2015/16

ČVUT-FIT MI-VMV

POROVNÁVÁNÍ SVG NA ZÁKLADĚ CEST A BAREV

VLADIMÍR MLÁZOVSKÝ A GEORGE DAUSHEYEV

Cíl

Cílem našeho projektu je vytvořit aplikaci, která dokáže mezi sebou porovnávat podobné SVG dokumenty.

Řešení

Jako aplikační rozhraní jsme si zvolili webovou aplikaci. Aplikace umožňuje uživateli nahrát SVG dokument, prozkoumat jeho vlastnosti a podívat se, které dokumenty vybrala aplikace jako podobné.

Porovnávací algoritmus

Jádrem celé aplikace je porovnávací algoritmus. Ten funguje v několika vrstvách. V první je parsován SVG dokument, v druhém jsou sestaveny histogramy úhlů a barev, ve třetí je vypočítána podobnost každý s každým a v posledním kroku jsou nejpodobnější dokumenty předloženy uživateli.

Parsování SVG

Aby bylo parsování dobře pochopeno, přiblížím nejprve strukturu SVG dokument. SVG dokument je obyčejný XML dokument. V jeho struktuře jsou zaneseny jednotlivé vrstvy a objekty, které následně interpret ve správném pořadí zobrazí. Stěžejním elementem v této struktuře je element `<path>`, ten definuje *cesty*. Cesta může být libovolná křivka, lomená čára nebo spojitý obrazec. Právě pro svoji univerzálnost je velmi často používán a lze na něj jakýkoliv jiný element převést. Dalšími prvky, které by bylo možné využít, jsou obdélník, elipsa nebo text.

Parsování probíhá tak, že se využije XML struktura a vyčtou se všechny výskyty `<path>`, z elementu se vybere atribut `d`, který popisuje cestu. Z cesty vyčteme body a na bodech napočítáme úhly, jak se cesta láme.

Z textu dokumentu se vyberou RGB barvy. Barva má vždy tvar `#HHHHHH`. My ji převedeme do HSL prostoru. Tím získáme relevantnější výsledky bližší cítění barev člověkem.

Sestavení histogramů

Připravili jsme histogramy pro úhly a barvy. Úhly mají připravených 30 přihrádek pro interval $\langle 0, \pi)$. Každá ze složek HSL má vyhrazených přihrádek 8 pro interval $\langle 0,1\rangle$. Vypočítané úhly a barvy sumarizujeme do histogramů.

Vypočítání podobnosti

Pro podobnost využíváme kvadratickou vzdálenost.

$$K = \sqrt{\overline{x^2}} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}}$$

Obecně lze říci, že velké rozdíly mezi dvě porovnávanými sloupci histogramu jsou pro kvadratickou vzdálenost méně důležité. Díky tomu se zanedbávají nesmyslné hodnoty. Čím je vzdálenost větší, tím jsou si histogramy méně podobné.

Poté, co porovnány histogramy, je jejich podobnost váženým průměrem agregována do jednoho čísla od na interval $\langle 0, \infty)$. Toto číslo je směrodatné pro zobrazování podobných dokumentů.

Výběr pro uživatele

V posledním kroku se podle vybraného dokumentu zobrazí uživateli podobné. Tedy zobrazí se ty, které mají pro podobnost nejmenší číslo.

Implementace

Aplikace je implementována jako webová.

Jazyky, nástroje, knihovny

- PHP, Python, MySQL
- HTML, JavaScript, CSS
- Nette framework
- Běžné knihovny PHP a Python
- Běžné frond-endové knihovny a frameworky

V PHP je implementováno uživatelské rozhraní a spojení front-end back-end. V Pythonu je porovnání histogramů. Data si předávají přes databázi.

Toto řešení jsme zvolili poté, co jsme si uvědomili, že porovnání pro větší, množství dokumentů má lineární složitost. Přestože je velmi rychlé, je lepší pro dobrou odezvu nechávat provést porovnání v odděleném procesu. Oddělený proces je spouštěn z PHP příkazem `exec` jako z příkazové řádky. Díky této paralelizaci běhá aplikace svižně a postupně doplňuje data. Uživatel není v použití limitován.

A Python jsme zvolili, protože je nám sympatický.

Cena

Pojďme si trochu rozebrat ceny jednotlivých operací.

N … velikost kolekce

t_d … čas potřebný pro zpracování dokumentu

t_h … čas potřebný pro porovnání dvou dokumentů (několika dvojic histogramů)

Vložení do kolekce

Jak náročné je vložení jednoho dokumentu do kolekce?

$$T = t_d + Nt_h$$

Při změně porovnávacího algoritmu je potřeba celou kolekci přepočítat.

$$T_{rebuild} = \frac{N^2 t_h}{2}$$

Díky tomu, že je relace podobnosti symetrická, se nám problém zmenšuje, avšak stále má kvadratickou složitost.

Odebrání z kolekce

O odebrání se stará databáze MySQL. Z pohledu PHP je to jeden dotaz – odeberou se informace o SVG souboru. V databázi se odeberou závislé řádky z podobnostní tabulky.

Požadavky na běh

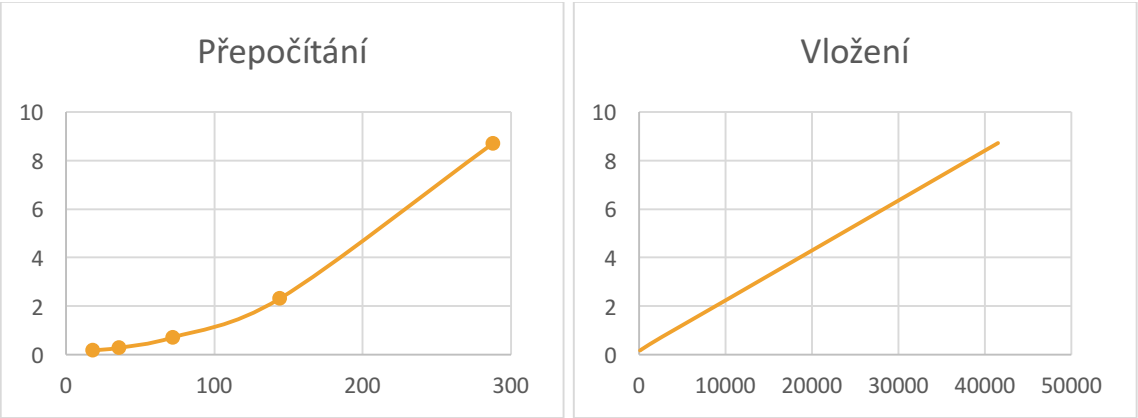
SW zázemí:

- Unix
- MySQL 5.6.21
- Apache 2.4.16
- PHP 5.6.12
- Python 2.7.10 s MySQLdb modulem
- Webový prohlížeč (Chrome)

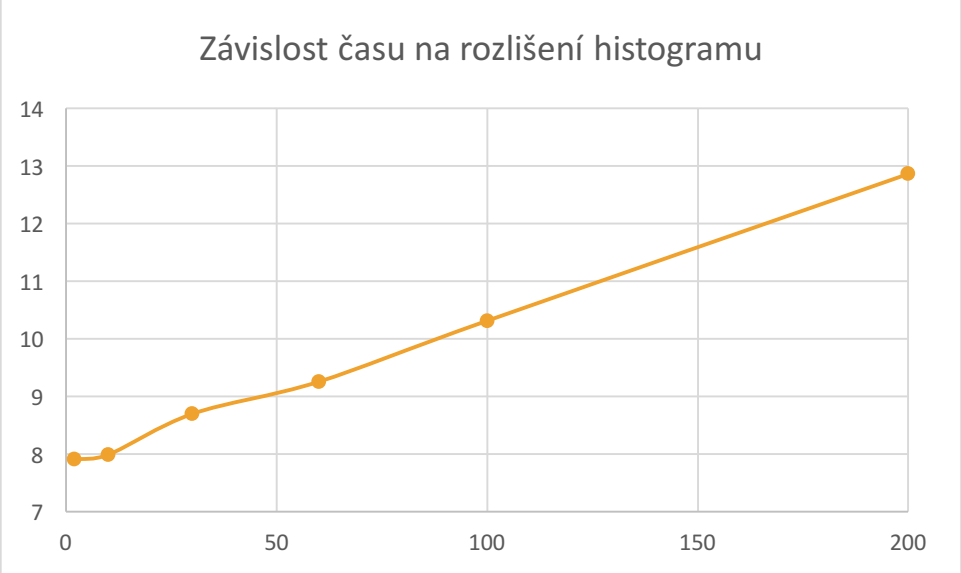
HW jsme netestovali, ale jsem si jistý, že cokoliv, na čem poběží výše uvedený software aplikaci hravě zvládne.

Experimentální sekce

Naše úloha je specifická v tom, že není téměř ovlivněná velikosti vstupu nebo výstupu. Největším problémem je velikost kolekce. Následující graf ukazuje narůstající čas pro přepočítání podobností a vložení nového prvku v závislosti na velikosti kolekce.



Zpřesnění by mělo přinášet zvýšení rozlišení histogramu. Zafixujeme tedy velikost problému na poslední na 41 472 porovnání a pozměňme rozlišení ze stávajících 30 na jiné hodnoty.



Diskuze

Zkusíme zde rozebrat dvě části z experimentální fáze, jak by se dali zlepšit.

Velikost problému a čas

Z grafů Přepočítání a Vložení je patrná lineární závislost mezi počtem porovnání a časem. To není špatné, ale ani to není nejlepší. Jak by to šlo udělat lépe?

Škálovatelnost a optimalizace

I na jednom stroji by šel výpočet výrazně zrychlit. Stačí, aby se začali vkládat obrázky rychleji za sebou, porovnávací skript se tím pouští vícekrát a v počítači se tak ideálně využívá více vláken naráz.

Dotazy do databáze mohou vkládat data hromadně. Insert mnoha dat jedním dotazem je rychlejší, než vkládat data po jednom řádku.

Databáze může být distribuovaná, takže jednotlivé dotazy nemusí na sebe navzájem čekat.

Co nepočítat

Z povahy dat by mělo dříve či později vyplynout, že některé skupiny obrázků jsou si vzájemně podobné. Z těchto skupin tedy vybrat reprezentativní vzorek (za každou skupinu jeden obrázek) a v prvním kroku udělat porovnání nového obrázku s reprezentanty. Z reprezentantů vybrat k nejlepších. Ve druhém porovnat nový obrázek jen s těmi obrázky, které jsou do určité vzdálenosti od reprezentanta. Pokud nebyl nalezen podobný reprezentant, pak je nový obrázek novým reprezentantem a musí mít vypočítanou podobnost se všemi ostatními obrázky.

Díky tomu se významně ušetří počet porovnání i paměťová náročnost uložených výsledků v databázi.

Závěr

Výstupem naší semestrální práce je prototyp na porovnávání SVG dokumentů na základě podobnosti křivek.
Výsledky jsou střídavé. Porovnání na testovací sadě 18 obrázků dopadá u složitějších kreseb lépe. Jistě by bylo zajímavé metodu více prozkoumat na větším problému.

Statistika přesnosti (subjektivní)

Nakonec jsme provedli takový malý test na našem vzorku, ve kterém jsme ke každému obrázku očekávali jeden podobný. Zde jsou výsledky:

- 57 % obrázků se trefilo na první místo dle očekávání
- 83 % obrázků se trefilo do třetího místa dle očekávání
- 11 % obrázků bylo vyhodnoceno proti očekávání.