ITS

Technická zpráva - FIT - VG20102015006 - 2011 - 05

ing. Aleš Láník, ing. Petr Nohejl,
ing. Jiří Král, Martin Kolář M.Sc.

Faculty of Information Technology, Brno University of Technology

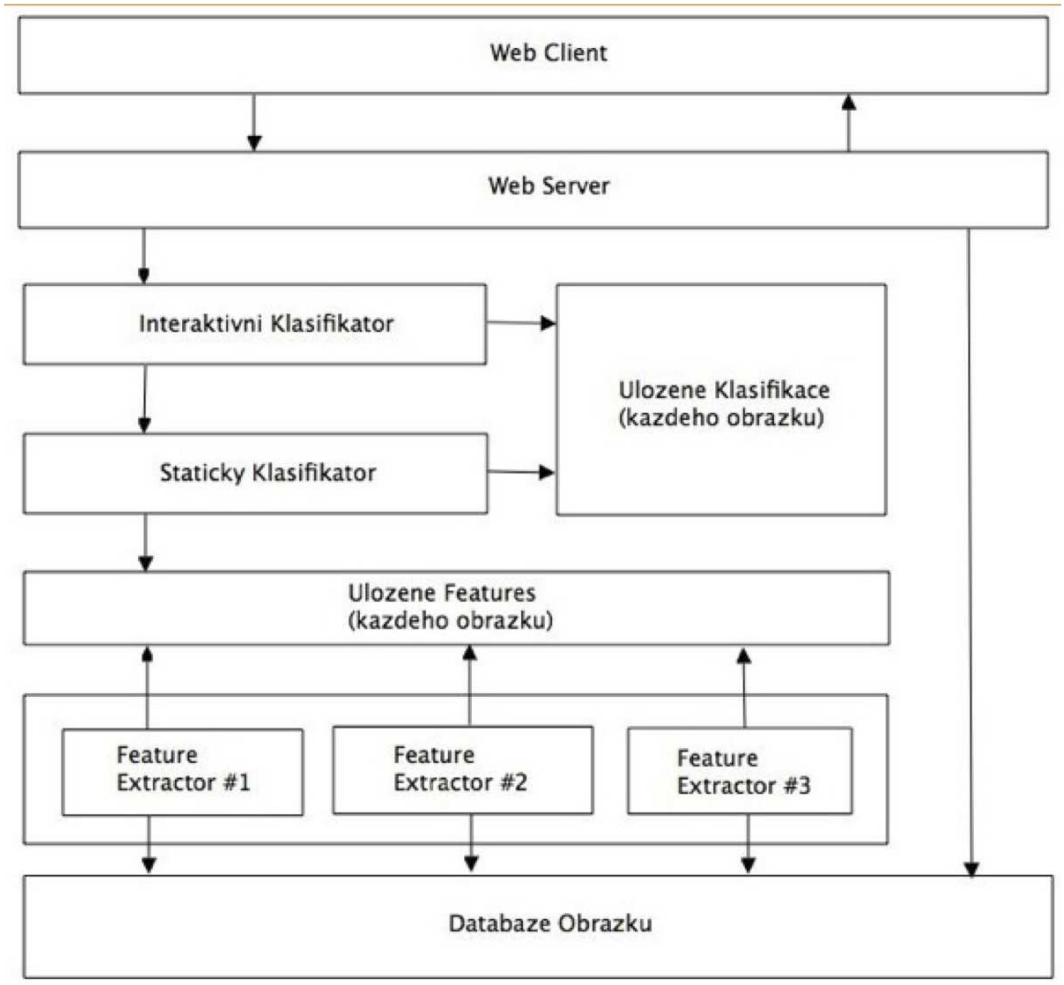
1.12.2011

Obsah

Abstrakt Cílem projektu ITS (Image Tag Suggestion) je vytvořit webovou aplikaci, která zajišťuje správu fotografií a automatické tagování na základě klasifikace obrazu. Aplikace funguje na principu sociálních sítí a umožňuje uživatelům sdílet data mezi sebou, vytvářet skupiny, vyhledání v obrazových datech atd. Hlavní oblastí zájmu v tomto projektu je však inteligentní tagování tzn. uživateli jsou automaticky generovány možné tagy v obraze a to jak na základě klasifikace z detektorů různých objektů (např. detektor obličejů, postav, aut ...) tak i na základě znalostí dodaných uživatelem (jím označených tagů) a pravděpodobnosti výskytu těchto tagů současně.

1 ITS

Projekt ITS si klade za cíl vytvořit webovou aplikaci, která zajišťuje správu fotografií a automatické tagování na základě klasifikace obrazu. Na diagramu 1.1 je znázorněna základní struktura systému.



Obrázek 1.1. Schéma funkčních bloků projektu ITS

– **Web Client**
Představuje webový prohlížeč, ve kterém se zobrazují HTML výstupy front-endové části projektu. Řada uživatelských funkcí, jako například tagování obrázků nebo nahrávání obrazových

dat na server zajišťuje klientský JavaScript, který skrze skrze JSON[1] komunikuje se serverem. Tento blok je podrobněji rozepsán v kapitole 2.

– **Web Server**
Web server zaštituje obsluhu klientských částí a spouštění klasifikačních bloků. Obrazová data jsou ukládána do datového uložiště na samostatném serveru přes Video Terror API (dále jen VTAPI). Více o tomto bloku v kapitole 3.

– **Interaktivní klasifikátor**
Interaktivní klasifikátor generuje strom možných tagů na základě pravděpobnosti tagů daných statickým klasifikátorem, tagů daných uživatelem a pravděpodobnosti výskytu jednotlivých tagů mezi sebou. V kapitole 5 je podrobněji popsána činnost tohoto bloku.

– **Statický klasifikátor**
Statický klasifikátor generuje pravděpodobnosti tagů pouze na základě příznaků z obrazových dat. Tento blok může obsahovat libovolné klasifikátory (např. AdaBoost, SVM , ...). Viz kapitola 4.

– **Feature extraktory**
Tento blok zajišťuje extrakci příznakových vektoru k obrazovým datům. Obecně se může jednat o libovolné příznaky (např. LBP[2], SURF[3], ...) podrobněji viz kapitola 4.

– **Datové uložiště**
Uložiště poskytující prostor pro ukládání a načítání obrazových ale i programových (příznakové vektory, výstupy statických klasifikátorů) dat. Podrobněji viz kapitola 6.

Datová komunikace mezi jednotlivými bloky je realizována pomocí VTAPI kromě předávání dat webového serveru s klientem, kde již z principu je využit protokol HTTP.

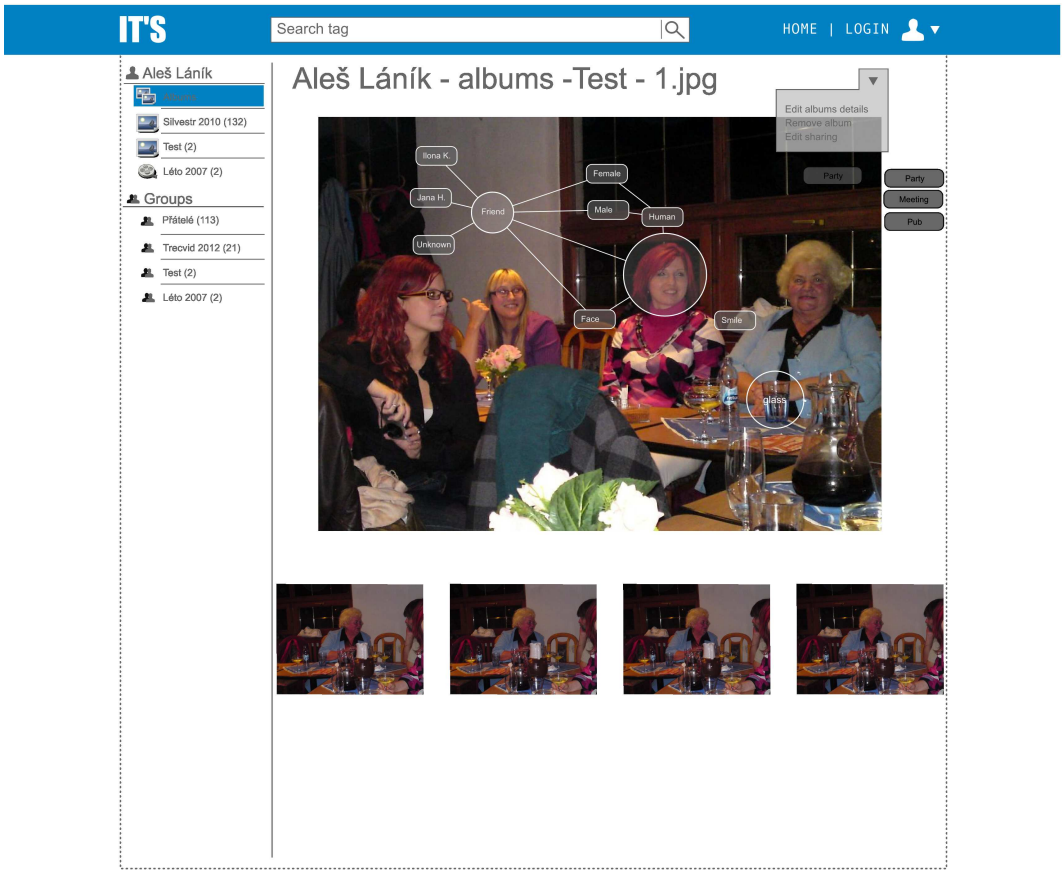
[1] http://www.json.org/
[2] Local binary patterns[3]
[3] Speeded-Up Robust Features[1]

2 Webový klient

Klientská část celého systému je generována jako dynamická webová stránka skrze webový framework Django (viz kapitola 3). Pro potřeby uživatelského rozhraní (dále jen UI) inteligentního tagování se využívají scripty běžící na klientské části napsané v jazyce Java script. Tyto scripty komunikují s webovým serverem skrze formát JSON. Jedná se především o dotazy na tagy obsáhlé v aktuálním obraze, přidání nového tagu a mimo jiné dotaz na navrhované tagy.

UI klientské části bylo navrhnuto s ohledem na jednoduchost použití a snaží se kopírovat trendy, které lze vysledovat na portálech obdobných služeb.

Na obrázku 2.1 je znázorněn návrh UI, respektive nejdůležitější části a to interaktivního tagovacího nástroje.



Obrázek 2.1. Návrh UI projektu ITS

V tomto návrhu je zhruba nastíněna možnost jakým způsobem by mělo být prováděno napovídání tagů, které se v obraze řadí do hierarchické stromové struktury. Do budoucna se počítá s implementací rozličných verzí této komponenty za účelem zlepšení použitelnosti rozhraní (např. 3D zobrazení zkrze WebGL[4]).

[4] Web-based Graphics Library

3 Webový server

Webové rozhraní ITS projektu je implementováno v jazyce Python s využitím frameworku Django.

3.1 Django framework

Django je otevřený komplexní webový framework napsaný v jazyce Python. Umožňuje rychlý vývoj webových aplikací a efektivní správu dat pomocí vlastního administračního rozhraní. Celý framework je založen na konceptu "DRY"(Don't repeat yourself) a znovupoužitelnosti jednotlivých komponent. Používá relační databázový model a podporuje několik databázových systémů včetně PostgreSQL, který projekt ITS používá. Django je založeno na softwarové architektuře MVC (Model-view-controller).

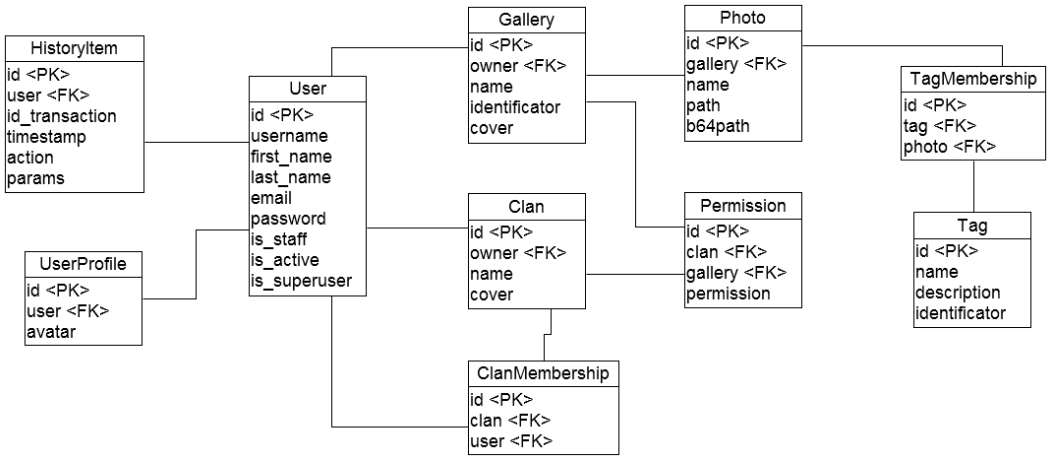
Jádro frameworku tvoří ORM (objektově-relační mapper), který propojuje datový model, reprezentovaný třídami, s relační databází. Vestavěný šablonovací systém umožňuje snadno renderovat modelová data do HTML stránek. Šablony rovněž podporují dědičnost. Django dále nabízí vlastní webserver určený pro vývoj a testování aplikací. Pro ostrý provoz se obvykle používá webserver Apache 2 s modulem Mod Python nebo WSGI.

3.2 Návrh webové aplikace

Webová aplikace umožňuje nahrávat, prohlížet, sdílet a spravovat fotografie. Fotografie lze sdružovat do galerií. Systém umožňuje registraci nového uživatele a přihlášení. Uživatel má možnost vytvářet a spravovat galerie, vytvářet a spravovat skupiny uživatelů, sdílet galerie mezi skupinami. Jednotlivým fotografiím je možné přiřazovat tagy. Uživatel může určit pozitivní i negativní klasifikaci tagu. Systém dále umožňuje ukládat historii všech operací a zobrazovat notifikace na význačné události (např. uživatel ve skupině, jejímž jsem členem, vytvořil novou galerii).

3.3 Relační databázový model

Databázový model webové aplikace znázorňuje následující diagram 3.1.



Obrázek 3.1. Databázový model django aplikace

Uvedený ER diagram představuje ORM model, definovaný jednotlivými třídami v jazyce Python. Třída User je součástí vestavěného modulu `django.contrib.auth` frameworku Django. Modul poskytuje metody pro ošetření autentifikace uživatele. Použití modulu je zároveň nezbytné pro spuštění administračního rozhraní Djanga. Příznak `is_staff` značí, zda má uživatel přístup do administračního rozhraní. Příznak `is_active` značí, zda se může uživatel přihlásit. Hodnota false obvykle znamená, že uživatel smazal svůj účet, nebo byl zablokován administrátorem. Položka `is_superuser` určuje, zda má uživatel plná práva upravovat data v administrátorském rozhraní.

Třída `UserProfile` rozšiřuje vestavěný model `User` a uchovává navíc cestu k avataru. Třída `HistoryItem` uchovává informace o provedené operaci v rámci jedné transakce (např. přidání fotografií do galerie, smazání, sdílení atd.). Položka `timestamp` ukládá časové razítko operace a proměnná action určuje typ operace.

Model `Gallery` uchovává všechny dostupné galerie. Důležitá je zde položka identificator, která představuje jednoznačný primární klíč do galerie v rozhraní VTAPI. Proměnná `cover` značí cestu k náhledovému obrázku galerie. Model `Photo` má svoje pojmenování a ukládá cestu k obrázku na serveru. Třída `Tag` uchovává jméno tagu, popis a podobně jako `Gallery` jednoznačný identificator tagu ve VTAPI. Model `TagMembership` určuje, která fotografie má přiřazený určitý tag.

Třída `Clan` představuje skupinu uživatelů. Každá skupina má svého vlastníka, název a vlastní avatar. Model `ClanMembership` definuje příslušnost jednotlivých uživatelů do skupin. Třída `Permission` ukládá informace o přístupových právech ke galerii. Definují se základní práva pro čtení, zápis a další rozšiřující práva.

3.4 Struktura Django projektu

Projekt používá několik externích modulů. Modul `django_schemata` zajišťuje práci s PostgreSQL schématem. ITS projekt používá dvě schémata. První schéma `Public` využívá přímo rozhraní VTAPI. Druhé schéma `Django` používá samotná webová aplikace.

Modul `sorl.thumbnail` umožňuje vytvářet náhledy obrázků a využívá se u galerií.

Modul `its.multiuploader` zajišťuje hromadné ukládání obrazových dat na server. Hlavní modul celé webové části je potom `its.itsapp`, který představuje front-endovou část aplikace.

V kořenovém adresáři projektu se nachází nástroje pro synchronizaci databázového modelu s objektovým modelem, spuštění vývojového webserveru, nastavení práv souborů a další potřebné skripty. Dále se zde nachází konfigurační soubor, nastavující připojení k databázi, schémata databáze, použité moduly, cestu ke statickým souborům, cestu k médiím, cesty k šablonám a různá další nastavení.

Hlavní modul `its.itsapp` tvoří 3 základní části. První je objektový datový model, jenž je pomocí ORM mapován na relační databázi. Další část tvoří URL dispatcher, který pomocí regulárních výrazů mapuje URL cesty na funkce, zpracovávající daný HTTP požadavek. Důležitou součástí modulu jsou views. Jedná se o metody, které volá URL dispatcher při zaslání HTTP požadavku. Na jejich vstupu jsou zadány parametry požadavku a výstupem je HTML stránka. View zpracuje zadané požadavky, provede požadovanou funkčnost a předá výsledná data šablonovacímu systému. Šablonovací systém data zpracuje a vygeneruje příslušnou HTML stránku.

4 Trénování SVM

4.1 Extrakce příznaků

Extrakce příznaků každého obrázku se skládá ze dvou částí, první je extrakce nízkoúrovňových příznaků pomocí programu `compute_descriptors_32bit.ln`[5]. Nízkoúrovňové příznaky jsou získávány například extraktory SIFT, CSIFT a dalšími. Tyto extraktory jsou dále parametrizovány, pro příklad uvedu parametr pro určení velikosti oblasti zájmu: `-dense 8 8`.

Ve druhé části se převádí příznaky z nízkoúrovňových extraktorů do Bag-of-visual-words reprezentace. Vstupem pro převod jsou příznaky, a slovník. Převod tedy spočívá v tom, že se počítá počet výskytů jednotlivých visuálních slov neboli kombinací nízkoúrovňových příznaků, pro daný obrázek. Každý nízkoúrovňový příznak může být navíc přiřazen do jednoho nebo více slov, podle toho jakou požadujeme přesnost. Velikost slovníku pak určuje počet vysokoúrovňových příznaků.

Výstupy extrakce příznaků jsou předány handleru, který je uloží do databáze pod názvem, který odpovídá dané parametrizaci. Například pro dense sampling s poloměrem 16 pixelů, CSIFT deskriptor, převod pomocí slovníku KM_L12 a nejisté přiřazování dostaneme název deskriptoru `DESC_DENSE16_CSIFT_NoA_UNC_K32_o10_L01_KM_L12`.

Pro extrakci příznaků byl vytvořeny skripty pro lokální výpočet a také pro výpočty na SGE. Oba tyto skripty byly vytvořeny tak aby minimalizovaly zápisy na síťové disky.

4.2 Stochastic Gradient Descent pro SVM

V současné době je dostupných mnoho implementací trénovacích algoritmů a nástrojů pro Support Vector Machine, s různými možnostmi použití. V tomto projektu je využit Stochastic Gragient Descent (SGD-SVM), protože je schopný, podobně jako jiné implementace lineárních SVM, pracovat s rozsáhlou datovou sadou. Výhodou SGD-SVM je však rychlost, se kterou zpracovává vstupy, je totiž schopný online zpracování dat.

SGD-SVM vychází z gradientních metod, které však díky stochastičnosti procesu mají mnohem menší šanci uváznout v lokálních extrémech. Historicky se metody založené na Stochastickém Gradient Descentu (SGD) pojily hlavně s neuronovými sítěmi, ale Léon Bottou vytvořil implementaci SGD pro většinu trénovacích algoritmů a právě i pro SVM. Implementace v tomto projektu je založená právě na jeho implementaci SGD-SVM, ale je implementována v Matlabu, a je rozšířena o Cross-validaci a s ní spojené obslužné postupy. Matlab byl vybrán pro implementaci kvůli rychlosti práce s maticemi.

Algoritmus je popsán v následujícím pseudokódu 0.1:

```
% stratifies input data into __cv_count__ bins according to y
[train test] = stratified_sampling(x,y,cv_count) % train and test are fields

for all bins
for all lambda
    init(params)
    for i = 1:epochs
        new_params = train_sgd_svm(cur_train,params)
        params = new_params
    end
    results = test_sgd_svm(cur_test)
    store(results,params)
end
end

best_bin_params =best_bin(results,params)
store(avg(best_bin_params))
```

[5] http://kahlan.eps.surrey.ac.uk/featurespace/web/desc/compute_descriptors_32bit.ln

Listing 0.1. Algoritmus pro trénování SGD-SVM

Jak je vidět z pseudokódu algoritmus provádí trénování na datech v takzvaných epochách, kterých je volitelný počet. Dále také je dobré si všimnout průměrování parametrů nejlepšího cross-validačního shluku. Úkolem metody vykonávající tento pseudokód je zjistit, jaké nastavení parametru λ je nejlepší, a spolu s tímto parametrem uložit odpovídající nastavení klasifikátoru pro danou třídu.

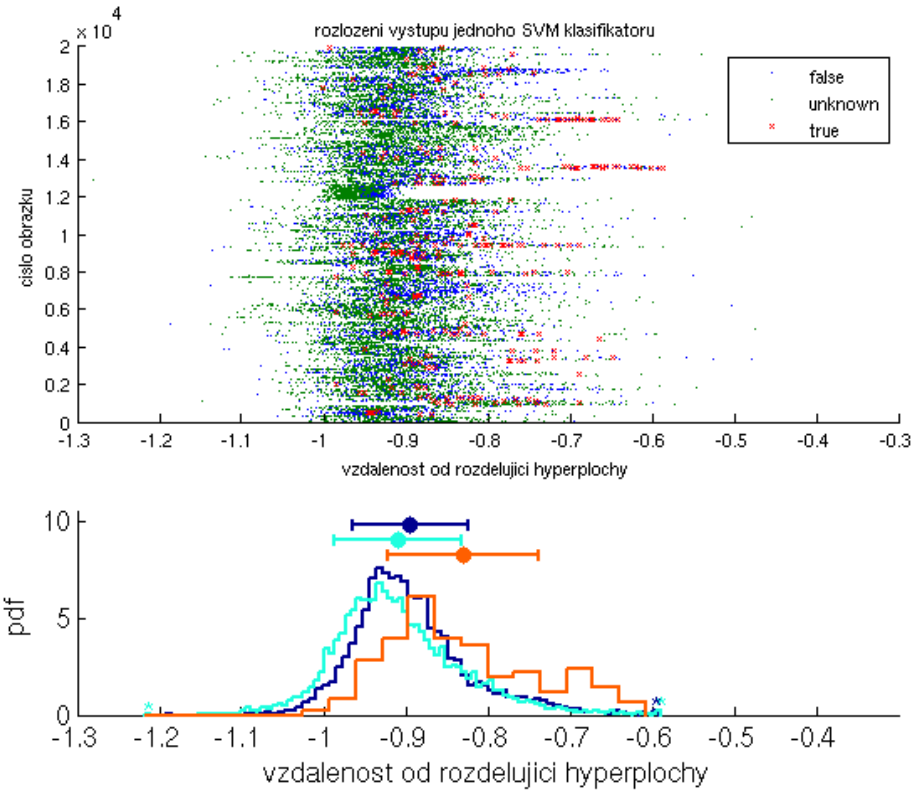
Výsledky trénování (parametry) ale i odezvy jednotlivých SVM prodané obrázky jsou uloženy do DB pomocí handleru, který je uloží do databáze pod zadaným názvem.

4.3 Handler pro obsluhu VideoTerror databáze

Tento program shrnuje funkcionalitu potřebnou pro práci s VideoTerror databází v rámci projektu ITS. Za pomocí VTAPI umožňuje spravovat a ukládat informace k obrázkům (intervalům), extraktorům deskriptorů, SVM klasifikátorům (procesům) a tagům. Pro každý obrázek budeme ukládat deskriptory (vektory typu float), odezvy SVM pro každý tag a každou parametrizaci (vektory typu float) a tagy (ohodnocení 1,0,-1 značící přítomnost,neznalost,nepřítomnost daného tagu). Pro deskriptory budeme ukládat pouze názvy parametrizací a parametry pro spuštění extraktoru. V procsu SVM budeme ukládat parametry SVM (vektor float a další float proměnné).

5 Klasifikace SVM skóre s použitím grafu spojitostí

Pro každý obrázek jsou známé výstupy SVM klasifikátoru pro každý tag (třídu), a také již známé uživatelské ohodnocení přítomnosti nebo nepřítomnosti některých tagů {`true, false`}. Tyto informace se zkombinují v grafu, který je konstruován na základě předchozích spojitostí. Na něm se následně použije inferenční algoritmus Gibbs Sampling. Výstupy SVM jsou však nedostatečně informativní (viz obrázek 5.1), a proto napomáhají informace o spojitosti mezi objekty.



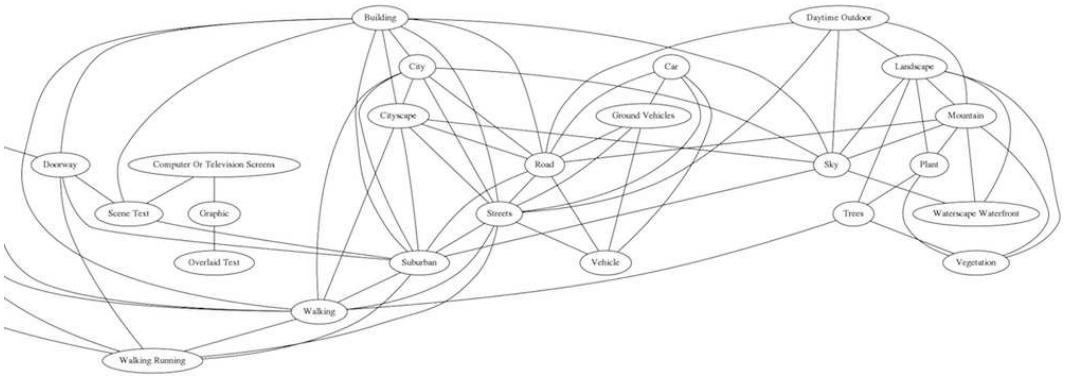
Obrázek 5.1. Visualizace SVM výstupů

Nejdříve se provede multi-dimensionální logistická regrese výstupů SVM. Tato regrese se kalibruje na všech SVM pro daný tag. Proces kalibrace je nezávislý na zbytku algoritmu a užívá pouze informace o stavu známých tagu. Výstupem kalibrace je ohodnocení {`true, false`} pro každý tag a pro každý obrázek.

Závislosti mezi všemi páry tagů (x, y) se seřadí podle $abs(p(x, y) - p(x)p(y))$, protože $p(x, y) = p(x)p(y)$ odpovídá nezávislosti. Vybere se jenom n nejvíce závislých, a z nich se vytvoří graf. (viz obrazek 5.2)

Pak se stejným způsobem do grafu přidají spojitosti mezi výstupem logistické regrese SVM pro každý tag. Testování bylo zatím prováděno nezávisle na této části.

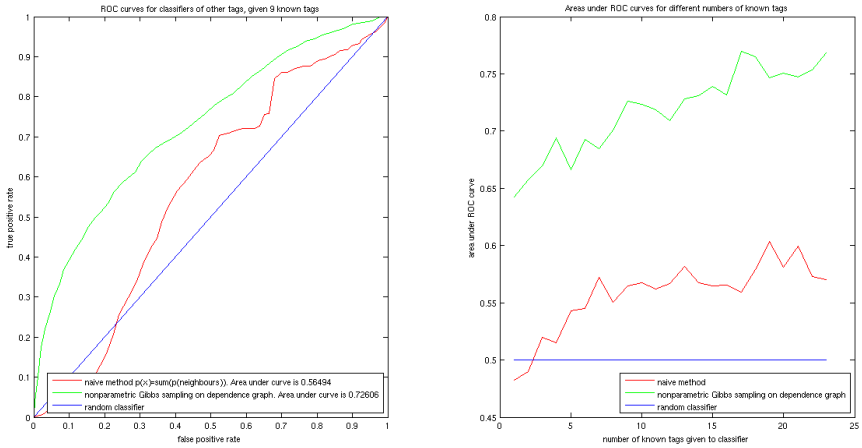
Pro daný obrázek jsou známé jisté tagy (ano a ne), a ty jsou zafixované v grafu. Na zbytku grafu se použije Gibbs sampling[2], tedy náhodná aktualizace náhodných uzlů podle aktuálního



Obrázek 5.2. Graf spojitostí tagů

stavu okolních uzlů a distribucí mezi nimi. Místo modelování distribuce každé hrany se používá neparametrické spojení skutečných dat, což řeší problém s propagací velmi nízké pravděpodobnosti každého tagu.

Tato metoda je optimalizována použitím mex[6] C kódu v matlabu. Samotný Matlab na inferenci jednoho obrázku potřebuje cca. 20 vteřin, a optimalizovaná metoda s předzpracováním umožní na stolním PC 500 inferencí za vteřinu.



Obrázek 5.3. ROC křivky

Na obrázku 5.3 jsou vyobrazeny výsledky metody pro subset obrázků, který má více než n známých tagů. N je použito na inferenci a ostatní na měření kvality předpovědi. ROC křivka vlevo zobrazuje true positive rate vs false positive rate pro případ $n = 9$. V pravo pak křivka zobrazuje povrch pod každou ROC křivkou pro hodnoty $n \in (1, 38)$.

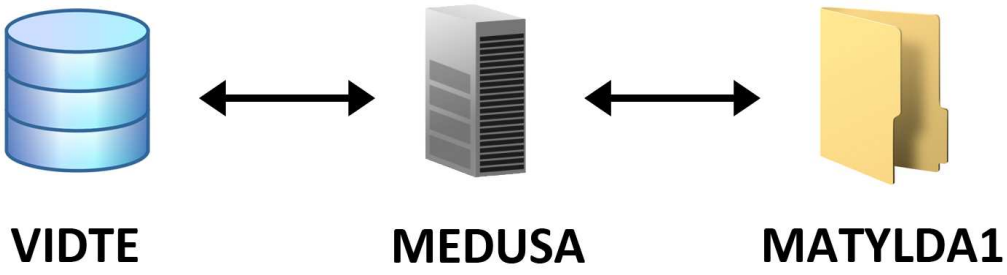
[6] MATLAB Executable - mechanizmus poskytující rozhraní mezi Matlabem a subrutinami napsanými v nativním kódu

6 Datové uložiště

Tagy, fotografie a veškeré informace o nich jsou uloženy ve VTAPI databázi, která používá databázový systém PostgreSQL. Server komunikuje s VTAPI databází pomocí speciálního middleware. Middleware představuje program napsaný v jazyce C a používající VTAPI rozhraní.

Pokud chce server komunikovat s VTAPI, spustí příslušný middleware jako konzolový program a ten mu vrátí požadovaná data ve formátu JSON. Tato data pak server dále zpracuje. Middleware tedy slouží jako spojovací můstek mezi serverem napsaným v jazyce Python a VTAPI napsaném v jazyce C. Do budoucna se počítá s odstraněním tohoto mezi bloku pomocí portu VTAPI do jazyka Python.

Databáze VTAPI běží na serveru `vidte.fit.vutbr.cz`. Samotná obrazová data jsou uchovávána na serveru `matylda1.fit.vutbr.cz` a všechny cesty k těmto souborům jsou uchovávány ve VTAPI databázi.



Obrázek 6.1. Datová komunikace mezi uložišti

Samotný webserver potom běží prozatím na serveru `medusa.fit.vutbr.cz`. Databáze VTAPI obsahuje dvě schémata. První schéma Public uchovává data galerií, fotografií, tagů a dalších dat, souvisejících se zpracováním obrazu a logicky se vztahuje k samotnému VTAPI rozhraní.

Druhé schéma Django uchovává informace o uživatelích, skupinách, oprávněních, sdílení apod. S tímto schématem přímo pracuje webserver. Uživatelská data jako např. avatary, obrázky skupin apod. se rovněž ukládají na server `matylda1.fit.vutbr.cz` a nejsou tedy uloženy lokálně na serveru medusa. Tato data se ukládají do speciálně určené galerie.

Příznakové vektory, které vzniknou zpracováním obrazu extraktory příznaků jsou ukládány přímo do VTAPI databáze odkud jsou čteny jednak statickými klasifikátory a také interaktivním klasifikátorem. Výsledky klasifikace z klasifikátoru jsou rovněž ukládány do VTAPI databáze.

7 Současný stav řešeného problému

V současné době (tedy prosinec 2011) jsou z navrhnovaného systému hotovy následující bloky.

- **Web Client**
  Základní verze klientské části je hotová, kromě části tagovacího nástroje, na které se v současné době dělá.
- **Web Server**
  Na serveru `medusa.fit.vutbr.cz` byl nainstalován framework Django a byl naimplementován webový server, který implementuje základní činost serveru.
- **Interaktivní klasifikátor**
  Byl zpracována základní verze interaktivního klasifikátoru, která nabízí vhodné tagy. Prozatím není tento blok provázán se zbytkem systému.
- **Statický klasifikátor a feature extraktory**
  Je zpracován základní mechanizmus multitřídní klasifikace. Jedná se o 500 tříd převzatých z tasku SIN projektu Trecvid (viz `http://www-nlpir.nist.gov/projects/tv2011/tv2011.html#sin`). Prozatím není tento blok provázán se zbytkem systému.
- **Datové uložiště**
  Na serveru `vidte.fit.vutbr.cz` byla nainstalována PostgreSQL databáze, ke které se přistupuje skrze VTAPI. Na serveru `matylda1.fit.vutbr.cz` vzniklo uložiště pro ukládání obrazových dat (tedy dat, na které referují odkazy z VTAPI databáze).

Reference

1. H Bay, a Ess, T Tuytelaars, and L Vangool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, června 2008.
2. S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.
3. Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.