Vyhledávání na webu – NDBI038y

Šimon Jůza, 2023/2024. Letní semest

Celkově o:
hledání, prohledávání, dotazování (querying)
webové stránky, multimédia a hledání pomocí nich

1. *Web space, search engines, web retrieval modes*

a)

Web 1.0 (PERSONAL) vs Web 2.0 (Blogy, sociální sítě)
Web Space je WWW, „internetový graf webových stránek“, http komunikace
web page – hypertextový dokument v HTML (plus PHP atd.) s odkazy a multimédii, je to vrchol grafu (multimédia -> anotace jako popis)
web site – webový podgraf související s web pages
data: strukturovaná (schémata – SQL, relační, …) x nestrukturovaná (bez schémat – text, slova)

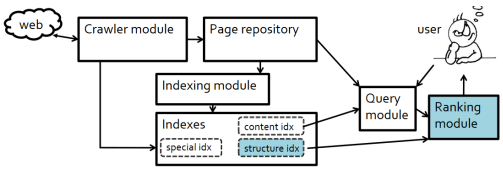
b)
web search engine – příjem informací na webu, založený např. na textu a ranku (**PageRank)**
(jiný je „**meta-search engine**“ – vícero web search enginů použitých naráz)

- **elementy search enginů**:

crawling – stahování obsahu (konkrétně webových „stránek“ – zde pages)
--- krátký program s instrukcemi pro robota, jaké stránky má stáhnout
--- může být náročné na výkon, problémy s etikou (šedá zóna)
---- **ethic crawling = omezení frekvence a objemu data stažených**

indexování – úprava do formátu vhodné k prohledávání
--- vestavění indexu, postupně zpracovávám (struktury, caches, …)

(pro)hledá(vá)ní – získání chtěného obsahu pomocí dotazu (query)
--- tradiční vyhledávací enginy buď indexují podle celý slov či hledají slova



c)
<u>web retrieval modes</u> (lze je i kombinovat)

query – výsledky *neseřazený set objektů* (ranking a re-ranking dle výsledků potřeba); explicitně formuluji query (slova, obrázek) -> různé query modely (keyword-, content-based)

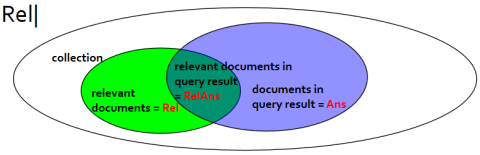
browsing – svůj dotaz přímo neznáme, iterativní navigování databází; výsledky jsou subsety databázových objektů; browsing modely: **explicitní graf** (linky URL), **virtuální** (série dotazů), **implicitní** (podobné obrázky na Googlu)

filtering (dopuručování) – explicitní (statické dotazy, RSS, odběry na YouTube) x **implicitní** (doporučování, „mohlo by se vám líbit“, personalizovaná reklama na Googlu); výsledek je dynamický a mění se v čase **kombinace** --- dotaz v e-shopu a seřazení dle recenzí

Modely potřebuje proto, že full-text search by byl v rámci webu opravdu náročný a mnohdy nepřesný. Full-text modely společně s indexováním mohou značně ulehčit a zpřesnit hledání.

web retrieval models – modely a indexovací techniky pro webové search enginy
--- tři základní: **booleovský** (s invertovaným indexem)**, vektorový** (s invertovaným indexem)**, pravděpodobnostní** (viz níže)

- P = |RelAns| / |Ans|
- R = |RelAns| / |Rel|



dokument – entita obsahující „full text“
kolekce – set dokumentů
term – slovo či fráze vyskytující se v dokumentu
slovník – set rozdílných termů objevujících se v dokumentu

předzpracování – odstranění nepodstatných slov (spojky, základní slovesa) a různých tvarů (jdi, jde, jdu, jít, jíti -> jít) --- získáme tak efektivně pouze cca 20 % původních termů!

kvalita příjmu informací – jak **přesné** (bude uživatel spokojen?)
relevantní dokument – přesně odpovídá či souvisí s požadavkem uživatele

precision (přesnost) – pravděpodobnost **relevance dokumentu z výsledku hledání**
recall (odpovídá?) – pravděpodobnost, **že je relevantní dokument výsledek** (pro uživatele)
(viz obrázek výše)

11 standard levels of recall
1) let's have a ranked query result
2) enlarging the query result till a certain level of recall is reached
3) precision is computed
4) goto 2)

QRref – ten query výsledek, který chci

QRsys – ten query výsledek, který mi vypadne z vyhodnocujícího systému

relative precision – dokument padne do „sys“, ale NEBYL to správný výsledek; **falešný poplach** (více poplachů = menší relativní přesnost)
relative recall – dokument nepadne do „sys“ i přesto, že to BYL správný výsledek; **falešné zamítnutí** (více jich = menší relativní recall)

P-R křivka – *poměr precisionu vůči recallu* (postupným zvětšováním recallu se doberu „nejlepšího“ výsledku – nikdy to nemusí být 100% přesné, kompromisy v poměr P a R)

2. *Boolean model of information retrieval* **BOOLOVSKÁ ALGEBRA + TEORIE MNOŽIN**

a)
- **dokument** = set termů, každý dokument **jako binární vektor** dimenze **m**
- **term =** set dokumentů, kde se vyskytuje (inverse), každý term tak reprezentujeme jako **binární vektor** dimenze **n**

- výsledkem je **term-dokument matice ŘÍDKÁ**
  ---- sloupce jsou dokument-vektory, řádky term-vektory
  ---- většinou hodně plná nul, protože hledám jen její velmi **malou pod-matici**

		document vector
		Antony and Cleopatra
term vector	Antony	1
	Brutus	1
	Caesar	1
	Calpurnia	0
	Cleopatra	1
	mercy	1
	worser	1

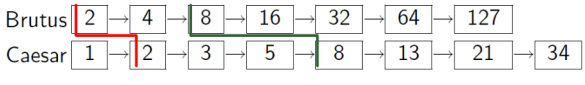
- příklad: chci zjistit, kde je „Antony“, aha, je v prvním dokumentu

- lze používat jako **operace** (AND, OR, NOT)

- **implementace:** matice je plná nul (řídká),

inverted index – pamatuji si u term vektorů, kde se vyskytly, pamatuji si ID dokumentů, kde byly
- časté termy neindexujeme (hodně dlouhé)

- q = Brutus AND Caesar
- result = (2, 8)



- sorted a static (aby se nemuselo znovu)

- query bráno jako merge sort
caesar and **(**brutus and calpurnia**)**
(caesar and brutus**)** and calpurnia **LEPŠÍ**

- optimalizace na základě frekvencí, velikostí atd.

VÝHODY: **jednoduchá implementace**, buď sedí či nesedí (najdu či nenajdu), efektivní implementace, ideál, když vím, co hledám přesně

NEVÝHODY: nutnost přesnosti, nešlo by řešit přes meta model (?), nemá ranking (hodně velké či žádné výsledky), **není uspořádané**

b)
Rozšířený booleovský model
- řeší problém např. toho, že query „mouse“ u normálních booleovského modelu nenajde „mice“
--- termy s váhami, částečné shody, konečně **seřazené výsledky**
--- na rozdíl od klasického modelu tedy nabízí i **částečně relevantní dokumenty**

! jedná se o spojení booleovské algebry s vektorovým modelem

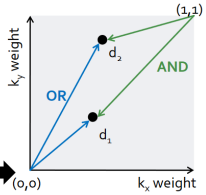
--- výpočet přes váhy termů v dokumentech, z čehož vyvozujeme „**skóre relevance**“ vůči query, to uděláme pomocí konjunktivní či disjunktivní formule a poté pomocí euklidovské vzdálenosti a rekurze (z toho mi vypadne seznam od nejlepšího po „nejhorší“)

--- lepší výsledky, ranky, efektivnost výsledků; avšak velmi náročné na výpočet a provozování

- q = k_1 AND k_2 AND ... AND k_t
  the relevancy of a document d_j to q is computed as
  $relev(q,d_j) = 1 - \sqrt{\frac{(1-w_{1,j})^2 + (1-w_{2,j})^2 \ldots + (1-w_{t,j})^2}{t}}$
- q = k_1 OR k_2 OR ... OR k_t
  here the relevancy is computed as
  $relev(q,d_j) = \sqrt{\frac{w_{1,j}^2 + w_{2,j}^2 \ldots + w_{t,j}^2}{t}}$

note: in the formulas we only consider terms appearing in the query q

what's the intuition behind the geometry?



q = (k_1 AND k_2) OR k_3 is recursively evaluated as

$$relev(q,d) = \sqrt{\frac{(1 - \sqrt{(\frac{(1-w_1)^2 + (1-w_2)^2}{2})})^2 + w_3^2}{2}}$$

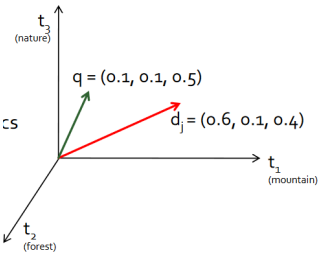
$$relev(q,d) = \sqrt[p]{\frac{(1 - \sqrt[p]{(\frac{(1-w_1)^p + (1-w_2)^p}{2})})^p + w_3^p}{2}}$$

3. Vector model of information retrieval ***TERM(VECTOR)-BY-DOCUMENT MATICE***

- model založený na **podobnosti obsahu** (kognitivní model)
- dokument je zde **vektor ve vícedimenzionálním prostoru**
--- vzdálenost **query vektoru** a **dokument vektoru**

- parametry -> jde vhodně parametrizovat, ranking, podporuje i feedback k relevanci výsledků

bag of words (BoW) model – dokument je tedy „bag“ termů (v tašce může být vícekrát stejný element)
-- **slovník tvořen m** termy (viz Bool.)
-- **dokument d** modelovaný vektorem dimenze **m**
(tedy dimenze dle počtu termů ve slovníku)
-- vysoce-dimenzionální vektorový prostor
-- **query** = „bag“ termů (jako dokument)



f_{ij} = frequency of term t_i in document d_j

tf-idf, popular *weighting scheme*

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (n/df_i)$$

jak určit váhu?
- na základě **frekvence termu**
- důležité termy tedy mají vyšší váhu
- na začátku normalizovaný na 0.1
- **RŮZNÉ ZPŮSOBY VÁŽENÍ:**
  tf-idf ------ frekvence v doc a kolekci, log

$tf_{ij} = f_{ij} / max_i\{f_{ij}\}$, where max returns the highest frequency of term t_i over the entire collection

Given a document **d** containing terms with given frequencies in **d** as
d = < mountain(3), forest(2), nature(1) >
Assume 10,000 documents and document frequencies of the terms as mountain(50), forest(1300), nature(250)

Then the *tf-idf* weights of the terms in **d** are:

mountain:	tf = 3/3; idf = $\log_2$(10000/50) = 7.6;	**tf-idf = 7.6**
forest:	tf = 2/3; idf = $\log_2$(10000/1300) = 2.9;	**tf-idf = 2.0**
nature:	tf = 1/3; idf = $\log_2$(10000/250) = 5.3;	**tf-idf = 1.8**

více časté termy = více důležité

„podobnost“ vektorového modelu
-> euklidovská vzdálenost není vhodná,
lepší přes úhly (cosinová podobnost)
vector range query ---- vyhodnocení a pak seřazení
-> query vector q, „hranice podobnosti“ určuje, kdy začnu registrovat daný výsledek jako dobrý
- dodáváním feedbacku a dodatečných informací lze „shiftovat“ a „posouvat“

implementace
-> velmi podobná booleovskému modelu --- **inverted index**, ukládám nejen id, ale i „term weight“
- ve výpočtu s používá kosinová podobnost na rozdíl od booleovských výrazů

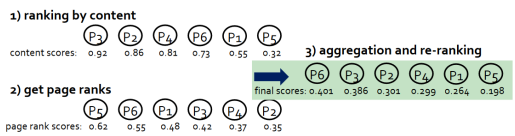
VÝHODY – jednoduchý a dobře definovaný přístup, query-by-example (ne nutně text)
NEVÝHODY – příliš jednoduché dotazy, nemá sílu vyjadřování booleovského modelu

další vektorový model
Latent semantic indexing (LSI) --- vylepšení vektorového modelu, řeší nedokonalost smyslu jako např. „George Bush“ a „garden bush“
-- high-level koncepty -> koncept-báze (vektory) U, koncept = lineární kombinace termů, seřazeno dle důležitosti
- rank-k SVD (přes k již ignorujeme)
- stále kosinová podobnost, avšak **NE inverted index**
- HUSTÝ VEKTOR DOTAZU, HUSTÁ !!!*CONCEPT-BY-DOCUMENT*!!! *MATICE* JAKO VÝSLEDEK
- výhody: založeno na konceptech, řeší částečně problém synonym a homonym, redukce dimenzí
- nevýhody: v praxi ne zcela vhodně jazykově funguje, matice je hustá, složité

4. *Link analysis and the web page ranking*

a) **Web graph** --- **vrcholy** grafu jsou webové stránky, **orientované OHODNOCENÉ hrany jsou URL dál**
inlink = link někam (na Youtube) KAM -- zde nevíme, kolik inlinků sem existuje
outlink = z nějaké platformy (Discord) někam ODKUD -- zde můžeme vědět kolik (dle HTML)
hub = stránka s hodně outlinky
authority = stránka s hodně inlinky (může být i obojí zároveň)
- analýza linků --- doporučení (1 -> 2), relevantní stránky (1 <-> 2), co-citace (1 -> 2, 3), tranzitivní doporučení (1 -> 2 -> 3)
---- lze hodnotit **popularitu stránek dle inlink statistik (ne moc část=)**

b) PageRank (POZDĚJI GOOGLE)
-> zkombinováno s query-dependent content rankem
- **ohodnotit dle obsahu (vektorový model), získat ranky stránek jako query result, pak přerankovat při pohledu na page rank skóre**



- při iteraci se našel problém:
některé stránky rychle klesají na skóre a jiné jsou na vrcholu; problémy s cykly

$$\pi^{(k+1)T} = \pi^{(k)T} H$$

- vylepšená formule:
matice je bohužel velmi hustá, takže najdeme bez-maticový výpočet „power method“
--- **Markov chain, transition probability matrix H**
--- H je stochastic, irreducible, aperiodic

GOOGLE MATICE

- Brin and Page defined the **Google matrix** as

$$G = \alpha S + (1-\alpha)\frac{1}{n} e e^T \quad \text{or simply} \quad G = \alpha S + (1-\alpha)E$$

where $E = \frac{1}{n} e e^T$

where $\alpha \in (0,1)$ is a parameter

- the second component in the formula ensures the matrix **G** is primitive – completely dense and positive (**no zero inside**)

- je tedy matice **PRIMITIVNÍ, HUSTÁ A POZITIVNÍ, NEOBSAHUJE NULY**

  - a dangling page gets outlinks to all pages (intuition on **S**)
  - from time to time, following just the outlinks is "boring", so the surfer "teleports" randomly anywhere (intuition on **G**)
    - the α parameter controls the proportion of "following" and "teleporting" of the surfer

---- parametr **ALFA zde udává pravděpodobnost, že půjde na stránku, na kterou z aktuální stránky nevedou odkazy (NEPŮJDE PŘES OUTLINKY)**

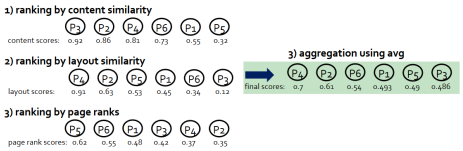
- since **G** is the desired matrix, the famous PageRank formula is: $\pi^{(k+1)T} = \pi^{(k)T} G$ (short version)

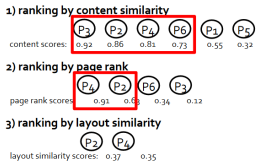
$$\pi^{(k+1)T} = \pi^{(k)T}(\alpha S + (1-\alpha)E)$$ (expanded version)

5. *Search engine ranking and optimization (SEO)*

---- potřeba celkově hodnotit stránky při výsledcích, **celkový rank a částečné rank funkce** (vektorový model + PageRank + další…)

- následně **finální rank** --- buď v *jednom kroku:*
  - např**. top-k operátor**
  - Faginův algoritmus
    v základu paralel, stačí pohlížet random přístupy
  - Threshold algoritmus ---- **pod top-k operátor**
    N-TICE, DEFINOVANÉ USPOŘÁDÁNÍ





Faginův algoritmus (FA):

1. **Průběh**:
   - FA má dvě fáze: první fáze je paralelní skenování seznamů, dokud není k-tá položka viděna ve všech seznamech. Ve druhé fázi se zjišťují přesné hodnoty položek, které byly viděny v první fázi, a vypočítá se jejich skóre.
   - Poté, co je viděna k-tá položka ve všech seznamech, algoritmus přejde do druhé fáze, ve které dokončuje kontrolu položek viděných v první fázi.
2. **Výkonnost**:
   - FA může být v některých případech méně efektivní než TA, protože může skenovat více položek, než je nezbytně nutné.
   - Má pevnou horní hranici na počet položek, které musí být skenovány (v nejhorším případě může být nutné přečíst všechny položky v každém seznamu).

Threshold algoritmus (TA):

1. **Průběh**:
   - TA také skenuje seznamy paralelně, ale sleduje prahovou hodnotu, která je dynamicky upravována během průběhu algoritmu.
   - Algoritmus vypočítává prahovou hodnotu na základě nejlepších (nejvyšších) aktuálně viditelných hodnot a využívá ji k určení, zda již byly nalezeny všechny top-k položky.
2. **Výkonnost**:
   - TA bývá obecně efektivnější, protože může často skončit dříve než FA díky dynamické prahové hodnotě.
   - Má tendenci číst méně položek, protože lépe využívá informace o aktuálních hodnotách k rozhodování o ukončení algoritmu.

Shrnutí:

- **Faginův algoritmus** má dvě jasné fáze (skenování a hodnocení) a může být jednodušší na pochopení a implementaci, ale nemusí být tak efektivní jako Threshold algoritmus v praxi.
- **Threshold algoritmus** je sofistikovanější, s dynamicky se měnící prahovou hodnotou, což mu umožňuje často skončit dříve a číst méně položek.

či *více krocích*:

- efektivnější (dělat vše v jednom kroku je těžké)

SEO (Search engine optimization)
-- **jak mít dobře ohodnocenou stránku?**
-- soft-skill pro tvorbu dobrých webových stránek
musí být dobré pro člověka i „robota“ (engine)
podstatné jsou **názvy souborů**, **tagy**, **headery**, optimalizace obrázků, outlinky

filenames – relevantní k obsahu stránky, co nejkratší URL, „“ = „-“ v URL
<title> tag – primární rank faktor
header tags – h1 ---- h6 (h1 je i pro engine nejpodstatnější)
meta tag „klíčová slova“ – podstatné robotické fráze (mix trefného a krátkého, obecné i specifické ve vyvážené kombinaci)
textové úpravy (zvýraznění) --- taky posiluje hodnocení webu
„alternative text“ pro obrázky – obrázek = zajímavější, ale musí jít přečíst „alt“

v textu opakovat „klíčová slova“, aby to dobře vyhodnotil např. vektorový model, **neduplikovat obsah** (to snižuje rank a dává penalty), **aktuálnost stránky**, blogy (**uživateli generovaný obsah**), **nákup zpětných odkazů je OK**

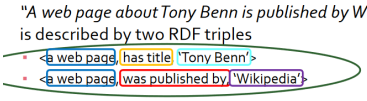
DŮLEŽITÉ -> pro **crawlery SITEMAP a ROBOTS.TXT** **(htaccess ne, to je pro Apache)**

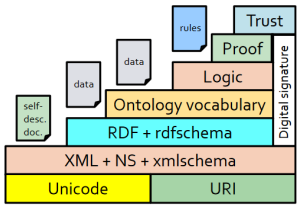
GWT --- Google Webmaster Tools !
-> queries, linky na stránku, důležitost slov, crawling errors (**NESLEDUJÍ PŘÍSTUPY Z RŮZNÝCH DOMÉN)**

6. *Semantic web and Linked data*

syntaktický web --- dnešní internet, založen na poměrně nízkém levelu sémantiky
-- informace, hyper-média, aplikační prostor, propojení lidí
-- nebere však hlubší otázky s nějakým smyslem navíc (či požadavkem)

sémantický web --- pokus o hluboké porozumění otázkám uživatele
-- ontologie: specifikace znalostí jako koncepty a vztahy mezi nimi, společný slovník (bytosti -> zvířata, rostliny)
-- RDF pro reprezentaci ontologií, RDFS (třídy, vztahy, …



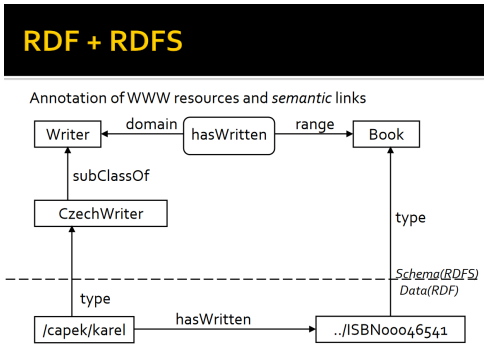


HTML vs. XML
-- html jako „prezentace“
-- XML (extensible markup language): **strom, skoro strojově-čitelný formát (METADATA)!!**
-- XSTL (stylesheet**) XML -> HTML PŘEVOD**

in RDF/XML:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
        <dc:title>Tony Benn</dc:title>
        <dc:publisher>Wikipedia</dc:publisher>
    </rdf:Description>
</rdf:RDF>
```

OWL – web Ontology Language
-- kompatibilita s **RDF schémata (model znalostí)**



FOAF – friend of a friend
-- strojově čitelné ontologie popisují lidi
- založené na RDF a OWL

SPARQL – query language pro RDF (*Select, Construct, Ask, Describe)*

pro přechod na sémantický web by se mnoho stránek
muselo přepsat… NEREÁLNÉ

Linked data
- jak správně publikovat strukturovaná data na webu dle architektury webu
URI = jméno pro věci, http URIs pro jednodušší nalezení
když si někdo najde to URI, dodej dobré RDF informace
přidej RDF s odkazy na související témata jinde

--- data by měla být globální a postavena na daných principech, entity propojeny linky

DBpedia – strojově čitelná Wikipedie

7. *Personalized search and social context*

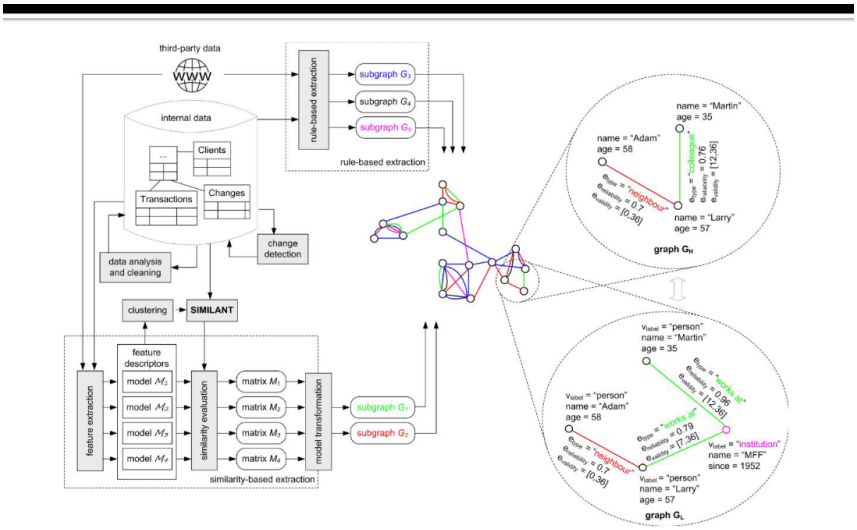
--- doporučení, filtrování, …
na základě top-k objektů se vybírá uživateli podle daných metod, co se mu ukáže přednostně
input: uživatel s historií, output: top-k předmětů v listu s ohodnocením výsledků
jak? --- **naber feedback, nauč se preference, doporučuj top-k věcí**
feedback: explicitní (ratings), implicitní (těžší – počty návštěv, prokliky…)

CF (kolaborativní filtrování) ---- uživatel-věc matice
- snaží se přirovnat uživatele k jinému uživateli a podle toho doporučit
- kosinová podobnost
- může dopadnout „špatně“, nedostatek dat, data jsou řídká

CBR (Content-based recommendation) --- co umí oblíbená věc
- na základě koupí a oblíbených věcí vyvodí doporučování, co uživateli nabídnout dále
- k nejbližších sousedů --- podle toho usuzujeme hodnocení pro ještě neviděné věci
- vektorový model (bag of words), podobnostní modely

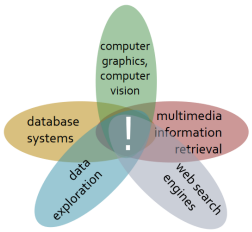
KBR (Knowledge-based recommendation)
- dané informacemi o uživateli, podle toho doporučuji
složité, trvá dlouho

Social Information retrieval
- multigraf
- social PageRank
(hashtagy, liky, reposty)
--- doporučovací systémy
(top-k pro kamarády)



Holubová, M. Svoboda, T. Skopal, D. Bernhauer, L. Peška. **Advanced Behavioral Analyses Using Inferred Social Networks: a Vision**, *TIR 2019*, Springer, 2019

8. Introduction to similarity search in multimedia databases



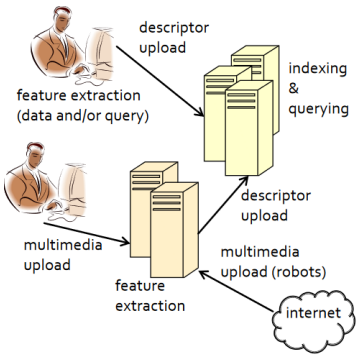
--- hledání podobností v multimédiích
    více druhů:
        tradiční (obrázky, videa, zvuky)
        rozšířené (nestrukturované – senzory, tvary)

kontext X obsah (obrázky, tvary, zvuk)
    kontext -> nějaký popis/anotace obsahu, tagy, GPS
        (manuální anotace – tagy, automatizovaná – dle lokace filtr)
    obsah -> RAW, bez popisu (samotné pixely, zvukové vlny)
- metrické indexování (rychlé hledání)

text-based hledání ---- může zaručit sémanticky dobré queries, fulltext indexování je dobré (booleovský či vektorový model), **kontextové hledání** bude velmi mocný nástroj; náročné na přímé hledání **manuální prací** (náhrada ML?)

struktura a sémantika dat (příklady)
- relační databáze (silná struktura – jasně dané, silná sémantika – uživatel ví, jak se ptát)
- **celo-textová databáze** (slabá struktura – jen slova, silná sémantika – query a text se jednoduše shodují)
- **multimediální databáze** (slabá struktura – pixely, slabá sémantika – na pixely se ptá jak?)

content-based hledání
-> když nemáme žádný kontext či anotaci, tak musím hledat jen přes obsah
(lze i kombinovat pro větší přesnost)
- **similarity search model** (do detailu viz níže)
    --- potřebuji **extrahovat prvky** např. z obrázků --- struktura
        -> lokální informace + jejich kombinace, aby se dosáhlo **expresivního popisu**
        -> vektory (histogramy), sety, seřazené sety (čas)
    --- **similarity function** (porovnání popisů, částečně se snaží o sémantiku)
        -> záleží na struktuře **popisovače**
        -> např. přes vektorové vzdálenosti (kosinové vzdálenosti s úhlem)
        -> či **adaptivní vzdálenosti,** nebo **sekvenční vzdálenosti**
    --- dotazování:
        -> **query-by-example concept**
            dostanu multimédium, vyextrahuji popisovač **q**
            přes **similarity f** porovnám **q** se všemi popisovači z databáze
            zobrazím nejbližší výsledky ke **q** (range query, k-nejbližších sousedů)
    --- aplikace: **retrieval** obrázků pomocí:
        globálních prvků (features)
            barvy (úpravy barev histogramů)
            struktura barev
        lokální prvky
            SIFT/SURF (body, feature vektory) – postupně namapuje obrázek
        deep learning
            -- např. zjistit všechny lidi z kamerového záznamu (kolik jich je v jeden moment na obrázku)



retrieval tvarů pomocí:

časové řady

-- dynamický vzhledem k času (postupně měřím) DTW

-- či **nejdelší stejný úsek** LCSS

retrieval zvuků pomocí:

zvukové stopy

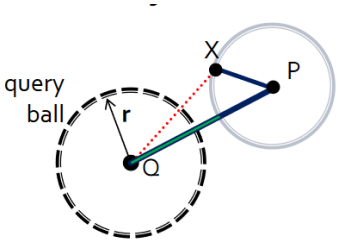
- pomocí vlnové spektrální informace
- lokální pro časový úsek (spektrogram), vícedim. časová série

melodie

- notový zápis převeden do 2 D bodů určujících výšku

--- výkon: sice drahé, ale velmi dobré pro uživatele, **indexování podobnosti** (zde je velmi podstatná náročnost **vzdálenostní funkce**) **LEPŠÍ VÝKON**

-> **metrické řešení**: trojúhelníková nerovnost, metrický index, hledám jen části



The task: check if **X** is inside query ball
- we know **δ(Q,P)**
- we know **δ(P,X)**
- we do not know **δ(Q,X)**
- we do not have to compute **δ(Q,X)**, because its lower bound **δ(Q,P)-δ (X,P)** is larger than **r**, so **X** surely cannot be in the query ball, so **X** is ignored

METRIC ACCESS METHODS:

-> **MAMs** – indexování pro efektivní hledání podobnosti v metrickém prostoru

- databáze částečně do ekvivalentních tříd
- po odstranění irelevantních výsledků – **sekvenčně**
- **pivotové tabulky**, stromy, hashované indexy, kombinace

9. Indexing metric similarity for efficient multimedia retrieval

--- *chceme indexovat metrickou podobnost, abychom zjednodušili a zpřesnili příjem multimédií*

- opět platí, že **efektivní = rychlé**

lower-bounding

-> místo „drahé“ vzdálenosti“ chci jen lower-bound

- mechanismus, jak **rychle odfiltrovat irelevantní** objekty z hledání
- **„metrické postuláty“** --- reflexivita (objekt je sám sobě identický), symetrie, troj. nerovnost
- boundování na:
  - -> objekty
  - -> regiony
  - -> kombinace „lower/upper bounds“

LAESA (TABULKA, K PIVOTŮ) --- nejsou MAM

= vybírám **k objektů jako pivoty**, sekvenčně procesuji vzdálenostní matici a filtruji

The task: check if **x** is inside query ball
- we know **δ(q,p)**
- we know **δ(x,p)**
- we do not know **δ(q,x)**
- we do not have to compute **δ(q,x)**, because its lower bound **|δ(q,p)-δ (p,x)|** is larger than **r**, so **x** surely cannot be in the query ball, so **x** is ignored

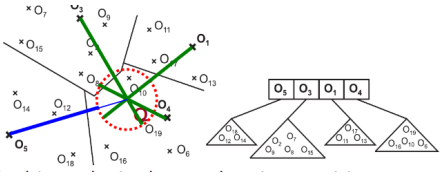
-- jednofázové = může sklouznout k naivním sekvenčnímu hledání

-> stále se mi zvětšuje počet pivotů

-- dvoufázové = query vektor určen query objektem, databáze seřazená, postupně projdi kandidáty **AESA (TABULKA, VŠECHNY PIVOTY)**

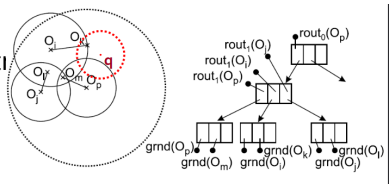
- použití **hierarchického MAM** --- rozdělení metrického prostoru na části, rekurzivně

---- používám **lokální pivoty** (založená na kontextu), ne globální



GNAT (n-ární STROM!)
-> hyper-prostorové rozdělení prostoru, range query procesování

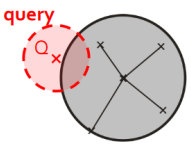
M-strom
-> podobné R-stromu s modifikací pro **metrické prostory** (míče, st



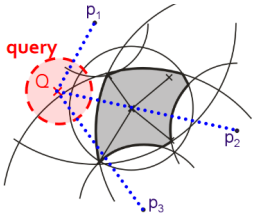
- použití **hashovacího MAM**
  -> každý objekt zahashován do „kyblíku“
  - lokálně-sensitivní hashovací funkce
  -> **D-index** (hash index) --- pro malé query radius vhodné, avšak volba parametrů je těžší

- použití **hybridního MAM**
  -> **kombinace** různých metod (např. pivotové tabulky s hierarchickými indexy)
  - PM-strom (M-strom s **p** globálními pivoty navíc, rychlejší filtering)

M-INDEX --- kombinace VŠECH principů (pivoty, hyperplochy, rozdělování, hashing)
-- **n** pivotů jako centra C
-- každý objekt **o** mapován (hashován) k indexu **i** nejbližšího pivotu
-- range query pak transformováno do setu intervalů, které chceme prohledat



M-tree region



PM-tree region

---- použijeme GNAT (multi-level M-index – rekurzivně rozděluje intervaly na ose)
--- zároveň si pamatujeme informace o dodatečném filtrování (M-tree)

VÝKON
- sekvenční sken jako referenční metoda
-> **DC** (výpočet vzdálenosti) = **počet vzdálenostních výpočtů** (drahá vzdálenost = náročné)
- **hierarchický MAM (s random access)** má smysl pro **hledání na SSD či RAM** (dříve i HDD)
- optimalizace MAM
  -> více sofistikovaný = větší overhead; **kNN processing** (DC super, ale hodně náročné časově)
  -- LAESA – nebude rychlejší s MAM než běžné sekvenční query

- **real-time cost** (wall-clock time)
  -> těžší naimplementovat, těžší vymyslet férové porovnání

10. Approximate similarity search

*úplná podobnost by byla velmi náročná, proto **částečná***
- používáme „divné **p**“ pro určení **struktury datasetu** (pokud je **p malé LOW INTRINSIC DIMENSIONALITY,** tak jde mezi objekty **pěkně rozlišovat TVOŘÍ SHLUKY**, pokud je **p velké**, tak se to dělá **těžko HIGH INSTRINSIC DIMENSIONALITY – NETVOŘÍ SHLUKY**)

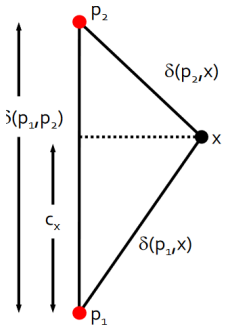
- „částečně/skoro“
  -> každý objekt bude alespoň **trochu relevantní**
  -> **pravděpodobnostní odhady** (relevance s nějakou pravděpodobnostní)
  - budeme indexovat (z **metrického do vektorového prostoru**)
  - před-procesování databáze (static i dynamické)
  - **mnoho mapovacích technik:**

-- založeno na **mapování do euklidovských vektorových prostorů**
-- pivotové tabulky, FastMap, MetricMap, SparseMap

- pivotové tabulky -> více dobrých pivotů zaručí **užší lower bound** (lepší odhady), nemusí fungovat (nedostatek dobrých pivotů)

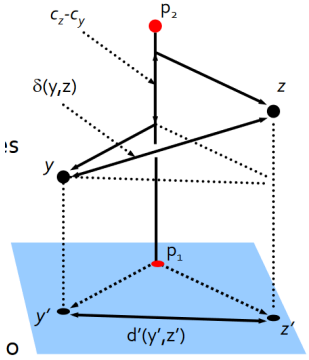
- FastMap -> dvojice pivotů (každá dvojice = jedna **pivot osa**) – Kosinův zákon, Pythagoras

kroky:

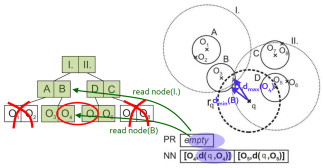


0: **k** je **číslo dimenzí cílového euklidovského prostoru**
1: **vyber první** (či další při opětovné iteraci) **nejlepší pivot dvojici**
--- vyber objekt **o náhodně**, vyber **p1** co nejdále od **o,** pak **p2** nejdále od **p1;** tím vznikne osa (další dimenze)
2: **Kosinova věta** pro pivoty **p1** a **p2**
- **x** je bod v databázi
-- **c**x je další souřadnice **x** v **cílovém prostoru**
3: **update** funkce, aby zahrnula již určené souřadnice objektů v databázi
-- spočítáme vzdálenost v ortogonální hyper-ploše přes Pythagora
--- následně návrat do **kroku 1**

$$d'(y',z') = \sqrt{\delta(y,z)^2 - (c_z - c_y)^2}$$



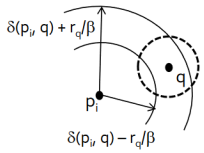
- **M-strom** -> metrické indexování („ball regions“)
-- kNN -> **prioritní fronta PR**, **NN array**
-- přesné kNN
-> do fronty se přidávají vrcholy, v array jsou kandidáti na nejbližší sousedy



-- **přibližné kNN**
-> pokud se NN array mění pomalu, eliminace
-> přibližně správné hledání – kandidáti budou blízko
-> dobré rozdělení
- PAC queries (**probably approximately correct**)

- nearest neighbors are searched, but guaranteeing only probability κ that the result objects will be at least (1+ε) nearest neighbors
- the terminating condition in the kNN algorithm
  - estimate the distance to k^{th} nearest neighbor (query radius r_q)
    - by testing $F(r_q)$, until $F(r_q)$ * size of the database = k
  - nodes from PR are filtered using $r_q' = r_q/(1+\varepsilon)$

- **přibližné hledání** v **pivot tabulkách** -> threshold, dělím



- **permutační indexy** -> s pivot tabulkou
-- místo geometrie a vzdáleností mám **permutace**
- kombinatorický přístup
- už v prvním kroku **sort**

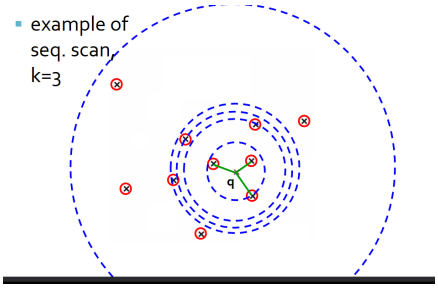
- Spearman Rho $S_\rho(\Pi_u, \Pi_q) = \sum_{1 \le i \le k} \left(\Pi_u^{-1}(i) - \Pi_q^{-1}(i)\right)^2$
- Spearman Footrule $F(\Pi_u, \Pi_q) = \sum_{1 \le i \le k} |\Pi_u^{-1}(i) - \Pi_q^{-1}(i)|$

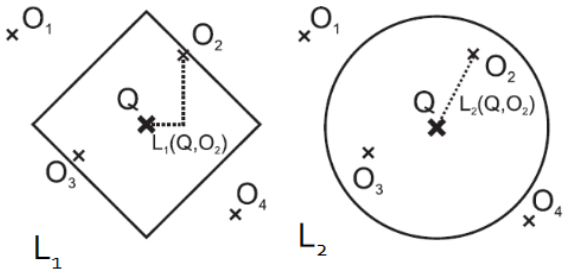
where $\Pi^{-1}{}_u(i)$ is the position of pivot p_i in the permutation Π_u

11. Similarity queries and multi-modal search

- **třízení dle podobnosti** where $\forall \mathbf{i} \in (1,|\mathbf{S}|)$: $\delta(\mathbf{q}, \text{SimOrder}(\mathbf{q})[i]) \leq \delta(\mathbf{q}, \text{SimOrder}(\mathbf{q})[i+1])$
- SimOrder: U -> S to the power of |S|
  --- v zásadě je SimOrder databáze seřazené sestupně dle vzdálenosti jejích elementů ke query q
- range query (radius **r** a range query tak vrací všechny **databázové deskriptory** pro vzdálenosti od **q,** které **nejsou větší než r**)

- knn Query





$$L_p(v_1, v_2) = (\sum_{i=1}^{D} |v_1[i] - v_2[i]|^p)^{\frac{1}{p}} \qquad (p \geq 1)$$

range vs. knn
- range, když uživatel **umí** specifikovat dotaz (sémantika)
- knn, když člověk **neví**

k reverse nejbližších sousedů (**krnn**)
-> zajímavé, u GPS používáno

databázový operátor – operace v databázi s výsledkem

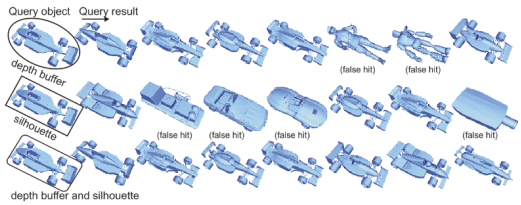
query – parametr, subset databáze je výsledkem
operátor – ne-parametrická operace, výsledek často velký

Similarity operators
similarity join -> deskriptory databáze A spojeny s deskriptory databáze B (či self-join)
k nejbližších párů – nejmenší vzdálenost z A a B

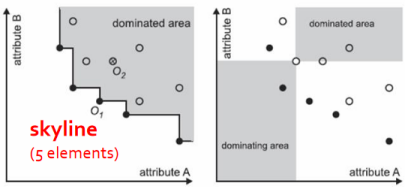
Multi-modal search
- **deskriptory více objektů** a queries z **různých modelů** a příkladů

a) Early
-> multimetrický přístup
(hloubkový buffer s deskriptorem, siluety mají deskriptory)

- $w_1 = 1, w_2 = 0$
- $w_1 = 0, w_2 = 1$
- $w_1 = 0.5, w_2 = 0.5$



b) Late
-> multi-example pro lepší hledání
- **skyline operátor** (Pareto set)
  -> databáze S jako seřazené domény (atributy)
  -> skyline = subset elementů S nedominovaných jinými elementy
  -> **použití třeba při filtraci hotelů**
  (cena + vzdálenost od letiště)
  - nemá ranking, mnohdy hodně rozsáhlé



------ zakomponování **multi-example similitarity query**
--- dynamické schéma

- **top-k operátor**
    -> „final ranking“
    - postupně je ohodnoceno různými způsoby, na konci spojím do „re-ranking“
- **query languages**
    -> rozšíření SQL --- databáze deskriptorů v relační databázi -> **RANGE, KNN**

```
SELECT Id FROM BioData WHERE
    range(JohnSmith.Fingerprint, BioData.FingerPrint, 0.01)

SELECT Id FROM DNAData WHERE
    kNN(MickeyMouse.DNA, DNAData.DNA, 1)

SELECT Mammal.Id, Insect.Id FROM Mammal
SIMILARITY JOIN Insect ON kNN(Mammal.DNA, Insect.DNA, 2)
```