-means clustering algorithm. The k-means problem is to find cluster centers that minimize the intra-class variance, the sum of squared distances from each data point being clustered to its cluster center (the center that is closest to it). The intuition behind this approach is that spreading out the k initial cluster centers is a good thing: the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the

remaining data points with probability proportional to its squared distance from the point's closest existing cluster center.This seeding method yields considerable improvement in the final error of k-means. Although the initial selection in the algorithm takes extra time, the k-means part itself converges very quickly after this seeding and thus the algorithm actually lowers the computation time. below we will present the steps of this algorithm.

Step 1: Choose one center uniformly at random among the data points.

Step 2: For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.

Step 3: Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.

Step 4: Repeat Steps 2 and 3 until k centers have been chosen.

Step 5: Now that the initial centers have been chosen, proceed using standard k-means clustering.

4.3 Other variants of K-Means

Even though K-Means algorithm has a long history and a practical performance, it does not scale to clustering millions of data points into thousands of clusters, for data in high dimension spaces. One of the disadvantages we mention, is the need to compute the nearest centroid for every data point at every iteration. Consequently, when the number of clusters is large, the cost will be high.

In order to avoid large cost, many algorithms were implemented as optimizations of K-Means, mainly, focusing in the optimization of the search that is requires in the assignment step. One of them worth mentioning, is the Binary K-Means algorithm [7], where all points and centroids vectors are represented in binary form and an approximate nearest neighbor search is applied. Each data point is assigned to a the nearest centroid using as a metric, Hamming distance. But a better alternative, in order to speed up the search, would be to invert the process of searching. In K-means and Binary K-means the way the search is performed, is each point of the dataset searches for the centroid that it best for it, meaning the centroid that has the smallest distance. But, for a better search, we can invert the roles so that centroids will be in the position of queries that search the fixed dataset points for the best vectors, that will be part of the cluster they represent. This choice requires less queries because centroids are fewer than dataset points. Points are examined more than once and one disadvantage that is noticed here is that not all points find their cluster, some of them remain unassigned. Despite this, distortion is not influenced much. Two algorithms we mention that follow this technique and are really efficient for large datasets, are Scalable K-means by Ranked Retrieval and IQ-Means.

5 Comparison of Scalable K-means by RR and IQ-Means

The main characteristic of both algorithms, is that their performace is really high because of the way they perform a search on big dataset. Although, in general the idea of the inverse search is the same, each of them uses a different data structure for fast search.Scalable K-means by Ranked Retrieval uses inverted indexes and IQ-Means uses inverted multi-indexes [3].

5.1 Inverted Indexes

An inverted index is a database index storing a mapping from content, such as words or numbers, to its locations in a table, or in a document or a set of documents. They are used for evaluation of similarity queries across many different domains. What precisely happens in an inverted multi index structure is described below:

In an inverted index, each term t ϵ has an associated postings list which contains all of the documents d ϵ D that contain t, those with d $\cap$ t $\neq \emptyset$. For each such document d, the list contains an entry called a posting. A posting is composed of the document id (DID) of d, and other relevant information necessary to evaluate the similarity of d to the query.

Furthermore, we will describe how inverted indexes work for data as images. After the feature vector extraction, we are left with a large number of vectors of large dimension. It would not be very helpful to start clustering all those data but instead, we can map some of the vectors into some codewords. One way such a mapping can happen , is to perform clustering, K-means in all the dataset and gain clusters that are represented as Voronoi cells as in *Figure 12* . There we have a representation of two codewords, one is part of the pink cluster and the other one part of the blue cluster. All the gray points, are points of the dataset that will be mapped to these particular codewords accordingly.

Also, in this approach we are able to use Vector Quantization , a popular technique used in applications such as image and voice compression, voice recognition and in volume rendering. A vector quantizer maps k-dimensional vectors in the vector space R^k into a finite set of vectors Y = {yi: i = 1, 2, ..., N}. Each vector y_i is called a code vector or a codeword. Associated with each codeword, y_i, is a nearest neighbor region called Voronoi region and this can be seen in *Figure 3*.

The power of inverted indexes is the low latency and the good scalability properties they have. To add more, they are very simple to develop and that is why they are popular. On the other hand, we notice some disadvantages that they have. For instance, if the input is an enormous dataset a large number of list will be generated for each codeword, as a result their memory is not efficient. So, the large storage overhead and high maintenance increases costs on update, delete and insert operations.

Consequently, the structure that Scalable K-means by ranked Retrieval used, is fast but if the dataset increases dramatically, as it would probably happen in the world of web, we might not have the results we desire regarding its performance.

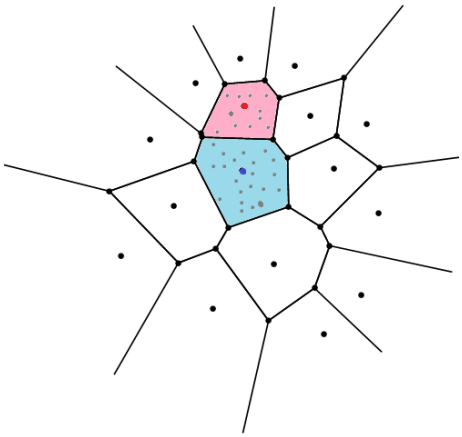


Figure 14: Inverted Indexes

5.2 Inverted Multi-Indexes

Inverted Multi-Indexes are new data structure for efficient similarity search in very large datasets of high-dimensional vectors. In general, they follow the inverted index idea, they present vectors with codewords, but in this case the approach is more efficient for large data because of the space subdivision. When using inverted multi-indexes, the vectors that are returned after querying, are better localized around the query point. Their great advantage is to produce much finer subdivision of the search space while keeping the query time at low costs, as well as the preprocessing time, compared to inverted indexes.

The main difference here, is the use of product quantization instead of vector quantization. Here a dimesion decomposition is applied. An simple example of such decomposition is showed below:
Given a point set X $\subset$ R^d , where $|X|$ = n and assuming that d is even, for simplicity purposes:
R^d is expressed as the Cartesian product of two orthogonal subspaces S^1, S^2. In the simplest form S^1 =S^2 = R^g , where $g = \frac{d}{2}$, i.e x=(x^1, x^2), where x^1 ϵ S^1 = R^g, x^2 ϵ S^2 = R^g .

The essence of Product Quantization is to decompose the high-dimensional vector space into the Cartesian product of subspaces and then quantize these subspaces separately. An example that shows how the search place is divides is presented in *Figure 13*. The dimesion decomposition can be performed recursively until the one-dimensional space is reached. The entries of this particular data structure, can be placed in a grid as possible tuples of codewords from the codebooks, set of all the codewords, corresponding to different dimension groups. Each of these entries of the table are associated with a part of the original vector space and contains a list of points that fall within that part.

IQ-means is an algorithm that e assume the same representation and codebook building as in multi-indexing. This is one of the reasons why it is considered faster than Scalable K-means by ranked retrieval.

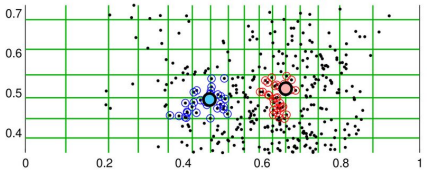


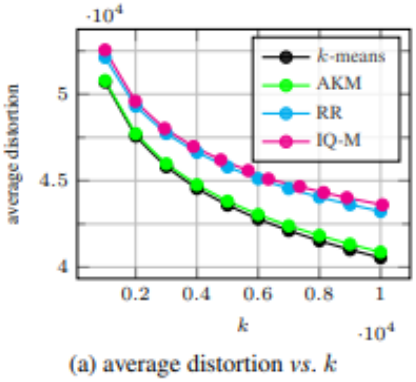
Figure 15: Inverted Multi-Indexes,Source: [3]

5.3 Experiments

We will present the experiments that were conducted in [1] between four know algorithms: k-means, approximate k-means, Scalable k-means by Ranked Retrieval and IQ-means.

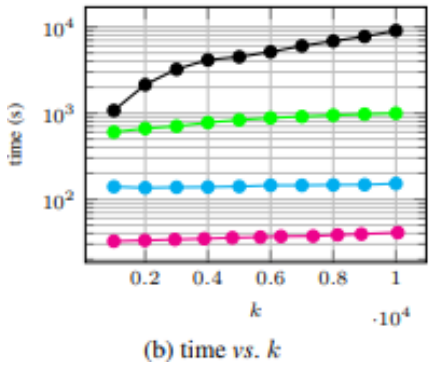
We will explain with a few words Approximate k-means algorithm [12]. In an attempt to reduce the cost of finding pairs that match, approximate nearest neighbor search was used to help. In this case each data point is assigned to the nearest centroid by ANN search. As it can be understood, this is closer to the traditional assignment, than to to the inverse search process that is applied in Scalable k-means by ranked retrieval and IQ-means. Novel quantization method based on randomized trees is used and to further improve query performance, an efficient spatial verification stage to re-rank the results returned from our bagof-words model and show that this consistently improves search quality, though by less of a margin when the visual vocabulary is large.

The dataset that has been used is SIFT1M consisting of 1M 128-dimensional SIFT vectors, and a learning set of 100K vectors. The average distortion is presented, and total time for 20 iterations on SIFT1M for varying number of clusters k. Time for IQ-means includes encoding of data points that is constant in k, but not codebook learning, which is performed on a different dataset.



(a) average distortion *vs.* k

Figure 16: Average Distortion vs k,Source: [1]



(b) time *vs.* k

Figure 17: Time vs k,Source: [1]

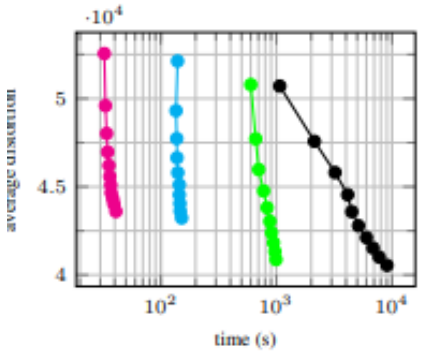


Figure 18: Average Distortion vs time ,Source: [1]

5.4 Conclusions

Both algorithms are based on inverted process of searching and they both have the same small disadvantage, which is the loss of accuracy due to unassigned points. The other thing they have in common is that they alternate between assignment and update step. But it is essential to notice that Scalable K-means by ranked retrieval is fast by inverted search, whereas IQ-means is even faster by more efficient inverted search, using a more advanced data structure, inverted multi-indexes.

6 Web scale clustering algorithms

As we have mentioned in other section, clustering digital images based on their content or even based on their color, is not an easy task, especially when we have to deal with a very big amount of data, which is the situation nowadays in the web. This is the reason why we have chosen to propose the following algorithms, as a way to optimize image retrieval systems: Scalable K-means by ranked retrieval and Inverted Quantized K-Means.

6.1 Scalable K-Means by Ranked retrieval

The main reason why this algorithm was developed, was to reduce the cost of K-means with a different kind of search, in order to speed up the computations. The key insight is to invert the process of point-to-centroid assignment by creating an inverted index over all the points and then using the current centroids as queries to this index to decide on cluster membership. In other words, instead of indexing the centroids and using the points as queries, we propose indexing the points and using centroids as queries. The benefits of this technique are two. One is the reduction of number of queries that is performed per operation, since there is one query per centroid, and the second one is that there is no need to rebuild the index between iterations because the points do not change.

6.1.1 Ranked Retrieval and WAND algorithm

One of the most special ingredients of this algorithm, is the method that it uses to retrieve the l points closest to a centroid, each time, and that is achieved by ranking. The structure that is used here is inverted index, which is optimized for retrieving most similar documents for a query under an additive scoring model. Let us suppose we have a query q such that $q \subseteq T$, where T contains some terms, each with a given weight $q(t)$. Given a similarity function, in this case cosine similarity function is used, the score depends on the weight q and on the d, where d is a vector of a certain document , lying in m-dimensional term space.

Specifically, we consider the $d(t)$ and assume that it represents a weight for the particular term in the document. Such weights can be derived by using tf-idf weighting framework. To explain this further, in information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. In our case the most weighted word can be translated as the most weighted feature.

To add more, a method that allows fast retrieval of the top-l ranked documents for a query is a modification of the WAND algorithm for similarity search, to remember the runner-ups in the current iteration and the radius of each cluster in the previous ones. The way that this algorithm works is based on keeping a heap of highest scoring top documents. The WAND algorithm finds an intelligent way to skip the documents that have no chance in making it to the top, as result this saves precious time. This is done by determining that the score of a document can not be higher than a certain score θ, which is required for a current query document. Below we will explain the simple WAND algorithm for Top-l Retrieval.

WAND OPERATOR

Suppose we have a list of Boolean variables: X_1, X_2, ...,X_l. A list of positive weights has been constructed based on the tf-idf weighting model: w_1, w_2,

...,w_l. Also, the threshold is known and it is θ. ***WAND***(X_1,w_1,...X_l,w_l,θ) is true iff $\sum_{1 \le i \le l} x_i w_i \ge \theta$, where $x_i = \begin{cases} 1, & \text{if } X_i \text{ is true} \\ 0, & \text{otherwise} \end{cases}$. If we evaluate for each document on the first level we will have ***WAND***(X_1,UB_1,...X_l,UB_l,θ), where X_i is the indicator whether the query term is in the document. UB_i represents the max frequency score for each query document. Also, l is the total number of query terms and θ is the minimum score in the current documents. As we have mentioned above, those documents are kept inside a heap. As a result, if we consider that ***WAND*** is true for a certain document, it means that the full score has possibility of being greater than threshold θ. This means that document d can be inserted to the heap where top documents are stored. If ***WAND*** is false, then document d has no chance to be inserted among the top documents because $UB(d,q) < \theta$, consequently, this is how the document will be skipped.

```
WAND
   Input: query q, cluster id i, table A, ℓ, threshold θ_i
   Output: updated table A
for all i ∈ q do
   Cursors[i].init
end for
H ← Heap of size ℓ
while true do
   θ_i ← max(θ_i, min(H))
   ⟨d, sim⟩ ←WAND-Next(Cursors, θ_i)
   if d == null then
      return A
   end if
   if (A[d] == null or A[d].sim < sim) then
      A[d].sim ← sim, A[d].clust ← i
      H.add(sim)
   end if
end while
```

```
wand-k-means:
   Input: D = Set of documents, k
   Output: Centroids, C = {c_1, ..., c_k}
φ = ∞
C ← InitializeCentroids
θ[1..k] ← 0
repeat
   φ* ← φ, φ ← 0, A ← ∅
   for all c_i ∈ C do {Find Nearby Documents}
      ⟨A, θ_i⟩ ← WAND(c_i, i, A, ℓ, θ_i)
   end for
   for 1 ≤ i ≤ k do {Compute Assignment}
      C_i ← {d ∈ D : A[d].clust = i}
   end for
   for 1 ≤ i ≤ k do {Recompute Centers}
      c_i ← 1/|C_i| Σ_{d∈C_i} d
      for all d ∈ C_i do {Update Cost Function}
         φ ← φ + A[d].sim
      end for
   end for
until φ = φ*
return C = {c_1, ..., c_k}
```

6.1.2 Conclusions

To summarize, the key idea behind the algorithm presented thus far is to consider only the l-closest points to the cluster center when executing the assignment phase of k-means.To find the l-closest points, the WAND algorithm gains added efficiency by giving up on the distance computation as soon as one can safely conclude that the point is further away than the desired threshold. Since this algorithm potentially examines only the closest l- points in each iteration, to ensure convergence, it is computed the cost of assigning all of the points every 50 iterations, and stop if no progress is made on this objective. The infrastructure initially designed for nearest neighbor retrieval to improve the speed of k-means is leveraged. Particularly, the inverted index data structure that is used for nearest neighbor retrieval is especially efficient in the case that the query itself is sparse, i.e. contains few non-zero entries. Also, an additional optimization is sparsifying each cluster center after computing it, effectively only keeping the top p features with the highest absolute value.

All things considered, it is shown that using centroids as queries into a nearest neighbor data structure built on the data points can be very effective in reducing the number of distance computations needed by the k-means clustering method.Also, considering the trade-off between the setting of the parameter l and the number of documents marked as unassigned by the algorithm is worth it, because the unassigned documents are those that do not appear in the top-l list any cluster center during the execution of WAND. Note that this algorithm can also be called, wand-k-means algorithm.

6.2 Inverted Quantize K-Means

The IQ-means Algorithm is implemented after studying the recent advances in approximate k-means variants and borrowing the best ingredients from each of them. The main idea about this algorithm, is to map the data points, as we have described in Inverted-multi Indexes *Section 4.2*, and perform an inverted search process, a concept from Scalable K-Means by Ranked retrieval. In this way, if the images are represented in the Euclidean space, a two dimensional grid is formed as in *Figure 17*.

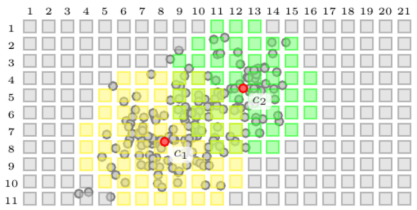


Figure 19: Mapping the points with IQ-means (Two-dimensional grid),Source: [1]

Each cell of this particular grid is composed of the Cartesian of product of two sub-code words. Each sub-code word comes from the sub-space that was produced by the dimension decomposition applied before. The complexity of this algorithm depends on this grid and not on the large number of datapoints because as they are discarded when the grid is created. This gives a major advantage to this

algorithm, to be really fast for large number of images. This, as well as the accuracy that it provides with its results, are the reasons why we propose that it would be suitable for Image Retrieval Systems, for example Google Search Image Engine and others, that their tast is to handle and process in no time, really big data.

6.2.1 Vector Quantization

One of the key underlying concepts is the quantization of datapoints, in a form of hashing and analogous to seed detection. Below, we will describe the process that is followed for quantization and the concept of dimension decomposition. To start with, we are given a set of points $X \subset R^d$, where the number of points in the dataset equals n, therefore $|X| = n$. For simplicity purposes we will assume that d is even, and we will proceed to the dimension decomposition step.

DIMENSION DECOMPOSITION
The purpose here is to express the Cartesian product of two orthogonal spaces. In our example, where d is even what we will do is separate it into S^1, S^2, where each of them equals $R^{\frac{d}{2}}$. As a result, if we consider a $x \epsilon X$, the x point can be expressed as $x = (x^1, x^2)$, where $x^1 \epsilon S^1$ and $x^2 \epsilon S^2$, meaning that we get the code-word that the x element is mapped in the first subspace S^1, and the code-word that also, x is mapped according to subspace 2, S^2. It is quite important to add that this particular method, can achieve a better mapping of points to code-words, because the way of approaching this points is more accurate, when we have to deal with large dimensions and large number of points. After this step, we have a new representation of points.

REPRESENTATION OF POINTS
Considering the ilustration in *Figure 5*, we assume that we have two sub-codebooks, or else two clusters, U^1, U^2, trained independently on projections of sample data that S^1, S^2, respectively. Each of them will contain s code-words. From the dimension decomposition, we know that if we take the Cartesian product of those two sub-spaces, we will reach the initial space, therefore we can write $R^d = S^1 \times S^2$. As a result, the Cartesian product of sub-codebooks U^1, U^1, will produce a grid U, that can be seen as a discrete two dimensional grid, and an element u can be seen as a cell, $u \epsilon U$. Moreover, this grid will contain $s \times s$ code-words. So, given two sub-codewords, one for each sub-codebook, $u_i^1 \epsilon U^1, u_j^2 \epsilon U^2$, where $i, j \epsilon [s] = \{1, ..., s\}$, we will present a cell that is made of both of those sub-codewords with the symbol u_α with α being the tuple (i, j).

QUANTIZATION
A set of tuples is created for each codeword in the grid, totally $s \times s$. Based on this information, we can write that $(i, j) \epsilon I = [s] \times [s]$. All things considered, every point that is part of dataset X can be quantized to a cell, $q(x) = (q^1(x^1), q^2(x^2))$, where $q(x)$ is a quantizer for point $x \epsilon X$. After all points are quantized on the grid, a two-dimensional discrete distribution p of points over cells is applied, where for each cell u_α it measures the empirical frequency of points of the dataset falling into u_α, $p_\alpha = \frac{|X_\alpha|}{n}$, where X_α is the set of all point of dataset that are

quantized into cell u_α.

6.2.2 Update Step

In order to continue to update and assignment step, we first, should mention how the initial centroids are generated.

THE INITIAL SET OF CENTROIDS
To begin with, the mean for each of these cells is found. The mean for one cell u_α is calculated with the formula $\mu_\alpha = \frac{1}{|X_\alpha|}\sum_{x\epsilon X} x$. So this can be explained as calculating the average number of points, that when quantized can fall into this particular u_α cell. This mean is kept for each cell.
After this procedure dataset X can be discarded. Instead of points of dataset X, the grid with the mappings will be used in the update and assignment step, which alternate as in k-means.

UPDATE
Starting with an arbitrary set C of k centroids, where k is specified by us, we will update each c_m, $c_m \epsilon C$, simply by computing the weighted average:

$$c_m = \frac{1}{P_m}\sum_{\alpha\epsilon A_m} p_\alpha \mu_\alpha$$

Here, the symbol P_m is the proportion of points that are assigned to centroid c_m, $P_m = \sum_{\alpha\epsilon A_m} p_m$. As for the symbol A_m, this is the set of all indices of the cells that are assigned to c_m, during the assignment step, $A_m = \{\alpha\epsilon I : a(u_\alpha) = m\}$. So if we use the current example of the grid that we have in *Figure 18*, we can describe, the cells that belong to each centroid. By looking at *Figure 6*, $m = 2$, because there are two centroids, in our case, and the first centroid c_1 has all the cells with yellow color assigned to it, whereas the c_2 has the green cells assigned to it. This means that cells $(8,8), (9,8), (4,6), (4,7)$ and others, are definitely assigned to centroid c_1, therefore the A_1 will contain their indices. As for A_2 , the indices that will be contained are coming from the cells that are assigned to the c_2 centroid and we can give an example for those cells $(5,12), (1,11), (2,10), (4,16)$ and other cells colored as green.

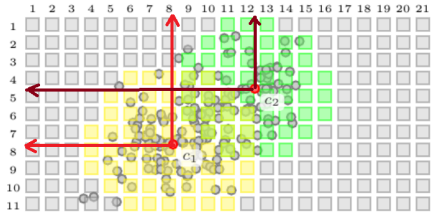


Figure 20: Mapping the points with IQ-means (Two-dimensional grid - indexes),Source: [1]

6.2.3 Assignment Step

This is a step where computations must happen, in order to create the clusters, each time. As a result, fast search is required. Specifically, the inverted process of search is used here, which means a centroid to cell search, following a multi-indexing approach using the multi-sequence algorithm. So, the purpose is to find the nearest cells for centroids and assign it to one of them.

ASSIGNMENT STEP
For each centroid c_i the w nearest subcodewords are found in U^1, U^2 and ordered by ascending distance toc_i. The w $\times$ w cells are then visited in order

via a priority queue. Upon visiting a cell a function f is called. In this case, it updates the current assignment α and lowest distance dist found for each cell u_α. It also terminates upon visiting a specified target T of points.

SEARCH PROCESS We will use again, as an example the *Figure 18* grid. By looking at its colors we are able to distinguish which cluster is assigned to each cell, by computing their distances. What we notice here is that the colors yellow and green for U_1, U_2 accordingly, overlay in some cells such as $(4,9),(5,9),(6,9)$. This means that they should be visited twice and with a comparison it can be a understood which is the nearest centroid to choose. The search process is illustrated for each centroid with a grid, where we start from the cell that the centroid already is , so we are refering to the minimum distance we can find, and then we proceed by placing the nearest cells that we find on the topleft corner of the grid.

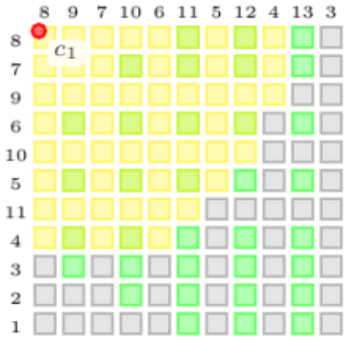


Figure 21: Centroid c_1,Source: [1]

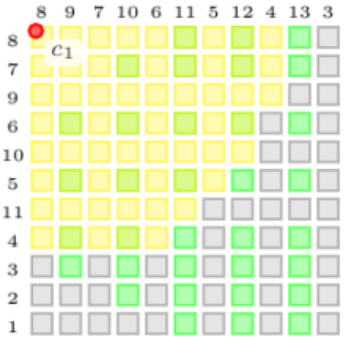


Figure 22: Centroid c_2,Source: [1]

Centroid to Cell Multi Sequence Search:

for $c_m \epsilon C$ **do**
  $(k^1, d^1) \leftarrow NN_u(c_m^1, U^1)$
  $(k^2, d^2) \leftarrow NN_u(c_m^2, U^2)$
  $f.Init$ ▷ initialize list
  $Q.Init$ ▷ initialize queue
  while $Q.NotEmpty()$ **do**
    $((i,j),d) \leftarrow Q.Min()$
    $\alpha \leftarrow (k_i^1, k_j^2)$
    $V \leftarrow V \cup \alpha; visit[\alpha] \leftarrow true$
    if $f.(m, \alpha, d)$ **then**
      `break;`
    end if
    if $i = 0 \vee visit[k_{i-1}^1, k_{j+1}^2]$ **then**
      $Q.Push((i, j+1), d_i^1 + ... + d_{j+1}^2)$
    end if
    if $j = 0 \vee visit[k_{i+1}^1, k_{j-1}^2]$ **then**
      $Q.Push((i+1, j), d_{i+1}^1 + ... + d_j^2)$
    end if
    for $a \epsilon V$ **do**
      $visit[\alpha] \leftarrow false$
    end for
  end while
end for

Above we have presented how the search algorithm works. We suppose

that we use a queue Q and f is the dunction that updates the distance and the assignment. For each centroid, we find its nearest neighbor according to the sub-codebook $U^1, U^{@}$. After that we initialize a list for a function f where we will keep the piece of information that we had mentioned above. Furthermore, we will initialize a queue and insert an element that is the origin, in order for the queue to not be empty and pass the while statement we have below the initialization. Each time, we visit a cell and record it. We check with function f if it terminated, or else if the target is reached or not, and if yes than we exit the loop. To continue with, a cell to the right is visited if the one above the right is visited. On the other hand, a cell below is visited, if the one below left is visited.

6.2.4 Dynamic IQ-Means

Dynamic IQ-Means' main difference from IQ-Means is the way of searching. It does not only search from centroid to cell, but it also searches from centroid to centroid, keeps track of the neighboring centroids of each centroid, and then uses this information to compute cluster overlaps and purge clusters between iterations in order to determine k. The centroid to centroid search process , also relies in the same structure of search we have already mentioned, as the centroids can certainly be quantized on the grid as if they were data points. When centroids that are nearby are found, the algorithm will be able to estimate the clustering overlap. Estimating k is a really useful, because one of the major disadvantages of k-means is the initialization of k. Depending on the k value we choose, the output of clustering will change. Consequently, the estimation of k is a pretty important property.

6.2.5 Conclusions

To summarize, IQ-Means is a fast algorithm appropriate for web scale image clustering, as its it discards the large dataset and continues with a high quality mapping of datapoints, using inverted-multi index approach. The finer the grid is, the higher the quality of data representation. But if the cells of the grid are too many, then this will affect directly the complexity of the algorithm. Moreover, a good partition of the space plays a great role. Finally, with the really clever method that the search is implemented in Dynamic IQ-Means, it is achieved the estimation of clusters in a single run at zero cost.

7 Future work

The research we have made for Image Retrieval Systems and algorithms that can handle a large number of data, has answered some questions about the way of searching and retrieving, but has also raised some questions. One of them is, if only one clustering algorithm applied, is enough for a perfect grouping of images, which is what we want to reach for an Image Retrieval System. If we consider a Content Based Image Retrieval System, an interesting way of clustering that has been proposed [8] is using hierarchical clustering for grouping images into clusters based on color and after retrieving the certain group that the color of query image seems to be more similar, applying a second clustering algorithm, k-means, grouping the images of the retrieved group, as

described in *Figure 21.* In this case , k-means is used as a similarity measure, to ensure that the system will retrieve the images that are most like the query image. We wonder if this can be done with IQ-Means algorithm instead of k-means, because in this way we will achieve a better performance, as well as a better retrieval accuracy.

The second question is raised due to the disadvantage that both algorithms , unfortunately have, and this if we can find a way to assign points that remain unassigned because of the way that the inverse process of search works. This would be very helpful for our studies, because we will eliminate one disadvantage of a very fast and accurate algorithm, that will be even more accurate then.

Thirdly, a really useful topic to be studied further, is the dimension decomposition, which is a very critical step for our algorithm performance. As the large number of code-words makes the algorithm slower, we would like to have a better separation of the space that we are searching for , in order to generate as little code-words as possible, but without losing accuracy.

Moreover, we can study the applications of IQ-means in other fields, such as medicine. For instance, if we have stored in large database images of an organ of patients that have a certain infection, as well as images of organs that are completely healthy, we can cluster with IQ-means and retrieve in no time the images of the patients that most probably have the infection.

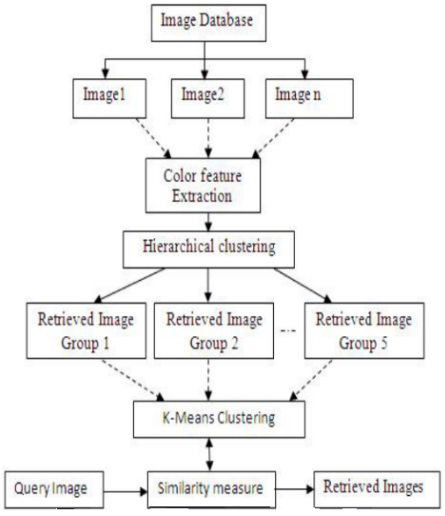


Figure 23: Hierarchical Clustering + K-Means for Content Based Image Retrieval System ,Source: [8]

8 Conclusions

In this paper , after studying some of the most known clustering algorithms, we conclude that for Image Retrieval Systems, wich contain a large scale of images, most appropriate algorithms to use are Scalable k-means by Ranked Retrieval, and for an even better performance, IQ-means. We also studied the effect that feature vectors based on different features of images, have upon our clustering. If the goal of the Image retrieval System is to return images based on their color, the feature vectors that we will extract should contain the information of color, and a clustering algorithm sould be applied to divide them into groups. As a result when the user will insert a query in the form of text or an image, it will search only for images that belong in

one group. The same procedure is followed if the system distinguishes images based on their content, but this time the use of invariant features that we had proposed wiil be really helpful.

References

[1] Yannis Avrithis, Yannis Kalantidis, Evangelos Anagnostopoulos and Ioannis Z. Emiris. *Web-scale image clustering revisited*. University of Athens, ‡Yahoo! Labs

[2] Andrei Broder, Lluis Garcia-Pueyo, Vanja Josifovski, Sergei Vassilvitskii, Srihari Venkatesan. *Scalable K-Means by Ranked Retrieval*.

[3] Artem Babenko and Victor Lempitsky. *The Inverted Multi-Index*. National Research University Higher School of Economics, Skolkovo Institute of Science and Technology.

[4] A. Malcom Marshall and Dr.S.Gunasekaran. *A Survey on Image Retrieval Methods*.

[5] Alberto Fernández Villán. *Mastering OpenCV 4 with Python*.

[6] Thomas Deselaers, Daniel Keysers, and Hermann Ney. *Clustering visually similar images to improve image search engines*.

[7] Yunchao Gong, Marcin Pawlowski, Fei Yang, Louis Brandy, Lubomir Boundev, Rob Fergus. *Web Scale Photo Hash Clustering on A Single Machine*.

[8] V.S.V.S. MURTHY, E.VAMSIDHAR, J.N.V.R. SWARUP KUMAR, P.SANKARA RAO *Content Based Image Retrieval using Hierarchical and K-Means Clustering Techniques*.

[9] Dr. P. Nithya, B. Uma Maheswari and A. Nandini *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES AND RESEARCH TECHNOLOGY, A STUDY ON IMAGE MINING TECHNIQUES, FRAMEWORK AND APPLICATIONS*.

[10] Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, Jason Zien *Efficient Query Evaluation using a Two-Level Retrieval Process*.

[11] Ravindra S. Hegadi *Image Processing: Research Opportunities and Challenges*.

[12] James Philbin , Ondˇrej Chum , Michael Isard , Josef Sivic and Andrew Zisserman *Object retrieval with large vocabularies and fast spatial matching*.