Gymnázium, Praha 6, Arabská 14

Obor Programování

Ročníková práce

Detekce duplicitních a podobných fotek

Vypracoval: Josef Liška

Duben 2024

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze, dne 2. dubna 2024 Josef Liška .....

OBSAH

1. Anotace

Detekce fotek té samé scény je netriviální záležitost. Zde je řešena pomocí vyhledávání založeném na Locality sensitive hashing, konkrétně vyhledávání za pomoci hashe získaného analýzou obrázku přes Azure Vision API. Výsledkem práce je desktopová aplikace s grafickým rozhraním, která umožňuje snadný výběr a přidání fotek do databáze. U takto vybraných fotek se kontroluje, zda nejsou duplicitami fotek již v databázi uložených a pokud ano, jsou zobrazeny uživateli, aby rozhodl, které duplicitní fotky ponechat a které smazat. Využité jazyky: Java, FXML, Bash skript a SQL.

2. Zadání

Popis: Počítačová aplikace, jejímž účelem je ve větším množství fotek vyhledat skupiny fotek velmi podobných, například ta samá scéna focená vícekrát, dále nabídnout uživateli takové duplicity vymazat nebo zvolit, aby se při příštím skenu skupina nezobrazovala. Pro výtažek dat z obrázku bude použito vybrané již existující online vektorizační api.

3. Úvod

Pro porovnávání obrázků musí člověk nutně zvolit podobnostní metriku (více v další kapitole). Správné určení podobných obrázků pak záleží zejména na ní. Je-li tato metrika již zvolena, jednou z možností nalezení podobných obrázků je, při přidání obrázku do databáze, porovnat jej se všemi již uloženými obrázky na základě této metriky. Tento postup hledání podobných obrázků však vede ke kvadratické složitosti. Místo toho zde využívám vyhledávání založené na Locality sensitive hashing, které má v tomto případě lineární složitost. V další kapitole bude popsán princip vyhledávání a v následujících kapitolách popsáno fungování aplikace.

4. Locality sensitive hashing – Princip fungování

Podobnostní metrika je funkce přiřazující reálné číslo určitým dvěma prvkům. Způsob přiřazení reflektuje naši ideu – co to znamená býti podobným, a to tak, že dvěma podobným prvkům přiřazuje číslo větší nežli dvěma prvkům nepodobným.

Obecně termín Locality sensitive hashing (dále jen LSH) označuje využívání hashovací funkce s vlastností: Pravděpodobnost kolize (neboli přiřazení stejného hashe) dvou prvků je vyšší pro prvky s vyšší podobností nežli pro prvky s nižší podobností (na základě podobnostní metriky)

Vyhledávání podobných prvků na základě LSH pak funguje následovně: Prvky běžně ukládáme do hash tabulky, ale při vyhledávání prvků, které jsou podobné danému prvku vezmeme kolonku

odpovídající hashi toho daného prvku a vrátíme všechny prvky z této kolonky. Takto jsem vyhledávání implementoval.

Jako LSH lze využít uživatelem zvolená hashovací metoda, přičemž vestavěné je jsou metody získávající hash ze slovního popisu obrázku získaného přes Azure Vision API. Místo podobnostní metriky používám identitu takto získaných hashů, což se může zdát cyklické, ale splňují-li vlastnost, že pro podobné obrázky je velká pravděpodobnost přiřazení stejného hashe, pak existuje podobnostní metrika představující tuto naši ideu – co to znamená býti podobným, a tyto hashovací metody se dají považovat za LSH metody

Dalšími úpravami vyhledávání na základě LSH mohou být: (Pro zvýšení šance nálezu) ukládání prvků do více hash tabulek, každá s různou LSH funkcí a poté vrácení prvků z odpovídajících kolonek ze všech těchto tabulek. (Nebo, pro zvýšení přesnosti) odfiltrování příliš odlišných takto vrácených prvků na základě podobnostní metriky. [1] Druhá úprava je v mé implementaci kvůli nedefinované podobnostní metrice neproveditelná.

5. Implementace

5.1. Konfigurační soubor – LSH moduly

config.txt:

```
{
 "database": "img_hashes.db",
 "hashMethods": [
  [
   "AzureVision",
   "src/hash_methods/AzureVision/analyze-img.txt",
   "C:/Program Files/Git/bin/bash.exe",
   "-c"
  ],
  [
   "AzureVisionNoMiddlepoint",
   "src/hash_methods/AzureVision/analyze-img2.sh",
   "C:/Program Files/Git/bin/bash.exe",
   "-c"
  ],
  [
   "hashMethodName",
   "/path/to/that/hashingProgram",
   "/path/to/interpreter",
   "--intepreterOptions"
  ]
 ],
 "def": 1,
 "middlePoint": "https://github.com/gyarab/2023-4e-liska-near_duplicate_img_detection"
}
```

Struktura json konfiguračního souboru: Jméno SQL databáze, kterou bude aplikace využívat; Uživatelem rozšiřitelný seznam LSH metod (izolovaných modulů) se všemi parametry k jejich spuštění; Defaultní číslo metody (která bude použita pro vyhledávání duplicit); Střední bod – pokud metoda vyžaduje veřejný přístup k obrázkům budou nahrány do tohoto repozitáře (nedoporučuje se). Databáze se sama přizpůsobuje konfiguračnímu souboru, tedy přidá sloupky pro nové LSH moduly.

5.1.1. Přidání vlasního LSH modulu

Jako validní modul v konfiguračním souboru je považován čtyřprvkový seznam který obsahuje v pořadí textové řetězce:

- Jméno hashovací metody (může být libovolné)
- Cesta k souboru se skriptem obstarávající hashování
- Cesta k interpreteru, který bude skript číst
- Options pro volání interpreteru pokud nutné

Standard který musí modul dodržovat:

a) Používá-li middlepoint
   1. argument = jméno souboru obrázku
   2. argument = adresář ve filesystému
   3. argument = adresa vzdáleného filesystému

b) Nepoužívá-li middlepoint
   1. argument = celá cesta k soubouru obrázku v lokálním filesystému

Všechny moduly vrací na standartní výstup hash jako textový řetězec

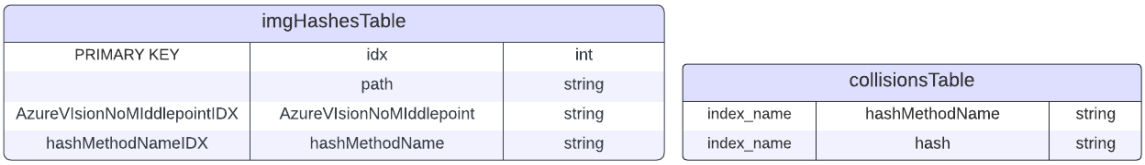
5.1.2. Vestavěné LSH moduly

Všechny vestavěné metody *AzureVision*, *AzureVisionNoMiddlepoint* a *img2soundex* pošlou Azure Vision API data souboru s obrázkem a dále zpracovávají textový popisek obrázku který API vrátí. Práce s API je postupuje dle příkladu užívání Azure Vision API [2]. *AzureVision* metoda přitom prvně nahraje obrázek na veřejně dostupné uložiště a teprv odtud nechá obrázek přenést. *AzureVision* a *AzureVisionNoMiddlepoint* z popisku spočtou md5 hash a ten vrátí. *Img2soundex* z popisku spočte Soundex hash [3], což je hashovací funkce, která text s foneticky podobným začátkem mapuje na stejný hash.

5.1.3. Rozložení v čase a běh LSH modulů

Vlákno *GatedQuerry* se spustí při potvzení přidání obrázků. Obrázky přidá do fronty a rozloženě v čase (kvůli požadavkům API) vyvolává LSH metodu na obrázky z fronty. LSH metoda se spouští pomocí *ProcessBuilder* jako samostatný proces s obrázkem jako parametr. Výstup [4], tedy hash, je pak předán *třídě CollisionHaldler* společně s cestou k obrázku. Ta se postará o zjištění a uložení kolize.

5.2. Databáze a třídy obstarávající práci s ní

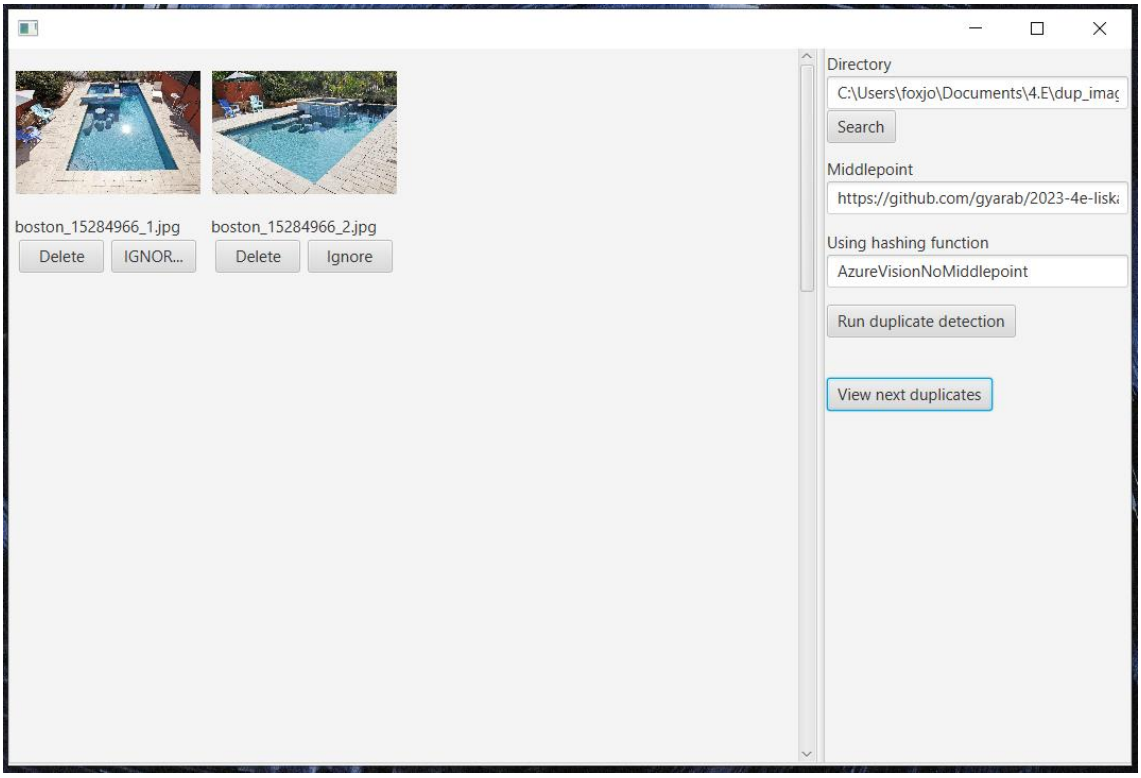


Obrázek č. 1 – Struktura tabulek v databázi

Při Inicializaci aplikace se vytvoří spojení s SQLite databází pomocí JDBC a toto spojení je využíváno v dalších částech. Třídy *ImgHashesTableConnection* a *CollisionHandler* spravují funkce jako přidávání, různé způsoby vyhledávání a mazání řádků v pořadí v tabulkách *imgHashesTable* a *collisionsTable. CollisionHandler* pracuje i na *imgHashesTable,* ale pouze prostřednictvím *ImgHashesTableConnection*.

5.3. Propojení s grafickým rozhraním

Grafické rozhraní je obstaráváno pomocí JavaFX a komponenty jsou definovány v .fxml souborech. Komponenty hlavního okna a položky obrázku mají vlastní FXMLController třídy. Uživateli je umožněno vybrat adresář s obrázky spuštěním nativního rozhraní pro výběr adresáře. Kolonky jako střední bod a jméno hashovací metody jsou automaticky doplněny z konfiguračního souboru. Spuštěním detekce duplicit se jak již bylo zmíněno vytvoří vlákno *GatedQuerry* s parametry vyplněnými v kolonkách. Zobrazení další skupiny duplicit pak od *CollisionHandler* zjistí další skupinu kolizí, vytvoří položky obrázků a vloží je na viditelnou plochu.



Obrázek č. 2 – Grafické rozhraní

6. Závěr

Výsledkem práce je funkční aplikace se snadným použitím. Možné další úpravy zahrnují podporu získání všech LSH hashů najednou pro jeden obrázek, užitečné pouze pokud uživatel používá jiné než vestavěné LSH metody, nebo přidání většího náhledu obrázku pro lepší porovnávání velmi podobných fotek.

7. Spuštění aplikace

7.1. Požadavky

- Bash nebo Git bash
- Java Runtime Environment - 1.8
- Klíč k Azure services, kontaktujte autora

7.2. Instalace

Stáhněte jq a přidejte do PATH
> curl -L -o /usr/bin/jq.exe https://github.com/stedolan/jq/releases/latest/download/jq-win64.exe

Vyklonujte repozitář a do ./src/hash_methods/AzureVision/ přidejte *resource_key.txt* s klíčem k Azure services.

7.3. Spuštění

Máte-li maven, lze spustit StartApp.java alternativně lze spustit ndid.jar pomocí:
> java --module-path <path/to/java-sdk/lib> --add-modules javafx.controls,javafx.fxml -jar target/ndid.jar

8. Zdroje

- [1] Více o LSH:
  https://www.fi.muni.cz/~xkohout7/Research/clanky_cizi/lsh/indyk.pdf
- [2] Quickstart for Azure image analysis:
  https://github.com/MicrosoftLearning/AI-900-AIFundamentals
- [3] Soundex implementace:
  https://github.com/oreillymedia/java_cookbook_3e/blob/master/javacooksrc/javacooksrc/main/java/strings/Soundex.java
- [4] Čtení výstupu procesu:
  https://stackoverflow.com/questions/16714127/how-to-redirect-processbuilders-output-to-a-string