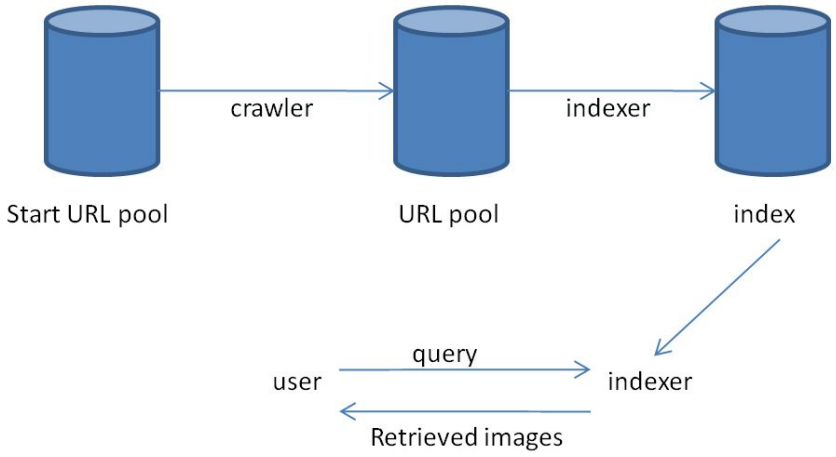
Web Search Engines

Course Project

Yixian Fu yf899 N14435631

This web search engine is a specialized image search engine, which can be used by images by designers, software engineers, etc. to search icon images. It contains 5 important parts: crawler, indexer, retriever, LIRE indexer, LIRE searcher. See README file for information about compiling and running.

1 Content-Based Image Retrieval



1.1 Crawler

I use some high-frequency keywords and some hub websites with high-quality icon images to construct the start URL lists. The crawler will generate a URL list after crawling, instead of the whole websites text, to reduce the cost of space. The Indexer then use this URL list to index the corresponding web pages.

1.2 Indexer

The most important thing for indexer is to tag Icon Images Correctly. For a given image, I use its alt attribute, its suffix, its context, its current URL, and its image source URL to tag the image properly.

Normally, the alt attribute of an image will describe the content of the image, which is the most important factor for image tagging. By check the suffix of the URL, the search engine eliminate some bad URLs. Context means text near the image, which is the content of the parent html tags of image tags. Current URL and image source URL might contain some useful information too.
When implement it, I use jsoup to extract the html information in order to tag the images properly.

1.3 Retriever

The Retriever get the query of the user, then return the corresponding web pages in descending order of the likelihood.
The interesting part is to do some near-duplicate detection for the returned images. This will be discussed in following section.

1.4 Updating Index

1.4.1 An Easy Solution

An easy solution to update index is to use linux crontab command to re-crawl.
For example, if I want to update index every Sunday at 3:00 AM, I could:

type "crontab -e"
Then write:
0 3 * * 7 yes | rm urls
0 3 * * 7 java -cp ./lib/*:. Crawler -m 10000
0 3 * * 7 java -cp ./lib/*:. Indexer

1.4.2 Next step

The disadvantage of the easy solution is it will update the whole index, though it is acceptable when the data is not too big. However, when data grows, it will be expensive to update the whole index.
To solve this problem, we could compute the signatures for all the icon images indexed by the Indexer. Then given an image that already be indexed, we could do the following things:
1. Resize the image to 64 * 64 resolution.
2. compute the signatures for the image
3. Update the index for the image if the new signature are different from the original one.

1.5 Evaluation

Content-based image retrieval is quite a success. The following graph list several query

words and their precision. It is not a systematic evaluation, but by trying some hot icon image keywords, the performance seems good.

Query	phone	sun	chat	download	search
Number of returned images	50	24	50	50	50
Number of relevant images	50	20	47	49	44
Precision	1.0	0.83	0.94	0.98	0.88

However, sometimes, the returned images could contain some relevant image which are not a real “icon”. An observation is that all icon images are in PNG format. So I use a pretty straightforward method to solve this problem by checking the suffix.
My next step is to use image characteristics to filter non-icon images. Since icon image are generally simple. So we could filter those images with many colors and complex graphics.

2 Using Image to Retrieve Image

2.1 My Implementation

To implement searching with image, I use the following method:

1) Maintain a standard icon image gallery. Every images in it is highly dissimilar with other images.

2) Use the image gallery to build a image index using APIs provided by LIRE - Lucene Image Retrieval.

3) Get the input image given by the user, then use LIRE APIs to retrieve the most likely image.

4) use the keyword corresponding to the most likely image to search.

I will discuss LIRE project in next section.

2.2 LIRE Indexer and LIRE Searcher

LIRE is a Java library that provides a simple way to retrieve images and photos based on color and texture characteristics.

The LIRE indexer will create image index for the images of the standard icon image gallery.

The LIRE searcher will get the path to the input image, and return the keyword of the most similar image, or output a list of image path and their corresponding score if it get an “-web” flag as one of the inputs. There is a threshold used to see whether there exists a similar image or not. I used 40.0, just out of observation, and it can be discussed in the future.

2.3 Problems

The first problem we might encounter is the precision of the result. Normally, an icon image tend to be simple, so there are fewer characteristic in an icon image than in a normal image, which will decrease the precision of the image retrieved by the search engine.

And there are some problem with LIRE too. LIRE only use color and texture characteristics to build the index. So it is highly possible, given an image of the Sun, LIRE will give a low score, which means high likelihood, to a person with a round face and yellow skin.

And what if the input image are vague or with low resolution? In this case, the returned images could be with very low precision.

2.4 Evaluation

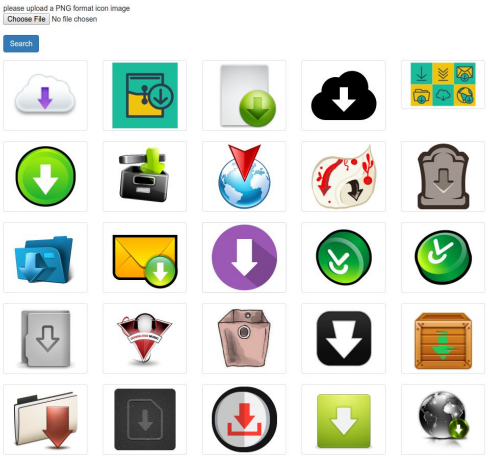
This part is not very satisfying. The precision of Image-based search is highly dependent on the quality of the standard icon image gallery. Since all icon images are pretty

simple, different icon images might be in the same shape and same color.

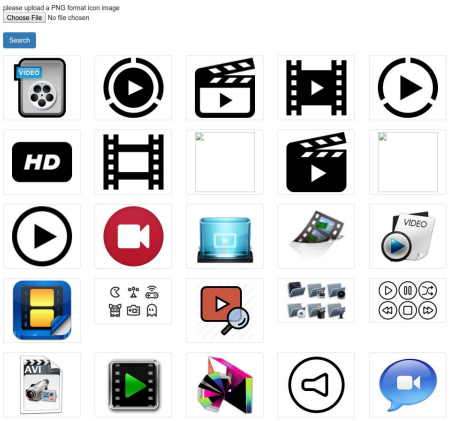
2.4.1 Two examples

There are two images I used as an input to search. The first one succeed, but the second one failed.

→



→



The reason why the second one failed is because the standard icon image gallery are not large enough to cover all kinds of input images. So to solve this problem, we need to use more images to indicate the same keyword, so an input won't be miss.

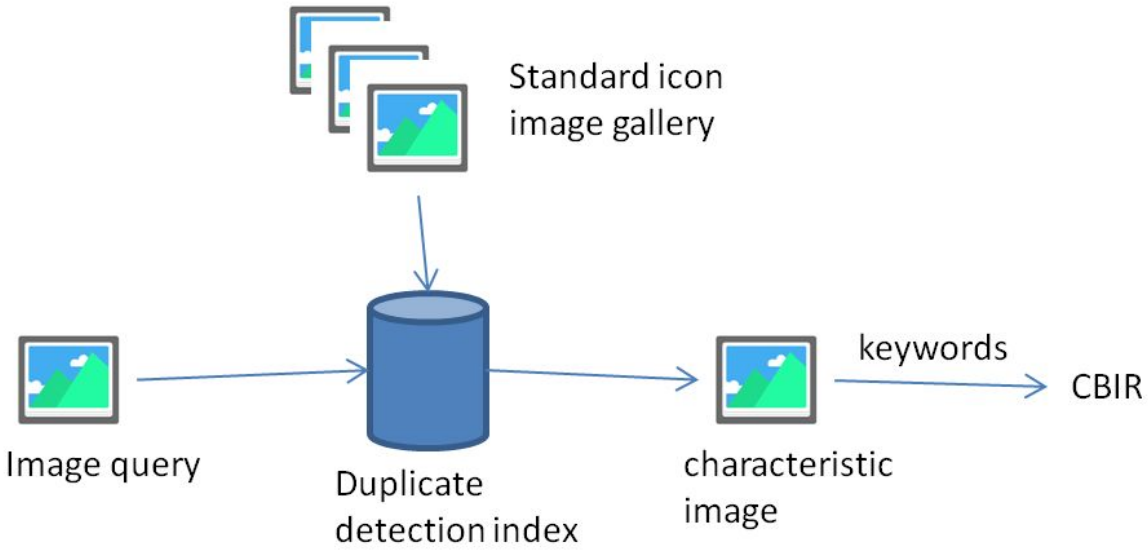
2.4.2 Problems and Solutions

Let's take a look at the following two images. Both of them are round shape, and both of them are blue. So if we use the first one as a standard icon image, and use the second one as a query image, then the system might return a page full of search icon instead of download icon.

An easy way I have tried to solve this problem is by using more than one similar images to provide the keywords. However, this approach will decrease the precision sometimes. If the user provide a download image, and the most similar image in the standard icon image gallery is a download image, this approach will decrease the precision a lot. My improvement is to set a “MAX_DIFF”. If the difference between the score of the former hit and the score of current hit, then the keyword of the current hit won’t be used.

3 Near-Duplicate Image Detection

3.1 Approach



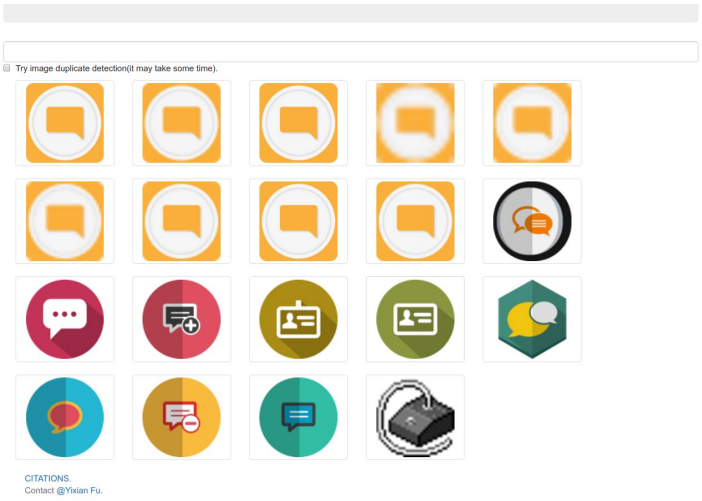
Given a image set returned by the Retriever, I use the following method to detect near-duplicate image:

1. resize all the images to 64*64 resolution.
2. Build index using the resized image set
3. If the score of the most-likely image of a given image(except itself) is greater than a threshold. I use 10.0 as the threshold, but it is purely out of observation. I plan to do some research regarding it in the future.

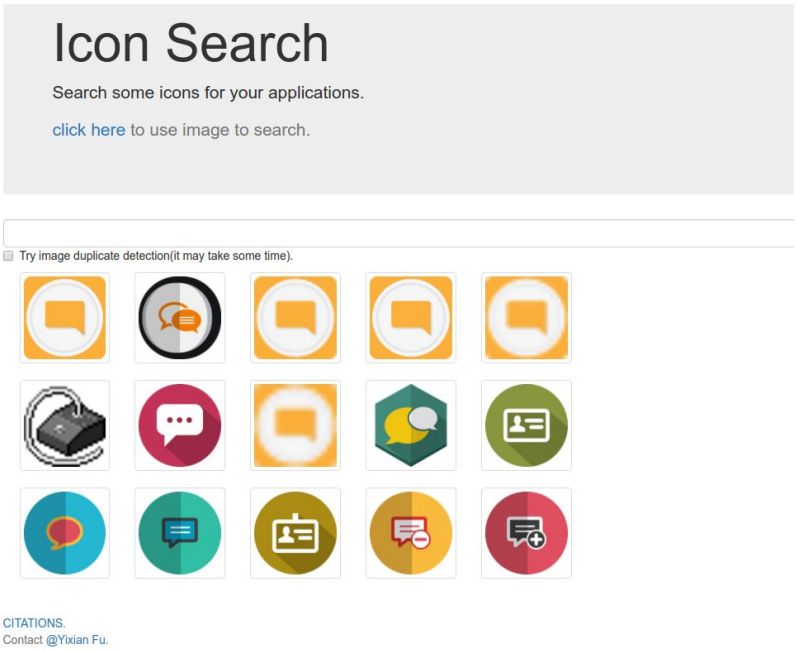
Why is it important to resize the images? The main reason is we need to handle the case the same image with different resolution, and according to the paper of A. Torralba, R. Fergus, and W. Freeman, if we resize the image to 64 * 64, there won't be a significant performance drop for the F-measure of the result.

3.2 Evaluation

When we search "msg" without checking the "try image duplicate detection" checkbox, we get the following results.



After checking the “try image duplicate detection” checkbox, we get the following results.



The method find 4 duplicate images, which seems good. But the score of other duplicate images are higher than 30.0, due to the reason I discussed in former sections.
Another problem is efficiency. On average, LIRE can process 5 to 7 images per second. So the system normally takes several seconds to remove duplicate images.

Citations

Millions thanks to following PROJECTS:
Lucene
LIRE

Millions thanks to following PAPERS:
80 Million Tiny Images: A Large Dataset for Non-parametric Object and Scene Recognition - A. Torralba, R. Fergus, and W. Freeman
Content-Based Image Retrieval - Approaches and Trends of the New Age - Ritendra Datta, Jia Li James, Z. Wang

Millions thanks to following BOOK:
Introduction to Information Retrieval - MR&S

Millions thanks to following WEBSITES:
ICONFINDER.com
ICONARCHIVE.com
FINDICONS.com