Similar Leaves

William Biondi

July 15, 2023

Abstract

This project work aims to build a detector of pairs of similar images over a dataset of images portraying leaves. I will use a pre-trained algorithm to extract features among the images and use Locality Sensitive Hashing to optimize this high-dimensional space research. Finally, we compute similarity among all the pairs and carry out results

1 Introduction

The identification of similar items in the domain of images presents unique challenges and some of them are specifically related to the leaves' nature :

1. Variations in lighting conditions
2. Presence of noise
3. High dimensionality of data
4. Limited computational resources

As a consequence, we need an accurate and efficient method to identify similar items in this framework.

1.1 Objectives

The objective of this project is to build an efficient and accurate detector for image similarity and this will be pursued by using a pre-trained algorithm to extract features from each image. These features will be processed using Locality Sensitive Hashing to optimize search. Everything will be based on resilient distributed datasets since we want our solution to be scalable to higher quantities of data.

1.2 Dataset

The dataset for this project is composed of 4503 jpeg images with a dimension of 6000 * 4000 pixels. This image dataset is structured into 11 types of plants. Each type of plant has distinct pictures of healthy and diseased conditions. Before diving into feature extraction, I proceed to resize images to 480 * 320 pixels and delete unlabeled images and diseased leaves images. Finally, the resulting dataset will count 2273 images with classes distributed as shown in Figure 1

2 Methodology

The methodology of this project consists of these macro steps:

1. Feature extraction with ORB
2. Minhashing
3. Optimize similarity search with LSH

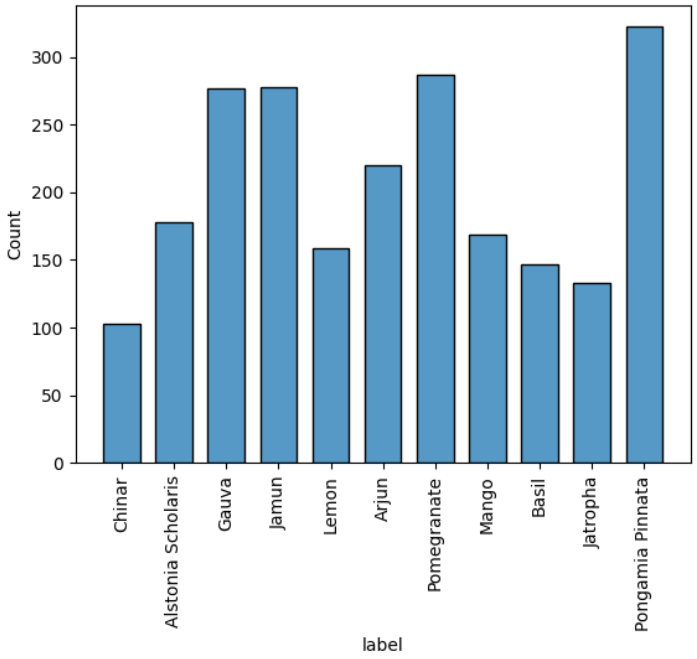

Figure 1: Images per Plant type

2.1 Feature extraction

Starting from the dataset of images we'll proceed to extract keypoints and descriptors for each image using ORB. ORB combines two algorithms called FAST and BRIEF: the former is designed to find keypoints while the latter is designed to find binary descriptors for each keypoint. In this context, this algorithm allows the detection of relevant regions of the image and their associated characteristics that will be used to compute similarities. So our RDD will have tuples composed of keypoints, descriptors, and the image associated with them.

2.2 Minhashing

The descriptors that we obtained for each image are high dimensional and sparse if we represent them in a characteristic matrix, so instead of them we need a compact representation for them that can also preserve their characteristics in order to compute the similarity.

2.2.1 Hash Signatures

Starting from the RDD with feature vectors we find the maximum value among all the descriptors lists for each image.Then, I generate n hash functions of the form

$$ax + b \mod maxvalue$$

where a and b are random integers and x is the descriptor value given as an input. After generating the functions we compute the signatures for each image by calculating $h_i(j)$ where j is j-th descriptor of the image and i the associated i-th hash function which is obtained with

$$i = j \mod n$$

So the vector $[h_1(j_1), ..., h_n(j_r)]$ is the hash signature of the image.

2.3 Signature Matrix

Starting from taking all the distinct values that we have in all the signatures for each we initialize a signature vector. We sort these distinct values and for each image which is a column we set to 1 all

the values in the image's signature vector that are present in the hash signature. For instance, if an image has the descriptor 29 in its hash signature, the corresponding index in the signature vector that corresponds to the descriptor 29 will be set to 1

2.4 Locality Sensitive Hashing

At this stage, Minhashing improved the space problem by compressing feature vectors into signature vectors. However, there's still a problem with efficiency in finding pairs with the best similarity: although parallel computing can be a feasible solution in time we're interested in finding only the most likely to be similar pairs. Locality Sensitive Hashing provides such focus by hashing the images such that the most likely to be similar are in the same group. From the signature matrix computed with Minhashing, we divide it into b bands with the number of rows r each. For each band we use

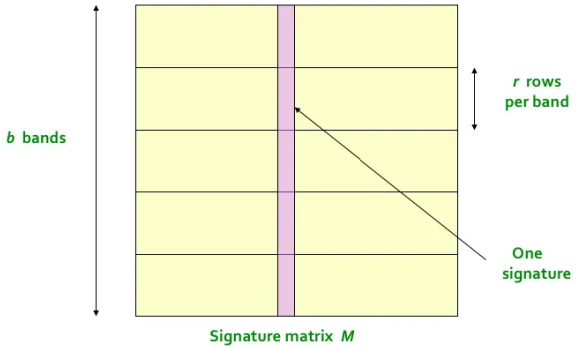

Figure 2: Bands in the signature matrix

a predetermined number of hash functions h that take each band item as an input, in this case, a subsequence of the column vector, of the size r, and assign it to a hash bucket. Hashing works similarly to the minhashing phase in Section 2.2.1 but in this case we have a preliminary step: the input is a binary vector that is converted into a decimal which we denote as x then the module is performed on the number of hash functions.

$$hashbucket = ax + b \mod h$$

2.5 Candidate Pairs

Once hash buckets were obtained, the internal combinations of the images that are present in each bucket will be considered a candidate pair since they collide for at least 1 band: this resulted in an enormous reduction of the similarities to compute compared to all the possible combinations

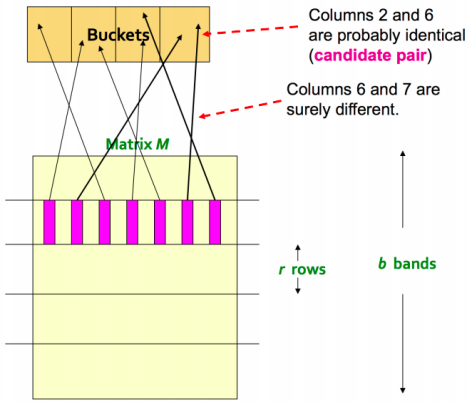

Figure 3: How a candidate pair is determined

2.5.1 Jaccard Similarity

Finally for each candidate pair the Jaccard similarity where A and B are the full signature vectors of the examined pair: the similarity is computed based on the cardinality of the intersection divided by the union.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In this case, the intersection is determined by summing the AND operation results on the same index component; likewise, the union is determined by summing the OR operation results on the same index component.

3 Results

In this section, I will report the result of the detector built on the hyperparameters indicated in the following table To compute metrics I considered the leaves which belong to the same type of plant as

N° Minhash Functions	N° Bands	N° LSH functions
500	50	300

Table 1: Parameters of Minhash-LSH detector

similar, dissimilar otherwise. By using a threshold I examined the similarities and if it is above the threshold, the pair is classified as similar, if it is below that threshold it will be classified as dissimilar. Furthermore, I show how the detector behaves inside plant classes by examining the similarities of the candidate pairs that belong exclusively to the same plant type

3.1 Overall Results

The detector metrics shown in Table 2 indicate a low precision since it considered similar a high number of pairs of images that do not belong to the same plant type: I expected this result since the dataset is composed of items that have a considerable number of features in common; the example in Figure 4 shows that even if the pair involves dissimilar items, the detector is still able to find features in common. An interesting result is the high recall which indicates that the detector is able to find relevant features

Precision	Recall	F1
10.63%	89%	0.19

Table 2: Metrics of the detector overall candidate pairs

Figure 4: A pair of dissimilar leaves

3.2 Plant type Results

The high recall in the previous section suggested the ability to detect relevant features: in this analysis, I excluded the pairs which involved different types of plants, group pairs by the type of plant in common,

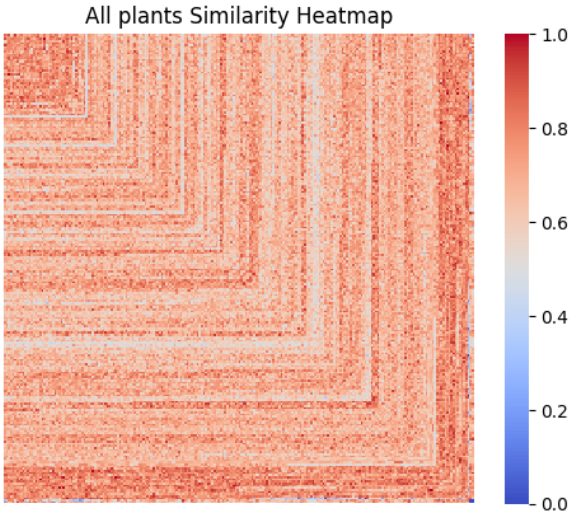

Figure 5: Overall heatmap of similarities

and compute metrics for each plant group.
Note that the precision is not considered since we're working on pairs that in fact are always similar These results indicate that the detector has high performance in detecting true similar pairs in general

Plant Type	Recall	F1
Mango	97.20%	0.99
Jamun	95.56%	0.98
Gauva	96.85%	0.98
Chinar	100%	1.0
Basil	100%	1.0
Pomegranate	72.84%	0.84
Arjun	83.88%	0.91
Alstonia Scholaris	97.71%	0.99
Pongamia Pinnata	90.74%	0.95
Jatropha	99.13%	1.0
Lemon	76.89%	0.87

Table 3: Metrics of the detector for each plant type group pairs

with a few exceptions such as Pomegranate and Lemon that can be clearly visualized in Table 3 and in the heatmaps in Figure 6

4 Conclusions and further work

The technique combinations of Minhashing and Locality Sensitive Hashing resulted efficient in finding similar items by filtering only relevant combinations by using an approximate representation in order to make feasible computations on large-scale datasets. Besides, further work can be done by the side of dissimilarity: the detector considers more of the features in common between images rather than the differences so working in it to counterbalance this side will improve the detector's performance.

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offenses in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

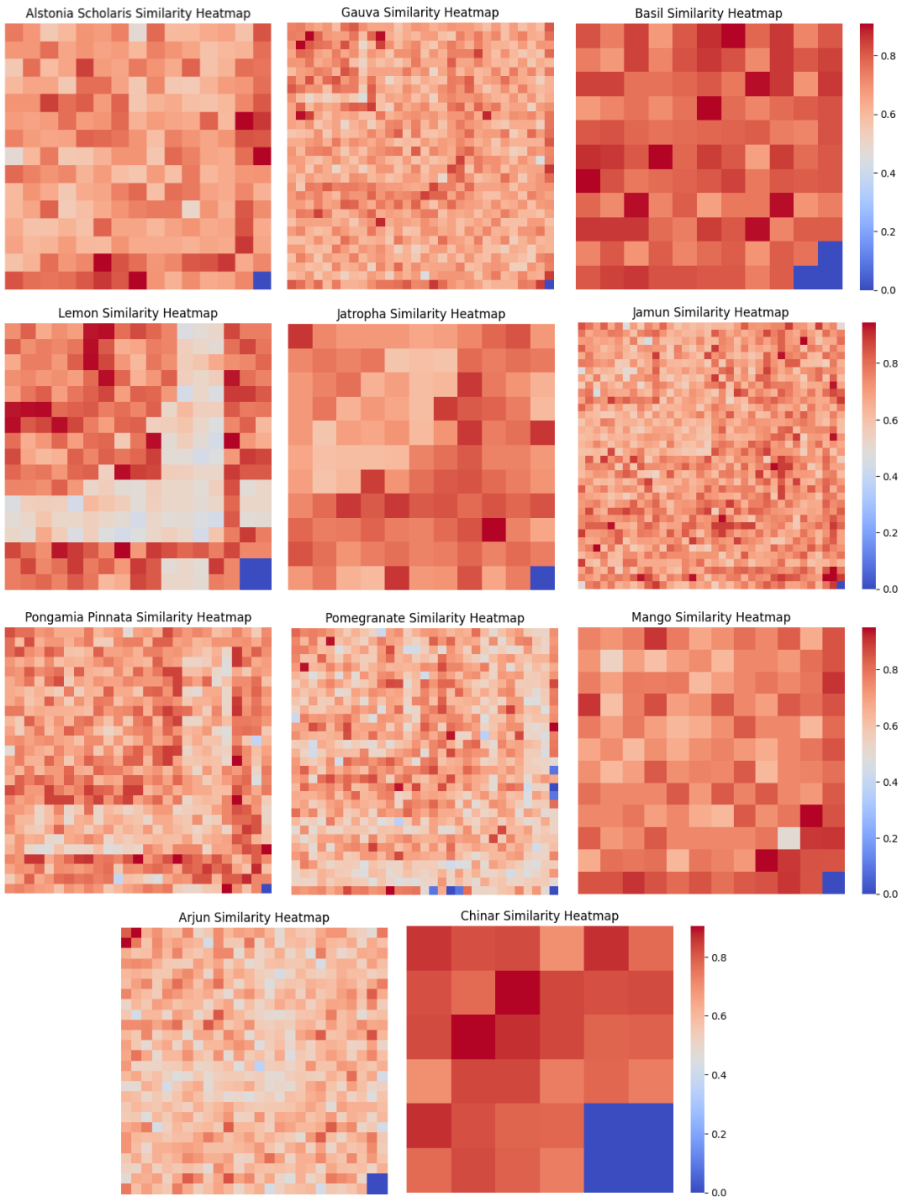

Figure 6: Heatmap of similarities within plant types