Reverse image search – Report

PGR208-1 Android Programming Exam

Introduction

In most cases, reverse image search is a subset of the information retrieval (IR) process. In the beginning, search engines like Google and Bing relied mainly on keyword search techniques that analyzed the user's text input using dictionaries with familiar words and their meanings. However, in order to extract useful information from an image, an IR process must use more complex methods than keyword searching. These techniques span from color extraction and boundary recognition to object recognition in photos (people, buildings, cars, and so on).

This exam focuses on building an Android application to perform reverse image search and organizing retrieved data in a virtual Android device (emulator) or a physical phone. It has been provided with an AWS server instance running at http://api-edu.gtl.ai/api/v1/imagesearch with four endpoints created specifically for this course. The server is constantly running a Flask Python server instance that accepts an image at one endpoint (/ upload) and performs the reverse image search on three more endpoints (/ google, /bing and /tineye).

Reverse image search is an android app created with the Kotlin programming language and the Android Studio IDE. This Android app is compatible with all Android devices that have an API level of 16 or above. The fundamental idea behind this app is to give the server a sample image and have it search for relevant stuff to that image. The sample image is what creates a search query in terms of information retrieval. Reverse image effectively eliminates the need for a user to make educated guesses about keywords that may or may not yield a correct result. Users can also utilize reverse image search to find articles connected to a given sample image or image popularity.

This app's strategy or the technique that we have used in this app, is that it allows the user to select an image from his Android device's gallery and upload it to an Amazon web service-based cloud server that runs a python script. When a user requests results based on an uploaded image, Google, Bing, and Tineye search engines return similar photos. We discovered that when the server detects items on the example image, it returns extremely comparable results.

Code structure and implementation

In this part of the report, the group will describe the code structure that has been used and the implementation of them. “MainActivity” and “FullScreenImgActivity” are the two primary activities in reverse image search. The fragments are named “InsertKotlingFrag”, “ResultsKotlinFrag” and “SavedResultKotlingFrag”. On the other note, the code structure does also contain variables and classes.

“MainActivity”
"MainActivity" is made up of three primary parts. Fragments are similar to sub-activities within a larger activity. Multiple fragments can be used on a single screen at the same time to create a single-screen program that gives a more efficient user experience.

The user is presented with two buttons: one to open the gallery and select an image, and the other to check for user permission to view the gallery. If permission has previously been given, the user uses implicit intent to navigate to the gallery and choose an image.
When a user selects a picture, a callback in fragment receives it, and we display it in the imageview widget, which is used in Android Studio to display images to users.

The second button allows the user to submit the selected image to the server using a http request, which converts the image to PNG format and then uploads it. When the server receives the image, it stores it and gives the image's downloading URL to the application.

The URL is passed to the result fragment, which then makes a get request to the other three Image search API end points. It makes a Get request to the server using Volley as a third-party library with that URL as an argument. Volley is a http library used by professional developers to make server requests. It provides an easy way to build requests and executes each request asynchronously without stopping UI components, which is the library's main benefit. To use this library, we only need to add its dependency, and then we may use it to effectively make server requests.

The first button on the result fragment displays the returned photographs in a recycler view widget, which is used in android studio to display a list of data. All of the retrieved photographs are saved in a list, and then, with the help of the recycler view adapter, they are displayed on the screen in a grid format with three images per row.

To display images in each recycler view item, we utilized a third-party library called round corner image view. This library can also be utilized by just including it as a dependency, and the benefit of doing so is that it improves the user experience by rounding the corners of picture views. The Image view widget of Android Studio, which is accessible by default, is an alternative to this library.

Another third-party library we use is the glide image processing library, which takes an image's URL and displays it asynchronously in a round corner image view without blocking UI components. An alternative to using this package is to write our own class that downloads an image from a URL and displays it in Android Studio.

The second button on the result fragment saves all of the photographs that have been downloaded to the database. Images are saved in the SQLite database, which is the app's local database provided by the Android operating system. We've designed a new class called

database that delivers database instances wherever they're needed. It creates a database table with columns for the original image as well as the results.

In the database, the user can save as many results as he wants. For each freshly saved result, the database creates a new row and assigns a unique id to it.

Saved fragment is used to display all saved database results. It retrieves all rows of the result table from the database and displays all photos associated with the result in a vertical recycler view. The sample image with all of its results is included in each recycler view item.
It also gives the user the option of pressing the delete button to remove the current entry from the database and refresh the list at runtime.

"FullScreenImgActivity"
"FullScreenImgActivity" has a single layout and is used to display images in a large size whenever the user clicks on one.
This activity takes use of the Photo view third-party library, which adds pinch-to-zoom and double-tap-to-zoom capability to image views, making it superior to Android Studio's default image view widget.

Discussion
The group's three members have included retrieval requests from all three search engines. Given that we have included retrieval requests, only Bing returns us results.

We wanted to get results from all three search engines back, in order to make the app more efficient. On such a basis, it can be said that, if the ant search engine does not return expected results, there may be another that can return the correct result and meet the user's expectation. In other words, this was not the result at the end, because only Bing returns results. This error may be due to some errors in the other two API endpoints.

If the future were to offer to work with such a project again, whether it is in a work or school context, we in the group would try to get results from maximum search engines.

Conclusion
To summarize, the reverse image search app provides a service that allows users to search for relevant material based on a sample image provided to the server end point. Which improves the accuracy of findings and meets user expectations in a more timely and consistent manner. This tool can also be useful in that it allows users to access results from numerous search engines rather than just one.

References:
https://github.com/bumptech/glide
https://github.com/Baseflow/PhotoView
https://github.com/vinc3m1/RoundedImageView
https://github.com/google/volley