NI-VMM

Ondřej Wrzecionko

ZS 2022/2023

Obsah

1 Úvod

1.1 Obsah předmětu

Témata:

- techniky patřící do information retrieval
- techniky související s počítačovým viděním
- indexace pro rychlejší vyhledávání a namodelování struktury databáze a způsobu dotazování

V přednáškách se budeme zabývat:

- platformy pro vyhledávání multimédií (2)
- slovníkové metody hledání obrázků (1)
- podobnostní vyhledávání v multimédiích (3)
- techniky extrakce vlastností (5)
- indexace multimédií (2)

1.2 Obsah multimédií

Biliony fotografií denně, 500 hodin videa se nahraje každou minutu na YouTube. To vše díky rychlému internetu, výpočetní síle, cloudu, principu webu 2.0 (každý je producent) a přesunu lidských aktivit na internet (sociální sítě, banky, e-shopy). Narůstá důležitost mobilních platforem, aplikace jsou používané mnohem častěji než desktop.

Multimédia jsou víc než jeden typ digitálního média (audio, obrázky, video), přesná definice ale neexistuje, je to buzzword. Dá se to všechno ale taky kombinovat – dokument s obsahem ke čtení, poslechu, vidění, nebo teoreticky i celý web. Většina typů jsou nestrukturované, jedná se ale i o člověkem zpracovaná data.

1.3 Strukturovaná vs nestrukturovaná data

Strukturovaná data šla postupně od relačních (SQL) přes objektové, XML, RDF a SPARQL k NoSQL.

Nestrukturovaná data jdou postupně od textu, řetězců, časových řad (série událostí, např. co člověk v čase sleduje), dat ze senzorů, až ke zpracování proudových/cloudových dat.

1.4 Kategorie producentů dat

Nejvíce dat produkují **free-time users** (lidé sdílí fotky z dovolené, z domu, venkovní) – hodně velká sémantika (lidé, město, příroda, sporty) s komplexními scénami. Tato data nejsou kritická, nemají anotace, jsou pro zábavu.

Pak jsou zde ale data pro **průmysl a výzkum**, tento obsah je vytvořen profesionály v mnoha velmi úzkých poddoménách (rentgeny, viditelné/neviditelné spektrum, ultrazvuk). Tato data jsou kritická, mají přesné anotace, jsou pro lékařskou diagnózu.

1.5 Deskriptory multimédií

- **anotace** – externí popis s přesnou sémantikou (explicitní: klíčová slova, odkaz, GPS a kontextové: lajky, komentáře, sdílení)
- **deskriptory vlastností na základě obsahu** (extrahované z pixelů, na nízké úrovni bez sémantiky, např. histogram barev v obrázku)
- **multimodální deskriptory** – více pohledů *(modalit)* na jednu věc (např. histogram barev a počet hran)

Například taková fotografie na Flickru obsahuje vizuální obsah (pixely obrázku), explicitní anotaci (popisek a tagy) a kontextovou anotaci (komentáře ostatních lidí v síti).

Anotace jsou dobré díky vysoké sémantice a jednoduchému vyhledávání, nevýhodou je ale subjektivnost a neúplnost. Generují je dnes také deep learning metody `https://cloud.google.com/vision/`

Obsahové vlastnosti jsou vždy přítomné a objektivní, mají ale nízkou sémantiku a hrozí tzv. *semantic gap*.

Anotovat lze v úzké doméně (kritické věci, scan mozku), ale vždy tam musí být člověk, pak jde anotovat i v široké doméně (úroveň objektů – stůl, počítač, židle, úroveň konceptů – kancelář, světlé, business).

1.6 SW architektury pro vyhledávání

Aplikace zaměřené na **data** (vyhledávací služby, hosting servery – vidím fotku hodinek → jsou to hodinky) nebo na **služby** (sociální sítě, interaktivní platformy – vidím fotku hodinek → zjistím, kde se dají koupit, co to je za značka, kolik stojí).

1.7 Vyhledávací zdroje

Hledáme textové dotazy, crawler stáhne všechen dostupný obsah webu, je nutné ale mít model pro samotné vyhledávání – nejjednodušší je **metadata-based**, ale u multimédií budeme potřebovat spíše **content-based**.

1.8 Hosting servery

Jedná se o **úložiště** obsahu a zároveň **search engine** *(vyhledávací zdroj)*, hledání ale **pouze** pomocí **textu** – překlápí se směrem k sociálním sítím. Existují **image** hosting servery: alba, galerie – Flickr, PhotoBucket, ImageShack, a také **video** hosting servery: YouTube, Dailymotion.

Jsou zde ale také **stock** servery, které obsahují **placený** obsah pro profesionální designéry, s **kontrolovanou** kvalitou obsahu – Corbis, Getty, iStockPhoto.

Největší služby umožňují také automatizované **API**, vhodné pro agregaci s ostatními daty (Google Maps, YouTube, Flickr).

1.9 Service-centric aplikace

Další krok vyhledávání multimédií, multimédia nejsou cíl, ale význam, finálním produktem je **služba**. Multimédium může být **přímý** konektor *(fotka produktu k nákupu)*, nebo **nepřímý** *(dva lidé jsou na jedné fotce = můžou mít nějaký vztah)*.

U těchto aplikací se často vyskytují také **map servery** – mapa je specifický velký obrázek, kde ale probíhá spíše **navigace**, než vyhledávání, hledat se dá textově (adresy). Pokročilejší aplikace pak můžou umožňovat zobrazení **specifických vrstev** – počasí, turistické informace.

Jednou ze skvělých věcí jsou dnes **mobilní platformy**, které umožňují snadné vyhledávání na základě multimédií, různé aplikace (Google Lens) a pracuje se i na **rozšířené realitě** (AR).

2 Způsoby a kvalita vyhledávání

2.1 Způsoby vyhledávání

Nezávisle na způsobu vyhledávání zjišťujeme **relevanci** výsledku. Buď je relevance **binární** (objekt je/není relevantní), nebo **více-hodnotová** (objekt je relevantní/podobný více či méně).

	dotazování	brouzdání	exploration	doporučení
anotace	**klíčová slova**	prohledávání	navigace	filtry
obsah	**příklad**	grafu	v hierarchii	

Pro hledání **anotacemi** je nejčastěji používáno hledání **klíčovými slovy** pomocí klasických modelů, které známe. Při navigaci v hierarchii jde o **ontologickou** hierarchii *(nábytek je nadřazený stolu)*.

Při hledání na základě **obsahu** je časté hledání příkladem a navigace v hierarchii konceptů. Filtry jsou založené na obsahu, ne na textu.

2.1.1 Dotaz

Předpokládáme, že víme, na co se chceme zeptat. **Dotaz** je explicitní popis toho, co hledám. Model může být založen na **textu** (anotace), nebo **obsahu** (nahraný obrázek, může být i kombinace).

Výsledkem dotazu je pak buď **neseřazená** množina objektů *(binární relevance)* nebo **seřazená** množina objektů *(více-hodnotová relevance)*. Na tento výsledek může uživatel poskytnout **zpětnou vazbu** a požádat o znovu-seřazení.

2.1.2 Browsing

Předpokládáme, že nedokážeme přesně popsat, co chceme najít. **Brouzdání** je iterativní navigací v databázi. Výsledkem je **podmnožina** objektů v databázi, případně nějaká hierarchie.

Hledat můžeme **explicitním** grafem (příspěvky konkrétního člověka na sociální síti) nebo **virtuálním** grafem (vytvořen specificky pro browsing, tvořen posloupností dotazů – hledám dárek na vánoce a náhodně klikám na podobné produkty).

2.1.3 Exploration

Chceme získat **představu** o obsahu **celé** databáze – agregované brouzdání, vizualizace databáze, důraz na real-time zpracování.

Příkladem může být prozkoumání prostoru autobusů – zoom in/out, čím jsou předměty **podobnější**, tím jsou **blíže** k sobě.

2.1.4 Recommendation

Rekomendace neboli filtrování, postaveno naproti dotazování, **doporučovací systémy** (explicitně co chci vs implicitně podle nastavení uživatele, historie hledání, kolaborativní filtrování…)

Kdysi toto souviselo s **relevance feedback**: 3 kroky → prvotní vyhledání objektů → možnost zpětné vazby *(expertní nebo neexpertní // explicitní, implicitní)* → vylepšený výsledek, probíhá ve **více iteracích**.

Dnes je relevance feedback častější implicitně – na základě toho, **kam** uživatel kliká jde zjistit, který produkt byl pro něj zajímavý a relevantní.

2.2 Kvalita vyhledávání

2.2.1 Semantic gap

Při zadávání hledaných výsledků dotazů jsou zadání často **vysokoúrovňové**, ale reprezentace dat je velmi nízkoúrovňová. Jedná se tedy o velký rozdíl mezi sémantikami = **semantic gap** *(např. barevné rozložení / tvar v obrázku).*

Pokud máme ale velmi **omezenou** doménu, tak jsme schopni na základě nízkoúrovňové reprezentace a nějakým deskriptorům toto rozpoznat. *(např. dům vypadá pořád stejně vs lidé / zvířata se můžou tvářit jinak).* U některých modelů to **nejsme schopni** vysvětlit, například při rozpoznávání obličejů.

2.2.2 Účinnost vyhledávání

Efectiveness (účinnost) = míra uživatelské spokojenosti *(neplést s effectivity, efektivitou, tam jde o rychlost)* s výsledkem vyhledávání. **Relevantní objekty** by měly být ve výsledku dotazu, určit, zda se jedná o relevantní objekt jde pomocí anotací nebo referencí.

Relevance je **subjektivní** (popsána lidmi, spojena s uživatelem) a **relativní** (porovnání dvou systémů, jeden referenční, částečně je ale pořád subjektivní).

Precision (přesnost) a **recall** (úplnost) se vztahují vždy ke konkrétnímu dotazu. **Precision** = pravděpodobnost, že objekt ve výsledku je relevantní *(Google).* **Recall** = pravděpodobnost, že se relevantní objekt vyskytne ve výsledku *(medicína, chci najít problém, pokud nějaký je).*

Když dostanu ve výsledku něco, co jsem nechtěl, je to **false alarm**. Když mi ve výsledku chybí něco, co tam patří, je to **false dismissal**. False alarm se dá odfiltrovat, není to takový problém, jako false dismissal.

Měřit účinnosti můžeme pomocí **P-R curve** (křivky přesnosti a úplnosti): spočítá se přesnost na 11 standardních hodnotách úplnosti. Takto můžeme porovnávat jednotlivé systémy.

Alternativní možnost měření je **P@k** (přesnost na top k výsledcích), **F-score** (harmonický průměr přesnosti a úplnosti), **mean precision** (průměrná přesnost přes všechny úrovně úplnosti) a další způsoby pro měření přesnosti více dotazů. Můžeme měřit také **coverage** (pokrytí) a **novelty** při více dotazech – kolik je nových, dosud neviděných objektů.

3 Textové a bag-of-words modely vyhledávání

Multimédia jsou často **spojena** s textovým dokumentem (obtékání na webové stránce) nebo můžou mít textovou anotaci.

3.1 Textové modely vyhledávání

Když vyhledávám v textu, provedu předzpracování, odstraním HTML tagy, stop slova a rozbiju ho na jednotlivé **term**y, které pak ještě **stemming** a zjednoduším na kořen slov = toto je v podstatě získání **textového deskriptoru**.

Kolekce = množina dokumentů, v kolekci je **slovník** (množina všech unikátních termů s identifikátory – specifikum pro text), **dokument** obsahuje nějakou část těchto termů. **Dotaz** je pak dokumentem nebo sadou klíčových slov.

3.1.1 Boolský model

Vytvoříme **binární** vektor – term x dokument, kolekce je pak matice, reprezentujeme jako **sparse** matici, protože většina prvků bude nula. Dotazy jsou pak **boolské** dotazy "*(hora NEBO les) A NE příroda*", výsledkem je množina relevantních dokumentů **bez pořadí**. V paměti reprezentujeme **invertovaným indexem** s řetězy.

Výhodou je jednoduchost implementace, **nevýhodou** je velmi omezená možnost vyjádření, nemožnost ovlivnit počet výsledků a jejich pořadí, ani v dotazu **nejde** použít relevance feedback.

3.1.2 Vektorový model

Vylepšení boolského modelu: **dokument** je multimnožina termů, uživatel specifikuje požadované termy a jejich požadované **váhy**, nejsou zde boolské podmínky. $q = <$ *mountain(0.5); forest (0.8); nature(0.2)* $>$

Vyhledávání je založeno na **podobnosti** dotazu a dokumentů, výsledné dokumenty lze **seřadit** na základě podobnosti *(četnosti klíčových slov v dotazu a dokumentu)*, jsme schopni provést relevance feedback (**posunout** vektor dotazu).

Reprezentace zůstává ve **sparse** matici, ale místo 1 zde bude normované číslo reprezentující **frekvenci** v daném dokumentu – počítá se ale také i s **inverzní** frekvencí *(když je term méně častý, je vzácnější a specifikuje dokument více)*.

Podobnost vektorů se počítá **kosinovou** podobností vektoru dotazu a vektorů jednotlivých dokumentů, v paměti opět invertovaným indexem.

3.2 LSI vektorový model

Vektorový model předpokládá **nezávislé** dimenze. V přirozeném jazyce ale dimenze souvisí, vektorový model se proto rozšíří o **sémantiku** (matice se pronásobí koncepty *(lineární kombinaci termů)*).

Matice se touto transformací stane **hustá**, tímpádem už nejde ukládat invertovaným indexem, matice je díky tomu ale malá.

3.3 Automatické anotování

Modely založené na textovém vyhledávání jsou užitečné, pokud vyhledáváme multimédia podle anotací. Pokud máme pouze obsah, můžeme hledat podle obsahu, nebo zkusit **automatické anotování** *(generování klíčových slov)*.

Anotaci provádí předtrénované **deep learning modely**, máme anotovanou trénovací kolekci. Vždy se zkouší fetchnout nejpodobnější dokument a použít některá jeho klíčová slova.

3.4 Bag of features

Pokud nemůžeme použít textovou anotaci, můžeme zkusit vyhledávat s **vizuálními slovy** místo skutečných slov. Obrázky můžeme reprezentovat **histogramy**.

Prvním krokem je **extrakce vlastností** (obyčejnou mřížkou z JPEG, body zájmu v SIFT) → výsledkem je **sada vlastností** vektorů v obrázku. Obecně získat sémantická vizuální slova je obtížná úloha.

Při **dotazování** extrahuji vlastnosti, vytvořím dotazový vektor, a počítám podobnost. **VLAD**: vektor lokálně agregovaných deskriptorů *(každému vizuálnímu slovu se přiřadí jeho nejbližší vizuální slovo)* → není zde žádný slovník.

3.5 Word2vec

Machine-learning metoda pro hledání **vektorových** reprezentací pro **nevektorová** data (slova) – užitečné pro NLP, hledání, doporučování, analýzu sentimentů.

Vezmeme slovo s jeho kontextem a začneme ho postupně vkládat do **CBOW** (continuos bag of words) modelu. Můžeme se tedy učit hledat **slovo** na základě kontextu, nebo **kontext** na základě slova. Pak můžeme **počítat** stylem king - man + woman = queen.

4 Podobnostní vyhledávání – modely a dotazy

Bude to především o vyhledávání podle **obsahu** (textový, obrazový, audio, video). Kvůli jednoduchému modelu budou dotazy ve formátu **query-by-example**. Komplexnější dotazy se skládají právě z těchto jednoduchých dotazů *(ale uvnitř to pořád bude o query-by-example).*

4.1 Základní pojmy

- **podobnost**: relevance k dotazu, mechanismus pro organizaci objektů v databázi mezi sebou, "vzdálenost" v prostoru
- **feature extraction**: model, který vytvoří deskriptory vlastností multimédií, odpovídá kompresi – zmenšuje velikost, ale zvyšuje sémantiku
- **dissimilarity space**: prostor, který se skládá z **univerza vlastností** U (vektorový prostor, databáze je pak sada deskriptorů $S \subset U$ a **podobnostní funkce** $\delta : U \text{ x } U \to \mathbb{R}$ (geometrizuje problém vyhledávání)

4.2 Feature extraction

Vždy si musíme pečlivě vybrat, co do daného modelu vybereme tak, aby to **dobře** reprezentovalo entitu *(např. uživatel – otisk prstu, hlas*, a zároveň aby to bylo **kompaktní**.

Můžeme zkusit reprezentovat **geometricky**: ať už užším modelem (vektorový prostor), obecným (metrický prostor) nebo rozdělením na více menším prostorů. Taky můžeme reprezentovat jedním, nebo dvěma deskriptory.

Vektor: histogram u korelovaných dimenzí, čísla u nezávislých dimenzí, nebo kombinace (spojené histogramy) – **nejčastější**.

Časová řada: je to vlastně vektor s danými dimenzemi korelovanými v čase, seřazená množina

Řetězec: posloupnost hodnot, může být symbolická časová řada *(DNA sekvence)*

Další: neuspořádaná množina, geometrická data, grafy, stromy

4.3 Podobnostní funkce

Musí být **kompatibilní** se zadaným deskriptorem, splňovat potřebné vlastnosti (metrické vlastnosti, adaptace, učení). Tato funkce může taky mít různou **složitost**: $O(n)$ až $O(2^n)$

4.4 Metrické postuláty

Pokud chceme dobrý kompromis mezi možností typu deskriptoru a omezením je **metrická podobnost**:

- **reflexivita**: $\delta(x, y) = 0 \iff x = y$ *(objekt je podobný sám sobě)*
- **nezápornost**: $\delta(x, y) > 0 \iff x \neq y$ *(mezi dvěma neidentickými objekty bude nějaká nepodobnost)*
- **symetrie**: $\delta(x, y) = \delta(y, x)$ *(nezáleží na pořadí porovnávání)*
- **trojúhelníková nerovnost**: $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$ *(podobnost je tranzitivní)*

4.5 Nemetrická podobnost

Důvodem k použití je jednoduchost podobnostního modelování, argumenty proti metrickým postulátům, **robustnost** a podobnost s black-boxem.

Výhodou je také fakt, že s tímto jsou schopni pracovat i **experti v doméně**, kteří by ty matematické vlastnosti ani nemuseli umět zaručit.

Otázkou je také to, zda **argumenty** metrické podobnosti jsou v reálu **pravdivé** – je objekt podobný sám sobě? Platí tranzitivita na podobnosti?

4.6 Chytré deskriptory

Jednou z možností je použít **vysoko**-úrovňový deskriptor a **jednoduchou** vzdálenostní funkci – feature extraction dělá největší část, musím být **schopný** to ale extrahovat – vektorové prostory, podobnost nezávislých vlastností $O(n)$ (**levné**).

Druhou je pak použít **nízko**-úrovňový deskriptor a **komplexní** vzdálenostní funkci. Podobnost se agreguje v průběhu dotazu, vlastnosti jsou korelované, počítání podobnosti je tedy **drahé** a **náročné** (alespoň $O(n^2)$).

4.7 Podobnostní dotazy

Jednoduché: rozsahové (range), kNN, RkNN, similarity join, nebo **složitější** (agregace s pomocí top-k operátoru), případně se dá také využít dotazové jazyky.

4.7.1 Rozsahové dotazy

Dotazem bude vždy nějaký **konkrétní příklad** Q a **poloměr** r_Q (hranice, threshold). Zajímají nás pak všechny objekty, pro které je $\delta(Q, Q_i) \leq r_Q$. Vyžaduje, aby člověk znal podobnostní funkci δ. Vzhledem k tomu, že je daná vzdálenost, bude vždy **100% úplnost**, ale **není** předem jasná **velikost** výsledku.

4.7.2 kNN dotaz

Říkáme, že chceme **k nejpodobnějších** objektů. Opět se v podstatě jedná o geometrickou kouli, ale neznáme předem poloměr. Tento model je **přívětivější** pro koncového uživatele (známe předem velikost výsledku), ale otázkou je úplnost.

4.7.3 Reverse kNN dotaz

Mám objekt Q a počet objektů, pro které je Q **mezi** jejich **k nejbližšími** sousedy.

4.7.4 Pokročilé podobnostní dotazy

Ve velkých databázích je hodně duplikátů a kNN dotaz **ztratí** jeho **vyjadřovací sílu** – řešením je použít **distinct** kNN.

Obecně ale stále můžeme mít problém najít jeden konkrétní výsledek. Pomocí může být **skyline** operátor = hledám podle více kritérií v k-dimenzionálním prostoru.

4.7.5 Agregované dotazy

Získám různě uspořádané výsledky a seřadím je **agregační** funkcí (top-k operátor). Tak stejně jsem schopný se dotazovat ve více částech a přerankovat to.

4.7.6 Dotazy v SQL

Některé DBMS umožňují přímo práci s podobnostními dotazy a deskriptory v rámci **SQL**. Není tímpádem třeba upravovat existující systémy. Nedochází ale k indexování, a je to tedy pomalé, a je třeba dost programování.

5 Podobnostní vyhledávání – podobnostní míry

5.1 Vektorové podobnostní míry

5.1.1 Základní

Nejzákladnější je euklidovská / manhattanská (Lp) vzdálenost, předpokládáme **nezávislé** dimenze, **levné** na výpočet se složitostí $O(D)$, kde D je počet dimenzí, jedná se o **metriku**.

Další možností je měřit na základě **úhlu** – kosinová podobnost, kosinová vzdálenost, úhlová vzdálenost, **ne**jedná se o **metriku**.

5.1.2 Histogramy

V případě, že porovnáváme **histogramy**, můžeme použít váženou euklidovskou vzdálenost (převážíme jednu ze složek – *např. zelenou barvu v RGB histogramu*), Kullback-Leibler, Jeffrey, nebo třeba χ^2 vzdálenost.

Nejjednodušší formou histogramové podobnosti je **kvadratická forma** (QDF) = vážím **každou kombinaci** dvou dimenzí (oproti jiným vzdálenostem drahá, $O(D^2)$, v případě pozitivně definitní matice metrická).

5.1.3 Pokročilejší

Pokročilou možností podobnosti je **EMD** (earth mover's distance), což je minimální množství práce, aby se histogram a změnil na b, je to **optimalizační** problém → pokud $D = m = n$, pak je složitost $O(D^3 \cdot log(D))$, jinak $O(2^D)$.

5.2 Podobnostní míry pro sekvence

Sekvence (časové řady) nejsem schopný pořádně převést do vektorových dimenzí. Řešením je **DTW** (dynamic time warping), které natáhne jednu řadu tak, aby byla zarovnaná s druhou, **ne**metrická, časová složitost $O(n^2)$.

DTW: Hledám cestu v mřížce, odpovídá dynamickému programování. Cesta ale nemůže vést libovolně, ale jen **v určitém pásu**, aby se to nezaseklo → tímto omezením se složitost omezí na $O((m+n) \cdot \omega)$

Podobná může být **editační** vzdálenost řetězců *(lze použít i pro porovnávání sekvencí DNA)* – $O(m \cdot n)$, **Hammingova** vzdálenost – $O(n)$ nebo nejdelší společná **podposloupnost** – $O(m \cdot n)$.

Podobnostní míry pro sekvence se hojně používají také v **biologii**, kde se měří podobnost proteinových sekvencí *(pravděpodobnost mutace aminokyselin)*.

5.3 Podobnostní míry pro množiny

Jednou z jednodušších vzdáleností je **Jaccardova** vzdálenost: $1 - \frac{A \cap B}{A \cup B}$. Například takto můžeme porovnat předměty, které uživatelé koupili na e-shopu.

Další možnou je **Hausdorffova** vzdálenost, kde probíhá hledání nejvzdálenějšího nejbližšího **souseda** *(otisky prstů)*, **mnohoúhelníková** a **grafová** podobnost nebo také **kvadratická** (SQFD). // tyto míry pak můžeme kombinovat.

6 Globální deskriptory obrázků

6.1 Historie

Kdysi bylo vyhledávání pouze na základě **textu** *(anotace)*. Na začátku 90. let začíná vyhledávání na základě **obsahu** pomocí feature extraction, od roku 2012 se začíná rozšiřovat **deep learning** (sémantické deskriptory).

6.2 Lokální a globální vlastnosti

Vlastnost může být **globální** (vlastnost celého obrázku / videa), nebo **lokální** (vztažená k určité části). Také jsou vlastnosti buď **nízkoúrovňové** (barva, tvar, textura, melodie), nebo **vysokoúrovňové** (objekty, hlasy). Některé můžou být také **doménově závislé** (otisky prstů, obličeje).

6.3 Semantic gap

U **vysokoúrovňových** vlastností je velmi **těžké** je extrahovat *(kůň na horách)*. **Nízkoúrovňové** vlastnosti se dá velmi **dobře** extrahovat, nedají se ale příliš dobře popsat uživatelem.

6.4 MPEG

Organizace, která vytváří **standardy** pro reprezentaci audiovizuálního obsahu (postupně MPEG-1, MPEG-2, MPEG-4). Standard **MPEG-7** slouží k popisu multimediálního obsahu pomocí **metadat** – definuje schéma, ale nedefinuje způsob, jak ho získat.

Vizuální deskriptory jsou nízkoúrovňové, a popisují DominantColor, ScalableColor, ColorLayout, ColorStructure, HomogeneousTexture, TextureBrowsing, EdgeHistogram, ale taky třeba RegionShape, ContourShape, pohyb…

Barevné deskriptory

Sémanticky nejnižší informací je barevná informace. Pro člověka je vizuálně barva **nejdůležitější**, umožňuje hledání v **barevném prostoru** červená, zelená, modrá – RGB **není** příliš vhodná pro modelování, hodí se spíše použít **HSV** (odstín – hue, saturation – sytost, value – hodnota, jas).

Existují i další různé modely (**YCbCr**: světlost, míra modré, míra zelené), HMMD (odstín, minimum/maximum R, G, B hodnot). Tyto barvy pak jsme schopni **kvantizovat** (redukovat počet unikátních barev v obrázku) – můžeme rovnoměrně, nebo nelineárně *(např. člověk vnímá více zelenou barvu)*.

Podobnost barev

Používá se **kvadratická forma** – máme redukovaný počet barev, korelujeme jednotlivé barvy.

Dominant Color Descriptor

Kompaktní popis **jednotlivých barev** (**vektor** barevných komponent, **procento** pixelů namapovaných na tuto barvu, **rozptyl** barevných hodnot těchto pixelů).

Scalable Color Descriptor

Histogram barev (256 binů) v obrázku, následně použiju **Haarovu transformaci**, která mi to ještě více **kvantizuje** (zmenší, komprimuji).

Color Layout Descriptor

Celý obrázek **zmenšíme** do ikonky 8x8, na které provedeme **diskrétní kosinovou transformaci**, kterou pak můžeme porovnávat.

Color Structure Descriptor

Reprezentuje jak **distribuci** barev, tak **lokální umístění** těchto barev v prostoru → tento deskriptor je **citlivý** i na vlastnosti, které histogramové deskriptory **nevidí**.

Podobnost textur

Textura odpovídá **určitým vlastnostem** obrázku, zajímá nás **struktura** (letecké fotografie, látka, materiál, lékařské využití).

Homogeneous Texture Descriptor

Předpokládá, že je textura **homogenní**, obrázek se tedy **naškáluje** na základě **orientace** (na který vzor reaguje nejvíce).

Texture Browsing Descriptor

Porovnává pravidelnost **textury** – jestli je někde jasně horizontální, nebo vertikální distribuce = určí dominantní **směr** a pravidelnost.

Edge Histogram Descriptor

Rozseká obrázek do více **histogramů** → směrovost, vertikální, horizontální, 45 stupňů, 135 stupňů = 80 binový hranový histogram.

Face Descriptor

Specifický deskriptor, který reprezentuje obličej jako **lineární kombinaci** jednotlivých vzorových obličejů, **nepoužívá se**.

7 Deep learning

Úvod

MPEG7 deskriptory mají problém se **semantic gap** mezi popisem uživatele a nízkoúrovňovým popisem. SIFTy pracují dobře pouze s "rigidními" vzory (architektura, pevné objekty), ale například přírodu nezvládnou.

Můžeme zkusit také **rozdělení** (segmentace) obrazu na více podproblémů, tady se ale dostáváme k ještě většímu problému **sémantické segmentace**.

Co nás bude zajímat, to je právě **machine learning** a neuronové sítě = naučíme síť na lidských deskriptorech a použijeme sítě s konvolučními vrstvami.

Původní snaha byla o kognitivní podobnostní učení = vrstva dokumentů, dotazů a podobnost oddělená skrytou vrstvou, kdy se neuronová síť učila **podobnostní** funkci. Kdysi byly tyto neuronové sítě mělké a byla malá trénovací data = moc to nešlo. Dnes už máme výkonné GPU, hluboké sítě **(deep network)** a velká trénovací data = **úspěch**.

7.1 Konvoluční neuronové sítě (CNN)

Konvoluce = speciální typ propojení do určitých shluků, CNN jsou podobné buňkám na sítnici. Funguje opravdu jako lidský zrak = nejprve se pracuje s low-level prvky, pak s high-level = jde to od **více vizuálních** / méně semantických věcí k méně vizuálním, **více sémantickým**.

Konvoluční síť vizualizujeme jako jednotlivé kvádry různých rozměrů napojených na sobě, popisují **propojení neuronů** v jednotlivých vrstvách. Klasická konvoluční síť, kde jsou v každé vlně propojeny všechny neurony, by zde moc nefungovala.

Při učení se konvoluční síť **učí** konvoluční masky = filtry, například filtr pro detekci hran. Konvoluční sítě obsahují klasické vrstvy, vision vrstvy, aktivační vrstvy, klasické vrstvy a vrstvy pro zpracování chyb.

Konvoluční vrstva

V každé konvoluční vrstvě jsou neurony **uspořádány** do 3D bloku, vrstvy. Každý neuron je *perceptron*, který je propojen s neuronem v předchozí vrstvě, uvnitř to provede součet, ven se provede aktivační funkce na agregovaném vstupu a vytvoří aktivační hodnotu, která půjde dál, nebo je nula a zůstane dál.

Specifikum konvoluční vrstvy je **jednotná** sada parametrů pro celý řez 3D blokem (na jedné ose). Navíc se hodnoty přijímají pouze z určitého **receptive field**.

Pooling vrstva

Neobsahuje váhy, pouze **redukuje rozměr** vstupu pro zlepšení výkonu, může být vynechána. Typicky se používá **maximum**, průměr nebo **L2-norma**.

ReLU vrstva

Typicky se používá funkce **sigmoida** (záporné do -1, kladné k 1, jinak jde brzy k 0), to se nám v multimedia retrieval ale moc nehodí.

Používá se proto ReLU: aktivační funkce je **maximum** (záporné ignorujeme), je výpočetně levná, nezávislá na škálovatelnosti.

Loss layer

Počítá **ztrátu** mezi predikcí a skutečností = softmax (1 z k), nebo sigmoidní entropie.

7.2 Transfer learning

Znovupoužití **již získaných** dat v **jiné** doméně *(pokud máme málo dat v cílové doméně, kvůli personalizaci, spolehlivosti)*. Necháme to, co už víme, a dotrénujeme to a **dospecifikujeme** na cílovou doménu.

Zkombinujeme tedy dvě konvoluční neuronové sítě, nižší vrstvy budou obsahovat tyto generické znalosti, a vyšší vrstvy právě tyto specifické vlastnosti (= **fine-tuned CNN)**.

8 Lokální deskriptory obrázků

8.1 Detekce objektů

Globální deskriptory (MPEG7, deep learning) jsou **nepřizpůsobitelná** řešení. Lokální deskriptory jsou nezávislé vlastnosti obrázku v daném pixelu, můžou být low-level (clustery), nebo high-level (sémantické objekty, bouding-boxes).

8.2 Low-level

Asymetrický problém **hledání vzoru** (patternu) z jednoho obrázku v druhém, jde spíše o **identifikaci** než o podobnost.

8.3 Lokalizace vlastností

Sada lokálních vlastností je lokalizována na **určité místo** v obrázku (náhodným vzorkováním, detekcí bodů zájmu, segmentací), každá lokální vlastnost je popsána vektorem, a ty pak porovnávám.

Jednotlivé body pak pomocí shlukování spojím do **clusterů** (shluků), se kterými následně můžu pracovat.

8.4 Shlukování vlastností

Může být provedeno pomocí **k-means** = vyberu náhodně centroidy, rozdělím databázi na k podmnožin podle centroidů, pak přepočítám **centroidy**, zapomenu shluky a počítám znovu.

8.5 Vytvoření deskriptoru

Vytvářím množinu **reprezentativních vzorků**, deskriptor je pak množina vlastností nebo množina centroidů.

Nebo můžu použít klasický **bag of features**, jednotlivé clustery si namapuji na slovník, a deskriptor je pak histogram v doméně těchto vizuálních slov, nebo použiju každou vlastnost jako **deskriptor**.

8.6 Měření vzdálenosti

V případě **množin** počítám QDF, Hausdorfovu vzdálenost (metricky) nebo nemetrické vzdálenosti množin, v případě **bag of features** počítám klasickou L_p vzdálenost nebo

8.7 Detekce zájmových bodů

Vlastnosti jsou založeny na **bodech zájmu** (je zde možnost detekovat hrany nebo tvary), určeno specificky pro **zpracování obrazu** (detekce objektů, modelování 3D scén, sledování lidí).

Použít můžu první derivaci (Sobelův filtr) – používaný pro rozmazání obrázku, nebo **druhou derivaci** Gaussiánu, ta je pak škálovatelná.

8.8 Zájmový bod

Zájmový bod by měl tedy být **dobře definován** a pozicován, měl by být opakovatelný i při změně jasu, měřítka, barvy. Dříve používány pro detekci hran, dnes se používají obecně k získání informací o **zájmových regionech** pro matchování obrázků a poznávání obrázků.

8.9 SIFT

Scale-invariant feature transform: **detektor** a **deskriptor** lokálních vlastností v obrázku, najde zájmové body, vygeneruje histogram 128 vektorů reprezentujících tyto body *(odstraní málo kontrastní body, body na hraně)*.

SIFT popisuje 4x4 **histogramy** na 16x16 vorku **pixelů** okolo zájmového bodu, tento histogram obsahuje 8 binů relativních k orientaci zájmového bodu, neobsahuje žádné **absolutní** úhly.

Image matching se SIFTem

Používá se bag of features s vizuálními slovy (1 slovo = SIFT, id obrázků v invertovaném souboru), po matchnutí se udělá ještě geometrický **re-ranking** *(aby se vrátila absolutní hodnota)*.

Této geometrické verifikaci se říká **hamming embedding**, kromě porovnávání vizuálního slova se porovnává i jejich hammingova vzdálenost.

Detekce objektů

Používá se strojové učení, následně převádí nízkoúrovňové deskriptory na vyšší sémantiku. Existují modely obrazové *(konvoluční vrstvy + detekční vrstvy)* i pro video.

9 Vyhledávání v 2D a 3D modelech

9.1 2D model

2D tvar je v podstatě stín, **obrys** objektu (mnohoúhelníky, množina bodů), může se jednat o **projekci 3D** modelů. Tvary můžeme získat také **vektorizací** objektů z **rastrových** obrázků *(předpoklad jednoduchých izolovaných objektů – produktové fotografie).*

V těchto modelech můžeme pracovat s deskriptory popisující **tvar**, nezávislými na **otočení**, natáhnutí *(projekci 3D objektu pod jiným úhlem).*

MPEG7 obsahuje také dva deskriptory popisující tvar – **Region Shape**: předpokládá tvary s obsahem, kde nás zajímá i **více** než jen obrys *(simuluje tedy částečně i popis textury – díry v hrnku* a **Contour Shape**: ten popisuje právě **obrys** bez vnitřku.

2D tvar **uzavřeného** mnohoúhelníku lze reprezentovat také zápisem **centroid**u tvaru, **rozbalit** tento tvar do časové řady a **porovnávat** pomocí DTW…

U **otevřených** mnohoúhelníků reprezentujeme úhly mezi přilehlými hranami do **časové řady** a to pak porovnáváme, je to ale **citlivé** na šum.

Časové řady u **shape matching** můžeme porovnávat pomocí nejdelší společné podposloupnosti (**LCSS**), pokud třeba vím, že mi bude chybět nějaká část.

9.2 Proteinové struktury

Proteiny jsou v reálu sice **3D struktury**, jsme ale schopni je zjednodušeně **reprezentovat** jako textové **řetězce** AT, CG, jsou tedy v podstatě 1D.

Při porovnávání proteinů tedy nejprve porovnáme tuto jednu dominující **dimenzi**, a následně to **ověříme** v **3D** struktuře.

9.3 3D modely

Při vyhledávání 3D modelů musíme umět tyto modely **reprezentovat**, zároveň se ale **mění** v čase (pohyb = zachytíme jiné pózy). Můžeme taky hledat základní low-level **geometrickou** podobnost, nebo high-level sémantickou.

3D deskriptory musí být **nezávislé** na rotaci, posunu, škálování a odrazech, robustnost vůči jednotlivým detailům *(jak moc je to jemné).*

Zadaný objekt vždycky **normalizuji** (pokusím se odstranit otočení, posun, škálu, šum → abstrahuji objekt *(typicky povrch)* → transformuji do **vektoru vlastností** a porovnám.

Nevýhodou těchto **globálních** deskriptorů je citlivost na **pózy** a neschopnost částečné podobnosti. Používají se proto **lokální** deskriptory.

Jedním z lokálních deskriptorů v 3D modelech jsou **zájmové body** *(místo jasové funkce z obrázků používáme tvarovou funkci a extrémy v rámci geometrické křivosti* → jedná se v podstatě o "3D SIFTy".

Na základě zájmových bodů jsme v 3D prostoru schopni detekovat **zájmové regiony** *(hlava, noha u člověka, zvířete)* a ty pak v geodetickém prostoru **clusterovat**.

10 Indexování multimédií

Je rozdíl mezi **efektivitou** (effectivity) a **výkonem** (effectiveness) – rychlostí vyhledávání, předpokládáme **drahý** výpočet vzdáleností a chceme si usnadnit počet výpočtů → **redukovat** jejich počet *(počet I/O operací, čas běhu)*.

10.1 Lower-bounding

Používáme levný **lower bound** místo drahé vzdálenostní funkce *(v metrickém modelu závisí na pivotech a metrických postulátech, umožní rozdělit metrický prostor a rozdělit databázi v závislosti a tomto modelu)*. Levná **vylučovací** funkce, drahá vzdálenostní funkce.

Metrické postuláty: reflexivita *(objekt je nejvíce podobný sám sobě)*, nezápornost, symetrie, trojúhelníková nerovnost.

Předpokládáme jednoduchý dotaz, query ball (o poloměru r od bodu Q), vysokou **selektivitu** *(vracím malou část databáze)*. Vyberu tedy **pivot**, znám $\delta(Q,P)$ a $\delta(P,X)$ a díky trojúhelníkové nerovnosti vím, že $|\delta(Q,P) - \delta(X,P)|$ je lower bound *(dolní mez)* $\delta(Q,X)$, pokud je tedy větší než **r**, vzdálenost je taky větší než r.

Vzdálenosti k pivotům si pak můžu uložit do metrického **indexu**, a předpočítávat si je *(vyhledávání je pak mnohem rychlejší)*.

Jednotlivé lower/upper boundy můžu samozřejmě **kombinovat** (hyper-plane partitioning) nebo počítat lower bound ke **clusteru** *(ne k jednotlivým pivotům)*. Pak potřebuji ty regiony nějak v metrickém prostoru organizovat, ukládat.

Pivot je dobrý, když je **blíž** k Q nebo x *(nesmí být stejně blízko ke všem)*, jinak je lower bound 0.

Pivoti můžou být **globální** (statická množina, každý lower-bound je počítán ze stejné množiny pivotů) nebo **lokální** (dynamicky vybrané během indexování, každý lower bound je pak počítán k jiné množině).

10.2 Metrické přístupové metody (MAM)

Algoritmy a struktury umožňující **efficient** (rychlé) podobnostní vyhledávání v metrickém modelu, založen na **metrickém indexu** (perzistentní, nebo v paměti), index se musí vybudovat.

Indexovatelnost: metrické postuláty nezaručují efficient (rychlé) indexování a vyhledávání, ani vhodné rozmístění v prostoru. Indexovatelnost je tedy **schopnost** databáze se rozdělit na **rozdílné** třídy (clustery) → vnitřní dimenze, faktor překrývání clusterů…

Vnitřní dimenze *(intrinsic dimensionality)*: $\rho(S,\delta) = \mu^2/2\sigma^2$, kde μ je průměr a σ^2 je rozptyl rozložení diskrétní vzdálenosti v datasetu $\mathbf{S}$ → nízké ρ (pod 10) znamená **dobrou strukturu**, vysoké = špatná struktura *(objekty jsou skoro stejně vzdálené)*.

Ball-overlap: všechny body si představím jako koule v prostoru, a pokud poměr všech dvojic, které se protínají ke všem bude velký *(vše se protíná)*, bude to špatně indexovatelné, jinak dobře indexovatelné.

U metrických metod hledáme rychle v databázi, používáme **metrické indexování**, závisející na lower-boundingu pomocí **pivotů**, musí zde platit metrické postuláty.

10.3 Tabulky pivotů

Třídy indexů, mapují data do prostoru pivotů, AESA (každý datový objekt je pivot, náročná **kvadratická** konstrukce, ale **konstantní** hledání), LAESA (je vybráno jen **k** objekůt jako pivoty), dotazovaní probíhá v L_∞ metrice.

Další metody závisející na tabulkách pivotů jsou TLAESA *(optimalizace pomocí struktury závisející na stromu)*, ROAESA *(redukce pro přechod matice vzdáleností)* atd.

10.4 Stromové indexy

Reprezentují metrický prostor přímo v hierarchické rovině.

- gh-tree (2), **GNAT** (n): reprezentují pomocí reprezentace v **nadrovinách**, binární strom, statická konstrukce z dvou (n) pivotů, rozdělení na **poloprostory**, pak lze filtrovat v binárním (n-árním) stromu
- vp-tree, mvp-tree: reprezentují pomocí **metrických koulí** (prstenců), rozvrství prostor na **vrstvy** jako cibule, vybere se pivot a k němu vztahuji další prostory *(vrstvy se můžou protínat u mvp-tree)*.
- M-tree: modifikace R-stromu do metrického prostoru, **hierarchie** vnořených koulí, vhodný pro **sekundární** ukládání, opět umožňuje **rychlé** procházení dotazu
- PM-tree: M strom, který ještě navíc používá **množinu** pivotů, každá koule je tedy ještě redukována prstenci danými pivoty

10.5 Hashované indexy

D-index: závisí na hashi, vypočítá si hashovací funkci *bps* (ternární hash) rozdělující na základě koule, která rozdělí každý objekt do určitých **bucketů**.

10.6 Bezindexové MAM

D-file: originální databáze za použití sekvenčního skenování, ale používá **D-cache**: strukturu, která si pamatuje již spočítané vzdálenosti a poskytuje **lower boundy** požadovaných vzdáleností. Není třeba žádné indexování.

10.7 Míry nákladů

Zajímá nás náročnost **indexování** a **dotazování**, to může být v kontextu počtu volání **vzdálenostní funkce**, ceny I/O operací *(počet přístupů na disk, RAM/HDD/SSD, sekvenční, náhodný)*, ceny vnitřních volání *(výpočetní výkon)*, ceny v **reálném** čase *(míra zkompilování)*.