



ALGORITHMS FINAL PROJECT

Image Recognition & Retrieval System (IRAR)

Presented By:

Rajat Agrawal

SUMMARY

The report summarizes the analysis performed on image data set to find similar images from the given input query.

Since, we have our image data set as unlabeled, we have used Un supervised machine learning algorithm to produce result.

The problem presented is divided into 5 sections:

Section 1: Extract Features

- Extract Features based on Color/ Texture/Edges
- Create a Feature File

Section 2: Index the Images

- Implement Indexing techniques to store Image object in Hash Map
- Use Hashing techniques

Section 3: Apply Machine Learning Algorithm

- Clustering using K –Means
- Clustering using K- Nearest Neighbor
- Based on Distance Function (Euclidean Distance)

Section 4: Search Image

- Extract Features of the image to be compared

Section 5: Display Best Results

INTRODUCTION

Digital image databases have been widely used in recent decades. As a result the need for efficient storage and retrieval methods for large image databases has been increased. Image searching is a routine task in these databases for which there are two main approaches:

- Uses image textual metadata
- Uses image content information

The first approach needs human to describe every image in the database; it seems to be impossible for very large collections, unless there are some reliable automatic image tagging or image annotation methods.

Thus, **Image Recognition & Retrieval (IRAR)** is the main solution for this problem. IRAR has a wide variety of applications in various domains like medical and crime . In a IRAR system, digital image processing techniques are used to extract a feature vector based on low level properties of the image, like color, shape and texture.

IMAGE DATASET

For training our model we have used a database that contains various images across different domains and widely used for creating a machine learning model.

Total Images present in image database is around 10,000 but we have trained our model on 1,000 images.

These image databases comprises of various image data set like:

- Flowers
- People
- Building
- Objects
- Scenery
- Food

PART 1 : FEATURE EXTRACTION

Feature Extraction is the key concept in this project. A certain number of features for each image are extracted, describing its high level content information. Then, per the similarity of these vectors, we can compare two specific images to each other.

There are two methods in this class which extracts the features:

- **extractSingleImage:** which extracts the features for a single image. It is used to extract the features of the input image.
- **extractAll:** which is used to extract the features for all of database images. It takes about half an hour for this method to complete and stores the result table in a text file named: "Features.txt"

The first step is the Color Extraction. Then the edge map of the given image is calculated. It is implemented in Canny Edge Detector class. The last step is Texture Extraction.

0.3186456418504902	0.2866097094691471	1.1104481070974226	0.27279276131025326	0.25171996225745275	1.131943291077445
0.22267503925398285	0.2238426793016338	1.209881998247813	0.14381277677623197	0.2182202750841706	1.4502472701545002
0.2742775082017488	0.24752610966123015	1.1110645063396187	0.3362007570778284	0.29366547871000737	1.0791117906756242
0.07518284315321135	0.002987851573676068	0.002963866526639678	0.002526139418225552	0.07444872939213325	0.0037219653347541617
0.0036979802877177715	0.0017920256571474585	0.07499695903867931	0.003477831820276616	0.003477831820276616	0.003477831820276616
0.002036159171625004	0.07383539747505984	0.004311312204791184	0.0043352972518275745	0.0011786937400740452	0.07353130134299131
0.004339580295941216	0.004287327157754794	0.0011744106959604042	0.07314668398158633	0.004506619016373221	0.004506619016373221
0.004454365878186799	0.0010073719755283991	0.07352616169005494	0.004729337310282561	0.004729337310282561	0.004729337310282561
7.846536816190592E-4	0.07283573497893599	0.004765314880837147	0.004817568019023569	6.964229728780515E-4	0.07288541829065423
0.004684793651500693	0.004605129030986968	8.291973404009272E-4	0.07250679719100835	0.004629970686846086	0.004629970686846086
0.004550306066332361	8.840203050555339E-4	0.07307986849341354	0.004956338648305542	0.004956338648305542	0.004956338648305542

Feature Extraction Procedure:

- The first step is the Color Extraction. It computes the feature vector consisting of 18 real numbers for each image. Please note that for each of the features, a normalization step is provided to fit the numbers between 0-10.
- Then the edge map of the given image is calculated. It is implemented in Canny Edge Detector class. It provides a binary image which consists of just zeros and ones. So, further computations will be much easier.
- The last step is Texture Extraction. It uses co-occurrence matrix to extract texture features, and provides 48 real numbers as feature vector.
- At last, these two vectors are concatenated to each other and a 66-length feature vector for each image is computed

As we will be comparing the features with different input images, it would not be feasible to generate features of all the 1000 images again for every search.

To optimize this process, we extract the features of all the images and store these results in a Hash Map with value as Image object consisting of various image attributes namely:

- Image Name
- Image Path
- Image Hash Value

- Image Features

We have chosen our data structure as Hash Map over here because of the following reason:

Lookup process is at the heart of Hash Map and almost all the complexity of hash Map lies here. The lookup process consists of 2 steps:

Step# 1: Quickly determine the bucket number in which this element may reside (using key.hashCode()).

Step# 2: Go over the mini-list and return the element that matches the key (using key.equals()).

In the worst case, a hashMap reduces to a linkedList.

PART 2: INDEXING ON IMAGES

The MD5 algorithm is a widely used hash function producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption.

In our project, I have used hash value of Image path value as Key and Image object as value and store this into a Hash Map.

As we have large amount of Image to be hashed, we need to make sure that we are not Indexing all the image on each image search. To save our time, I have store the Hash table result and on every search, we I will check whether the Hash table is empty or not. If empty, I will de serialize my hash map and load it into my Hash Map.

```

        ImageAttr img = new ImageAttr();
        img.setImageName("img-" + j);
        img.setImagePath("C:\\Users\\Rajat\\Desktop\\Final\\sts-bundle\\sts-3.6.4.RELEASE\\image\\"
            + j + ".jpg");
        img.setFeatures(doubleArray);
        img.setHashValue(j);
        ImageDB.getImageDB().put(j, img);
        System.out.println(j);
        j++;
    }
    reader.close();
    serializeHashMap(ImageDB.getImageDB());
    printMap(ImageDB.getImageDB());
    infoBox("Images have been Indexed Now !", "Indexed");
}

public void serializeHashMap(Map NewMap) {
    try {
        try (FileOutputStream fos = new FileOutputStream(
            "C:\\Users\\Rajat\\Desktop\\Final\\sts-bundle\\sts-3.6.4.RELEASE\\hashmap.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(NewMap);
        }
        // System.out.println("Serialized HashMap data is saved in hashmap.ser");
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

```

PART 3 : SELECTING MACHINE LEARNING ALGORITHM

For this project, I have used three different scenario to test my result. Since, the problem comprises of pattern recognition, I have divided my image data set into training and testing data set.

User have option to select one of the following Algorithm :

K – Means Clustering :

k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

CLUSTERS FORMED

```
[com.edu.myneu.pojo.ImageAttr@e36c464]
[com.edu.myneu.pojo.ImageAttr@2d978886, com.edu.myneu.pojo.ImageAttr@37842ccc, com.edu.myneu.pojo.ImageAttr@4699218d, com.edu.myneu.pojo.ImageAttr@6f69
[com.edu.myneu.pojo.ImageAttr@4b7b53f2, com.edu.myneu.pojo.ImageAttr@dc45b95, com.edu.myneu.pojo.ImageAttr@56589a42, com.edu.myneu.pojo.ImageAttr@52f78
[com.edu.myneu.pojo.ImageAttr@2cfd6862, com.edu.myneu.pojo.ImageAttr@30d01101, com.edu.myneu.pojo.ImageAttr@2c0c71b0, com.edu.myneu.pojo.ImageAttr@64fb
[com.edu.myneu.pojo.ImageAttr@10e88aac, com.edu.myneu.pojo.ImageAttr@12e82ae0, com.edu.myneu.pojo.ImageAttr@4856eld2, com.edu.myneu.pojo.ImageAttr@4003
[com.edu.myneu.pojo.ImageAttr@29ceabdb, com.edu.myneu.pojo.ImageAttr@3f74cc4e, com.edu.myneu.pojo.ImageAttr@d11356e, com.edu.myneu.pojo.ImageAttr@64cac
[com.edu.myneu.pojo.ImageAttr@2467ee4d, com.edu.myneu.pojo.ImageAttr@234e13e7, com.edu.myneu.pojo.ImageAttr@5afcb4cc, com.edu.myneu.pojo.ImageAttr@84c1e
[com.edu.myneu.pojo.ImageAttr@7af76743, com.edu.myneu.pojo.ImageAttr@9c9ec7, com.edu.myneu.pojo.ImageAttr@627c1f3b, com.edu.myneu.pojo.ImageAttr@42e68
[com.edu.myneu.pojo.ImageAttr@4a1f826d, com.edu.myneu.pojo.ImageAttr@5aabe81f, com.edu.myneu.pojo.ImageAttr@4c1479ba, com.edu.myneu.pojo.ImageAttr@5443
[com.edu.myneu.pojo.ImageAttr@3c96adf8, com.edu.myneu.pojo.ImageAttr@487793db, com.edu.myneu.pojo.ImageAttr@228c3408, com.edu.myneu.pojo.ImageAttr@22ac
[com.edu.myneu.pojo.ImageAttr@13f23e56, com.edu.myneu.pojo.ImageAttr@4da8a5e5, com.edu.myneu.pojo.ImageAttr@827afd4f0, com.edu.myneu.pojo.ImageAttr@8c4ec
[com.edu.myneu.pojo.ImageAttr@42ad3f8c, com.edu.myneu.pojo.ImageAttr@7c272286, com.edu.myneu.pojo.ImageAttr@6d4bcb17, com.edu.myneu.pojo.ImageAttr@b028
[com.edu.myneu.pojo.ImageAttr@6a7a8cb9, com.edu.myneu.pojo.ImageAttr@13acfc4c, com.edu.myneu.pojo.ImageAttr@41e7ebab, com.edu.myneu.pojo.ImageAttr@6121
[com.edu.myneu.pojo.ImageAttr@7b49e2bb, com.edu.myneu.pojo.ImageAttr@2fc4a239, com.edu.myneu.pojo.ImageAttr@167664af, com.edu.myneu.pojo.ImageAttr@36cc
[com.edu.myneu.pojo.ImageAttr@8d7b381, com.edu.myneu.pojo.ImageAttr@949deb0, com.edu.myneu.pojo.ImageAttr@50baf993, com.edu.myneu.pojo.ImageAttr@23574b
[com.edu.myneu.pojo.ImageAttr@49b96230, com.edu.myneu.pojo.ImageAttr@29dd6320, com.edu.myneu.pojo.ImageAttr@596df59, com.edu.myneu.pojo.ImageAttr@74214
[com.edu.myneu.pojo.ImageAttr@17495c9e, com.edu.myneu.pojo.ImageAttr@515f9503, com.edu.myneu.pojo.ImageAttr@5a3021b1, com.edu.myneu.pojo.ImageAttr@df41
[com.edu.myneu.pojo.ImageAttr@1dfcf773, com.edu.myneu.pojo.ImageAttr@37f5e68, com.edu.myneu.pojo.ImageAttr@1f0effa3, com.edu.myneu.pojo.ImageAttr@11b13
[com.edu.myneu.pojo.ImageAttr@6ef4ebd, com.edu.myneu.pojo.ImageAttr@491a5dd9, com.edu.myneu.pojo.ImageAttr@6207770d, com.edu.myneu.pojo.ImageAttr@58192
[com.edu.myneu.pojo.ImageAttr@450f9073, com.edu.myneu.pojo.ImageAttr@449171, com.edu.myneu.pojo.ImageAttr@3989224, com.edu.myneu.pojo.ImageAttr@15a27b5
[com.edu.myneu.pojo.ImageAttr@7371ece, com.edu.myneu.pojo.ImageAttr@33c76a15, com.edu.myneu.pojo.ImageAttr@6905a341, com.edu.myneu.pojo.ImageAttr@71213
[com.edu.myneu.pojo.ImageAttr@55243942, com.edu.myneu.pojo.ImageAttr@3d2b7b9, com.edu.myneu.pojo.ImageAttr@6471e20e, com.edu.myneu.pojo.ImageAttr@70b0
[com.edu.myneu.pojo.ImageAttr@38f8dc2b, com.edu.myneu.pojo.ImageAttr@59b85ce0, com.edu.myneu.pojo.ImageAttr@2eb9f428, com.edu.myneu.pojo.ImageAttr@324c8
[com.edu.myneu.pojo.ImageAttr@7685501b, com.edu.myneu.pojo.ImageAttr@2a4e5967, com.edu.myneu.pojo.ImageAttr@7dcf8ee3, com.edu.myneu.pojo.ImageAttr@45ae
[com.edu.myneu.pojo.ImageAttr@52b3135e, com.edu.myneu.pojo.ImageAttr@6abcaabc, com.edu.myneu.pojo.ImageAttr@86d5363, com.edu.myneu.pojo.ImageAttr@417d8
[com.edu.myneu.pojo.ImageAttr@7895d11c, com.edu.myneu.pojo.ImageAttr@33d3c31d, com.edu.myneu.pojo.ImageAttr@13aebf80, com.edu.myneu.pojo.ImageAttr@359e
[com.edu.myneu.pojo.ImageAttr@2ee2c067, com.edu.myneu.pojo.ImageAttr@28c5c23d, com.edu.myneu.pojo.ImageAttr@4f5347b4, com.edu.myneu.pojo.ImageAttr@6488
[com.edu.myneu.pojo.ImageAttr@53b5c3cc, com.edu.myneu.pojo.ImageAttr@4249b103, com.edu.myneu.pojo.ImageAttr@74045df4, com.edu.myneu.pojo.ImageAttr@4ab4
[com.edu.myneu.pojo.ImageAttr@4b0a833a, com.edu.myneu.pojo.ImageAttr@1f2d7b53, com.edu.myneu.pojo.ImageAttr@62fb9622, com.edu.myneu.pojo.ImageAttr@733d
[com.edu.myneu.pojo.ImageAttr@5c4d26f9, com.edu.myneu.pojo.ImageAttr@4d39d4c6, com.edu.myneu.pojo.ImageAttr@fc03ac6, com.edu.myneu.pojo.ImageAttr@1552a
[com.edu.myneu.pojo.ImageAttr@42bed00f, com.edu.myneu.pojo.ImageAttr@7d4dab04, com.edu.myneu.pojo.ImageAttr@7616d3e0, com.edu.myneu.pojo.ImageAttr@54b0
[com.edu.myneu.pojo.ImageAttr@2a0dc492, com.edu.myneu.pojo.ImageAttr@6de462cf, com.edu.myneu.pojo.ImageAttr@2d133c66, com.edu.myneu.pojo.ImageAttr@49e2
[com.edu.myneu.pojo.ImageAttr@500d17b7, com.edu.myneu.pojo.ImageAttr@b901870, com.edu.myneu.pojo.ImageAttr@213466fe, com.edu.myneu.pojo.ImageAttr@7755b
[com.edu.myneu.pojo.ImageAttr@7a203667, com.edu.myneu.pojo.ImageAttr@58d3e2d4, com.edu.myneu.pojo.ImageAttr@3ed0f9db, com.edu.myneu.pojo.ImageAttr@9c3b
```

===== BISECTING KMEANS - SPMF 2.09 - STATS =====

Distance function: euclidian

Total time ~: 75210 ms

SSE (Sum of Squared Errors) (lower is better) : 26.826391045824327

Max memory:137.3939437866211 mb

=====

73.17360895417568

```

    if (checkedValue.equalsIgnoreCase("KMeans")) {
        int kM = 100;
        DistanceFunction distanceFunction = new EuclidDistance();
        BisectingKMeans algoKMeans = new BisectingKMeans();
        List<ClusterWithMean> clusters = algoKMeans
            .runAlgorithm(ImageDB.getImageDB(), kM,
                distanceFunction, 100);
        algoKMeans.printStatistics();
        double error = ClustersEvaluation.calculateSSE(
            clusters, distanceFunction);
        accuracy = 100 - error;
        System.out.println(accuracy);
        for (int i = 0; i < clusters.size(); i++) {
            for (ImageAttr img : clusters.get(i).getVectors()) {
                if (img.getImageName().contains(
                    inputImage.getImageName())) {
                    for (ImageAttr images : clusters.get(i)
                        .getVectors()) {
                        similar.add(images);
                    }
                }
            }
        }
    }
}

```

K- Nearest Neighbors :

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression

We have used KD Tree as a data structure to store the result and get the nearest neighbors.

```

if (checkedValue.equalsIgnoreCase("KNearset")) {
    KDTree tree = new KDTree();
    List<ImageAttr> images = new ArrayList<ImageAttr>(
        ImageDB.getImageDB().values());
    tree.buildtree(images);
    System.out.println("\nTREE: \n" + tree.toString()
        + " \n\n Number of elements in tree: "
        + tree.size());

    int KNN = 10;
    RBTREE<NearestNeighbour> result = tree.knearest(
        inputImage, KNN);
    System.out.println(result.toString());
    Iterator<NearestNeighbour> iterator = result.iterator();
    while (iterator.hasNext()) {
        NearestNeighbour point = iterator.next();
        ImageAttr img = point.image;
        similar.add(img);
    }
}

```

Euclidean Distance:

Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space. We display top 10 result based on the distance calculated.

PART 4: RETRIVAL

First, it loads the "Features.txt". Then it gets the input image number, between 0 - 1000, and computes the feature vector for this single image.

In "retrieve" method, the feature vector for this single image is compared to all other feature vectors, and they are sorted in increasing order, based on the ML method selected.

After all, the first ten images with the least distance will be given as the output.

In K -Means, results are given based in the clusters formed and the cluster in which the image is present.

In KNN, we use k-d tree, or k-dimensional tree, is a data structure used in computer science for organizing some number of points in a space with k dimensions. It is a binary search tree with other constraints imposed on it. K-d trees are very useful for range and nearest neighbor searches.

PART 5: WEBSITE

Upload the Image to be Searched:

Image Recognition & Retrieval (IRAR)

[Home](#)[Upload](#)[Features](#)[Indexing](#)[About IRAR](#)



Upload Image

Browse Image


☐ K-Nearest Neighbours☐ K- Means Clustering☐ Normal

Search Image

ADD New Images:

Image Recognition & Retrieval (IRAR)

[Home](#)[Upload](#)[Features](#)[Indexing](#)[About IRAR](#)




Upload Image

Browse Image

Upload Image

Choose the Algorithm:

Northeastern University




Upload Image

Browse Image 410.jpg

☐ K-Nearest Neighbours ☐ K- Means Clustering ☐ Normal

Search Image




Results:

Northeastern University




Accuracy : 61.948676340361914

Image Searched

Image Path: C:\Users\Rajat\Desktop\Final\sts-bundle\sts-3.6.4.RELEASE\image\429.jpg



Result Images

Northeastern University

Image Seached

Image Path: C:\Users\Rajat\Desktop\Finalists-bundle\sts-3.6.4.RELEASE\image\994.jpg



Result Images



Thank you.