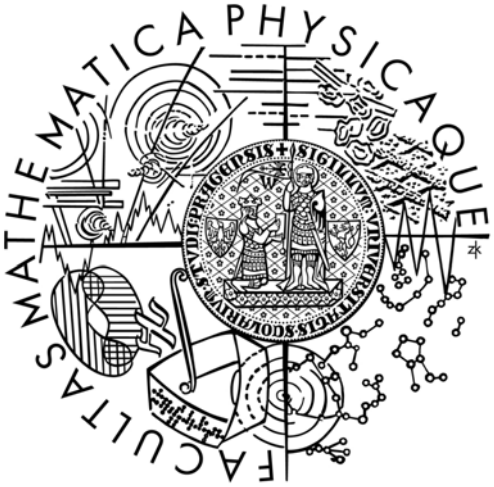
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Martin Košalko

Alternativní vyhledávač systému EGOTHOR

Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Michal Kopecký, Ph.D.
Studijní program: Informatika (softwarové systémy)

Za neoceniteľnú pomoc pri tvorbe tejto práce by som sa chcel poďakovať svojmu vedúcemu RNDr. Michalovi Kopeckému, Ph.D., ktorý svojimi cennými radami, návrhmi a pripomienkami významným spôsobom prispel k výslednej podobe textu a implementácie.

Moja vďaka patrí aj autorovi systému Egothor, RNDr. Leovi Galambošovi Ph.D., za uvedenie do problematiky vyhľadávača Egothor a následné konzultácie.

V neposlednom rade by som chcel poďakovať svojej priateľke Barbare Nádašiovej, vďaka ktorej nie je práca prehliadkou gramatických a štylistických chýb.

Vyhlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 21.4.2006 Martin Košalko

1. Obsah

Název práce: Alternativní vyhledávač systému EGOTHOR
Autor: Martin Košalko
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Michal Kopecký, Ph.D.
e-mail vedoucího: michal.kopecky@mff.cuni.cz
Abstrakt: V současnosti roste potřeba zpracování velkého množství informací, čemuž se přizpůsobují i aktuální trendy v informatice. Cílem práce je teoretický rozbor a implementace nástroje určeného na zpracování a prohledávání velkých kolekcí nestrukturovaných textů - dokumentografického informačního systému (DIS). Práce navazuje na existující systém Egothor, který je implementací boolského modelu. Kvůli možnosti porovnávat výsledky různých modelů DIS, byl implementovaný systém založený na „konkurenčním“ vektorovém modelu. V rámci diplomové práce byl vytvořený i modulární systém umožňující porovnávat a zpracovávat výsledky nezávislých DIS do jednoho uceleného přehledu. Takový systém se nazývá systém optimálního vyhledávání. Uživateli umožňuje najít takovou kombinaci vyhledávacích algoritmů, která je pro něj subjektivně optimální - maximalizují se subjektivně kvalitativní parametry systému, přesnost a úplnost. Systém optimálního vyhledávání byl otestovaný v konfiguraci se dvěma podřízenými DIS, co potvrdilo jeho výhody.
Klíčová slova: vektorový dokumentografický informační systém optimální vyhledávaní

Title: The alternative information retrieval system to EGOTHOR system
Author: Martin Košalko
Department: Department of software engineering
Supervisor: RNDr. Michal Kopecký, Ph.D.
Supervisor's e-mail address: michal.kopecky@mff.cuni.cz
Abstract: Nowadays, it is necessary to process huge amount of information what is reflected by actual trends in informatics. The objective of the thesis is to give the theoretical analysis and implementation of processing and searching tool which allows the user to go through a huge number of unstructured document collections. Such system is called the information retrieval system. This work is an alternative to already existed system Egothor, which is the implementation of boolean model. Because of the possibility of comparing the results from retrieval system models, the implemented system is established on a competitive vector model. In this work is also created one modular system which allows comparing and processing the results from any independent information retrieval systems to one integrated review. This system is called the optimal retrieval system. The precision and recall – the subjective qualitative parameters of system are growing up. The optimal retrieval system was tested in configuration with two underlying systems what practically approved it’s advantages.
Keywords: vector information retrieval system optimal search

2. Úvod do problematiky a vysvetlenie pojmov

„V súčasnej dobe neustále rastie potreba spracovania veľkého množstva informácií (novinových článkov, odbornej literatúry, korešpondencie, agentúrnych správ, zákonov, príspevkov do konferencií ...), ktoré sú prístupné prostredníctvom počítača. Pre používateľa je väčšinou problém, že nevie, kde sa nachádzajú pre neho zaujímavé informácie, prípadne vôbec nevie, či sa žiadané informácie v dostupnej kolekcii textov (dokumentov) vôbec nachádzajú.“ [2D] Práve potreba spracovania veľkého množstva informácií viedla k vzniku samostatného vedného oboru „*vyhľadávanie (získavanie) informácií“* – pravda, výstižnejší je anglický termín „*information retrieval (IR)“*. Softwarové systémy určené na spracovávanie, ukladanie a vyberanie informácií sa nazývajú *dokumentografické informačné systémy (DIS)*. Veľmi výstižnú definíciu DIS uvádza autor v [6D] (paradoxne, definícia začína tým, čo DIS nie je): „Dokumentografický informačný systém neinformuje (vo význame podania znalostí) používateľa priamo o predmete jeho záujmu špecifikovaného dotazom, ale iba ponúka informáciu o existencii, prípadne neexistencii dokumentov (textov) vzťahujúcich sa k dotazu používateľa.“ Trochu nadnesene možno povedať, že DIS ponúka len informácie o infomáciách o predmete záujmu používateľa. Práve toto je hlavný fakt, ktorý oddeľuje vedný obor *získavanie informácií* (*information retrieval*) od oboru *získavanie dát* (*data retrieval - DR*).

Predstaviteľom oboru *získavanie dát* je *faktografický informačný systém (FIS)*. Rozdielom DIS a FIS je objekt ich spracovania. Zatiaľ čo FIS pracujú nad štruktúrovanými dátami (podnikové dáta – objednávky, zamestananci, úlohy...), DIS pracujú s dokumentmi a textami, ktoré nie sú štruktúrované. Odlišný je zvyčajne aj spôsob uloženia dát. FIS používa pevnú dátovú štruktúru (napr. relačnú databázu), čo dátam dáva presnú sémantiku, ktorú informácie v DIS nemajú. Ďalšie podrobnosti a rozdiely medzi DIS a FIS, resp. IR a DR, možno nájsť v domácej aj zahraničnej literatúre [2D], [5D].

Dotazovanie na DIS

Úlohou každého DIS je riešiť vyhľadávací problém, čo znamená nájsť k používateľskému dotazu *relevantné dokumenty*. Teda dokumenty, ktoré istým spôsobom spĺňajú požiadavky kladené používateľom. Požiadavky používateľ formuluje prostredníctvom dotazu. Dokumenty vrátené vo výsledku vyhľadávania sa nazývajú *hity* (z anglického hit = zásah). Je to množina dokumentov, o ktorých DIS predpokladá, že by mohli byť pre používateľa relevantné. Relevancia dokumentu je subjektívny pojem, pretože je viazaný na používateľa. Keďže DIS pracuje s neštruktúrovanými dokumentmi a textami, dotazovanie na DIS je intuitívne, vágne a teda nepresné. Prejavuje sa to, ak chce používateľ získať informácie o nejakej problematike. V žiadnom prípade nemá zaručené, že slová, ktoré použije ako dotaz, použili aj autori dokumentov s relevantnou problematikou.

Napriek tomu je možné dotaz na DIS definovať exaktne ako vyčísliteľnú funkciu Q nad stavom kolekcie textov (databázy) C, ktorej hodnotou je podmnožina dokumentov z C. Prakticky býva dotazom nejaká množina slov, ktoré môžu byť navyše ohodnotené váhou a môžu medzi nimi existovať nejaké vzťahy. Napríklad $([0{,}2 : T1]\ OR\ [0{,}8 : T2])\ AND\ [0{,}95 : T3]\ AND\ [0{,}1 : T4]$, kde *T1, T2, T3* a *T4* sú slová, ktoré by mali obsahovať dokumenty z výsledku. Číselné hodnoty v dotaze

predstavujú mieru dôležitosti slova pre používateľa a logické spojky vyjadrujú vzťahy medzi slovami. Formát dotazu však silno závisí na konkrétnom modeli DIS.

Modely DIS

Aj keď DIS pracuje nad neštruktúrovanými dokumentami, je potrebné s nimi narábať na istej úrovni abstrakcie – vytvárať štruktúry, ktoré ich modelujú. *Model DIS* je podľa [2D] definovaný ako súbor pojmov a nástrojov na:

- reprezentáciu dokumentu – teda formálny popis informácie obsiahnutej v dokumente
- reprezentáciu dotazu – umožňuje formálne špecifikovať požiadavku na informácie
- reprezentáciu pravidiel a procedúr, ktoré umožňujú určiť zhodu medzi dotazom a dokumentmi v kolekcii textov.

Model DIS je teoretickým základom pre dátové štruktúry, ktoré implementujú jednotlivé entity a interakcie medzi nimi. Prakticky najpoužívanejšími klasickými modelmi sú v súčasnosti *(rozšírený) boolský model*, *vektorový model* a *pravdepodobnostný model*.

V prípade *boolského* a *vektorového modelu* sa využíva princíp vyhľadávania podľa umiestnenia – to znamená, že entity (reprezentácie) dokumentov sú pevne (staticky) uložené na vhodných označkovaných miestach a dynamicky sa organizujú len odkazy na dokumenty.

- Pre *boolský model* sú odkazmi dvojice slovo-dokument. Slovo charakterizujúce dokument „vie“ (drží si ukazovateľ), kde je uložený dokument. Takéto slová sa tiež nazývajú *termy* = kľúče, deskriptory. Typickou štruktúrou princípu vyhľadávania podľa umiestnenia je *invertovaný zoznam*. Ten obsahuje ku každému termu zoznam odkazov na dokumenty, pre ktoré je term deskriptorom. Abstraktnou reprezentáciou dokumentu je množina relevantných slov – termov. Skutočne ide o abstraktnú reprezentáciu, pretože vo vnútornej štruktúre, ktorou je zvyčajne invertovaný zoznam, nie je dokument takto reprezentovaný. Dotazovanie nad týmto modelom prebieha pomocou logickej formuly, ktorej atómy sú termy. Vnútorne to znamená aplikáciu množinových operácií na množiny dokumentov prislúchajúcich k jednotlivým termom z dotazu (OR ~ zjednotenie, AND ~ prienik, NOT ~ množinový rozdiel).
- V prípade *vektorového modelu* je dokument reprezentovaný ohodnoteným vektorom termov. Rovnako je reprezentovaný aj dotaz. Číslo na pozícii i vo vektore dokumentu udáva mieru dôležitosti i-tého termu pre daný dokument. To znamená, že ak je vo vektore dokumentu nulová hodnota, príslušný term je pre popis daného dokumentu nevýznamný. Dotazovanie potom spočíva v porovnávaní vektora dotazu s vektormi dokumentov z kolekcie textov. Prirodzene, podobnosť vektora dotazu a dokumentu je priamo úmerná ich vhodnosti, respektíve relevancii. Typickou štruktúrou modelu je riedka matica, kde na pozícii $[i, j]$ je miera dôležitosť i-tého termu v j-tom dokumente. „Odkazmi“ na dokumenty sú potom druhé zložky súradníc $[i, j]$ príslušného prvku matice.

Pravdepodobnostný model je založený na princípe pravdepodobnostného usporiadania. Dokumenty sú usporiadané podľa predpokladanej relevancie. Tie s vyššou pravdepodobnosťou relevancie sú uprednostňované pred ostatnými. V praxi to znamená, že sa využíva spätná väzba. Po zobrazení výsledku dotazu používateľ označí niektoré dokumenty za relevantné a úlohou DISu je nájsť množinu

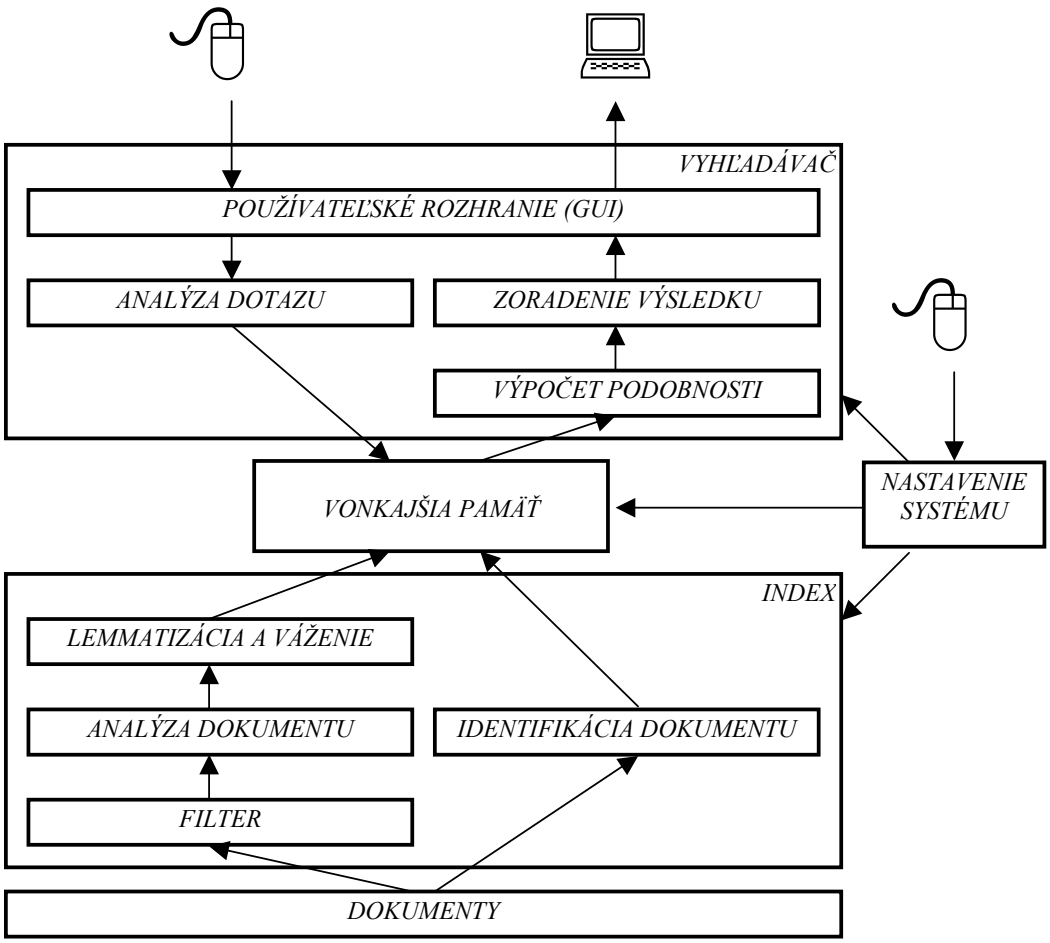
relevantných dokumentov, pričom zohľadňuje koeficienty podobnosti medzi dokumentmi.

Vzhľadom na implementáciu DIS vo vektorovom modeli a existujúci DIS Egothor v rozšírenom boolskom modeli je v ďalšom texte detailnejšie rozobratý len *rozšírený boolský* a *vektorový systém*.

Ďalšie pojmy

- Modul obsahujúci samotnú kolekciu textov ako dátový zdroj spoločne s nástrojmi na jej údržbu sa nazýva *index* DIS.
- Miera dôležitosti termu v dokumente sa nazýva *váha termu*. Ide o číselnú hodnotu, obyčajne z intervalu $\langle 0,1 \rangle$, ktorá je odvodená z *termovej frekvencie* termu a *inverznej dokumentovej frekvencie* termu. *Termová frekvencia* termu je veličina udávajúca počet výskytov termu v dokumente v pomere k celkovému počtu slov, a preto s rastúcim počtom výskytov termu rastie. Obdobne, *dokumentová frekvencia* udáva počet dokumentov v ktorých sa term vyskytuje vzhľadom na celkový počet dokumentov kolekcie. Pre výpočet *váhy termu* sa však používa *inverzná dokumentová frekvencia*, ktorá klesá s rastúcim počtom dokumentov, kde sa term vyskytuje. Používané vzorce na výpočet váhy termu sú uvedené v kapitole o indexových súboroch 4.3.1. Nateraz nie sú dôležité.

2.1 Architektúra DIS



Každý väčší softwarový systém možno rozdeliť na niekoľko menších častí (modulov) s presne definovanou funkcionalitou. Kladie to síce zvýšené nároky na exaktnú definíciu komunikačných rozhraní a rozdelenie funkcionality systému na jednotlivé moduly, no umožňuje separátny vývoj jednotlivých modulov a zameranie sa na optimalizáciu výkonnosti každého z nich. Na výkon celého systému má samozrejme vplyv aj vzájomná kooperácia činností jednotlivých modulov. Aj DIS majú svoju typickú architektúru. Tá sa síce model od modelu líši, no možno nájsť isté spoločné črty všetkých modelov DIS.

Schéma názorne ukazuje, že DIS má dva základné abstraktné moduly: INDEX a VYHĽADÁVAČ a jeden vedľajší modul slúžiaci na nastavenie vlastností a chovania sa celého systému. Hlavné moduly INDEX a VYHĽADÁVAČ operujú nad spoločnými dátami vo vonkajšej pamäti. Moduly odpovedajú dvom základným funkciám DIS, ktorými sú riešenie vyhľadávacieho problému a indexácia dokumentov. Pri indexácii sa do vonkajšej pamäte zapisuje, pri vyhľadávaní naopak číta. V praxi je častejšie, že VYHĽADÁVAČ nepristupuje k dátam na vonkajšej pamäti priamo, ako je to naznačené v schéme, ale dáta sú súčasťou INDEXu a VYHĽADÁVAČ k ním pristupuje prostredníctvom metód modulu INDEX.

2.1.1 Indexovanie

Indexovanie dokumentov je ich transformáciou do štruktúry vhodnej na vyhľadávanie. Do procesu idexácie vstupujú dokumenty rôznych druhov a formátov (HTML, PDF, DOC), preto ich je potrebné upraviť do podoby vhodnej na indexáciu. Toto je úloha modulu FILTER. Filtrovaním sa z formátovaných dokumentov a textov stane postupnosť neformátovaných slov, značiek, číslic a interpunkčných znamienok… Tento „čistý“ text je predaný na ANALÝZU DOKUMENTU, kde sa postupnosti znakov rozdelia na „slová“ a „neslová“ (neslová sú všetky interpunkčné znamienka, zátvorky…), pričom do ďalšieho spracovania postupujú len slová. Vykonáva sa takzvaná lexikálna analýza textu.

Pre indexovanie a vyhľadávanie nie je vhodné používať morfologické tvary slov, ktorých je vďaka skloňovaniu a časovaniu relatívne mnoho, ale ich slovné základy. Má to opodstatnenie v tom, že používateľ nemusí v dotaze použiť ten istý tvar slova, ako použil autor v texte, no DIS aj dokument s iným morfologickým tvarom označí ako vyhovujúci dotazu. Navyše sa tým významne šetrí miesto v súboroch na vonkajšej pamäti. Prevod na základný tvar slova sa nazýva LEMMATIZÁCIA. Po lemmatizácii sa zo slov stávajú termy používané na identifikáciu dokumentov. V tom istom kroku sa vykonáva aj váženie termov a tvorba slovníka nevýznamových slov, kam patria všetky termy, ktoré sa nepoužívajú na identifikáciu dokumentov. Ide zvyčajne o slová s gramatickým významom, no bez obsahovej identity.

V závere procesu indexácie sú všetky významové termy dokumentu uložené do vonkajšej pamäte spolu s jeho jednoznačným identifikátorom (výsledok IDENTIFIKÁCIE DOKUMENTU). V prípade dokumentov na internete je jednoznačným identifikátorom URL.

Schéma ukazuje len základnú verziu indexácie, ktorá bližšie nešpecifikuje spôsob, akým sa dokumenty dostanú do procesu indexácie. V praxi to silno závisí na type dokumentov a ich zdroji. V prípade internetu ako dátového zdroja zvyčajne existuje špeciálny modul ROBOT, ktorý v pravidelných intervaloch prechádza definované

domény a zisťuje, či na nich nenastala nejaká zmena, či nejaké dokumenty (HTML stránky) nepribudli, prípadne, či nejaké neboli odstránené. V prípade zmeny je dokument stiahnutý na lokálny počítač (server) a predaný modulu INDEX na spracovanie.

2.1.2 Vyhľadávanie

Vyhľadávanie začína zadaním dotazu cez POUŽÍVATEĽSKÉ ROZHRANIE, ktoré môže mať rôznu podobu - grafickú, textovú, súborom... Dotaz je spracovaný ANALYZÁTOROM DOTAZU, kde prebieha podobný proces ako pri spracovaní dokumentu. Časti dotazu sa rozdelia na „slová“ a „neslová“ a vykoná sa lemmatizácia. Výsledkom je štruktúrovaný dotaz, ktorý sa už dá porovnávať s dokumentmi na VONKAJŠEJ PAMÄTI. Dokumenty sa ohodnotia (*ranking*) podľa podobnosti s dotazom (modul VÝPOČET PODOBNOSTI) a usporiadajú vzostupne (modul ZORADENIE VÝSLEDKU). Výsledok je poskytnutý používateľovi používateľským rozhraním.

2.2 Meranie kvality a výkonu DIS

Výkon dokumentografických informačných systémov sa zvyčajne sleduje v počte zaindexovaných/prehľadaných dokumentov za jednotku času. Nepochybne je to dôležitý ukazovateľ, no zaujímavejšie je sledovať ako sa mení výkon pri paralelnom prístupe viacerých používateľov, čo je v praxi bežnejšia situácia.

Pre používateľa je okrem rýchlosti odozvy dôležitá aj kvalita výstupu systému. Tú určujú dve veličiny: presnosť P (z anglického precision) a úplnosť R (z anglického recall). Nech:

- n_v je počet dokumentov vrátených systémom – sú do dokumenty, o ktorých si systém „myslí“, že sú relevantné – sú to tzv. dotazu vyhovujúce dokumenty
- n_{vr} je počet vrátených relevantných dokumentov – sú to dokumenty, ktoré považuje za relevantné používateľ systému (vybrané len z množiny vrátených dokumentov)
- n_r je celkový počet relevantných dokumentov v systéme

Potom presnosť systému $P = n_{vr} / n_v$ určuje pravdepodobnosť, že dokument zaradený do odpovede systému je relevantný a úplnosť $R = n_{vr} / n_r$ určuje pravdepodobnosť, že skutočne relevantný dokument je zaradený v odpovedi. Obe veličiny sú subjektívne, pretože ten istý vrátený výstup môže uspokojovať dvoch rôznych používateľov rôznou mierou. Medzi veličinami presnosť a úplnosť existuje nepriama úmernosť: $P * R = konš \tan ta < 1$. Viac informácií možno nájsť napríklad v [1D], [2D].

2.3 Zoznam použitých značiek a skratiek

DIS – dokumentografický informačný systém
$\boldsymbol{DF_j}$ – frekvencia *j*-tého termu v kolekcii dokumentov (dokumentová frekvencia)
$\boldsymbol{IDF_j}$ – inverzná frekvencia *j*-tého termu v kolekcii dokumentov
TF – termová frekvencia (frekvencia termu v jednom dokumente)
$\boldsymbol{NTF_{i,j}}$ – normalizovaná termová frekvencia *j*-tého termu v *i*-tom dokumente
$\boldsymbol{w_{i,j}}$ – váha *j*-tého termu v *i*-tom dokumente

3. Cieľ diplomovej práce

3.1 Rozbor východiskového stavu

Ako naznačuje názov diplomovej práce „Alternativní vyhledávač systému EGOTHOR“, cieľom diplomovej práce je teoretický rozbor a implementácia alternatívneho DIS k systému Egothor. Bolo by však samoúčelné a pomerne zbytočné vytvoriť ďalší DIS bez toho, aby bolo možné výsledky oboch systémov porovnávať a vzájomne dopĺňať. Preto si práca okrem vytvorenia alternatívneho vyhľadávača kladie za cieľ aj vybudovanie systému, ktorý by umožňoval zadať jeden používateľský dotaz viacerým „podradeným“ DIS a následne dokázal spracovať výsledky vyhľadávania jednotlivých DIS do uceleného prehľadu. Výsledok odvodený z výsledkov viacerých DIS by mal byť presnejší a úplnejší ako čiastkové výsledky jednotlivých DIS.

Systém Egothor je DIS založený na rozšírenom boolskom modeli. Teoretický podklad rozšírených boolských systémov tvoria *fuzzy množiny*. Teóriu fuzzy množín je možné chápať ako „formalizáciu vágnosti“. Ak nie je možné určiť hranicu triedy určenej vágnym pojmom (napríklad vysoký, chudobný – všetko sú to relatívne pojmy), potom mieru náležania nejakého objektu do triedy je možné vyjadriť číselnou hodnotou z nejakého intervalu, napríklad $\langle 0,1 \rangle$, kde 0 znamená, že objekt do triedy nepatrí a 1, že tam patrí. Prirodzene, nižšia hodnota príslušnosti objektu znamená, že objekt do vágne vymedzenej triedy patrí menej ako vyššia miera príslušnosti objektu k triede. Viac o teórii fuzzy množín možno nájsť v prácach [6D], [7D].

V kontexte DIS rozšíreného boolského modelu sú to práve termy, ktoré „vágne“ vymedzujú triedy. Dokumenty sú objekty, ktoré do termami definovaných tried náležia. To znamená, bez ohľadu na použitý submodel (MMM, Paicov, s merítkom P, viac o submodeloch v práci [2D]), je každý dokument charakterizovaný ohodnoteným vektorom termov, kde ohodnotenie vyjadruje mieru akou dokument patrí do triedy vymedzenej príslušným termom. V praxi sú koeficienty náležania odvodzované zo štatistických údajov výskytu termu v dokumentoch – teda termovej frekvencie a dokumentovej frevencie. Presne takto je to aj v systéme Egothor.

Pre ďalšie smerovanie práce je to kľúčový poznatok. Práve existencia údajov termová a dokumentová frekvencia od seba odlišuje indexy vektorového a boolského systému. Pre rýdzo boolský model sú zbytočné, zatiaľ čo pre vektorový model DIS nenahraditeľné. Všetky informácie potrebné na výpočet novovzniknutého DIS založeného na vektorovom modeli sú uložené v indexových súboroch systému Egothor. Jedným z cieľov práce je preto v čo najväčšej miere využiť už existujúci index. Pravdou je, že organizácia a štruktúra indexových súborov systému Egothor úplne nevyhovuje spôsobu práce DIS vo vektorovom modeli, a preto je potrebné indexové súbory podrobne analyzovať a vykonať niekoľko zmien. Podrobne je problematika indexových súborov rozpracovaná v kapitole 4.

Vo výsledku budú oba systémy pracovať nad spoločným dátovým zdrojom, čo má hneď tri dôsledky:

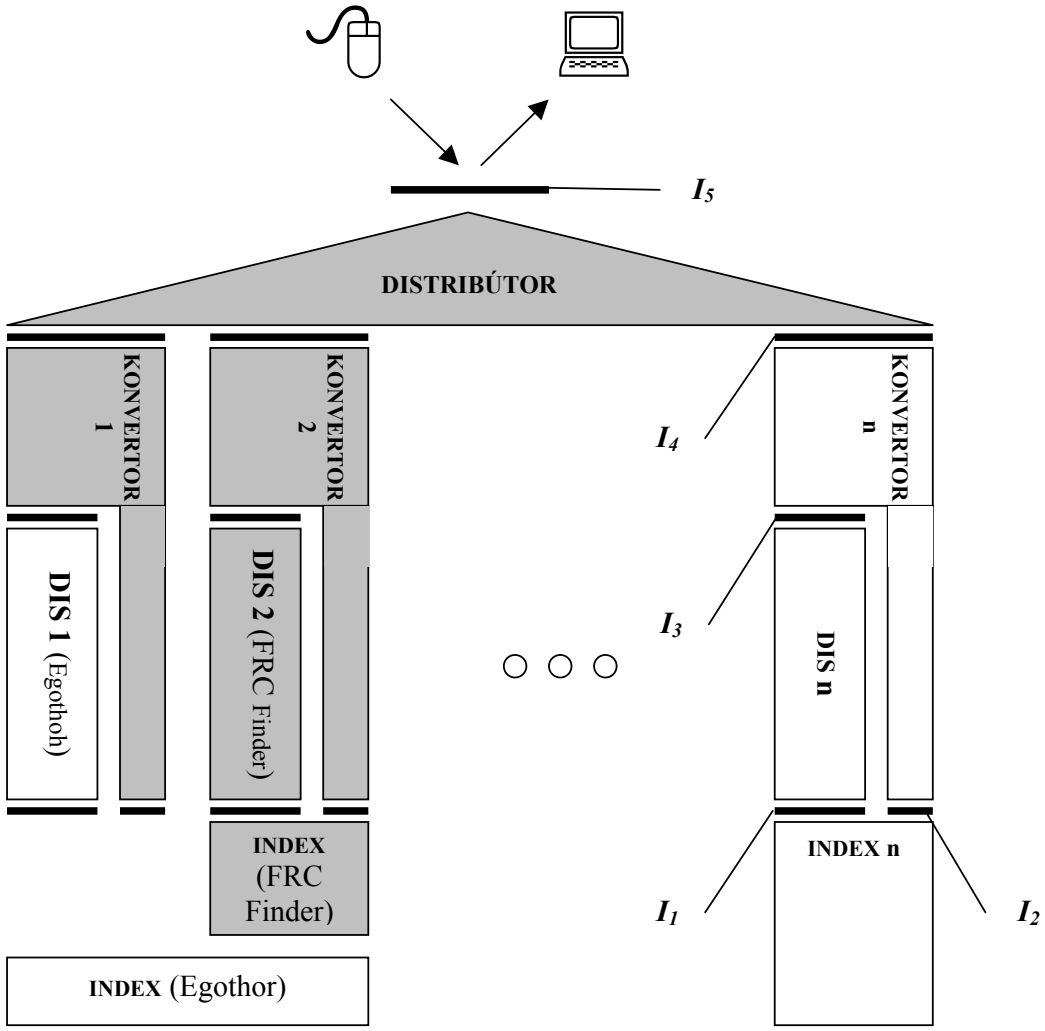
- v porovnaní správania sa oboch systémov hrá dôležitú úlohu len spôsob (algoritmus) výpočtu podobnosti dokumentu a dotazu, čo umožní presnejšie charakterizovať spoločné črty a rozdiely v správaní sa jednotlivých modelov
- nie je potrebné implementovať indexačný algoritmus alternatívneho vyhľadávača, ktorý je nahradený algoritmom na vytvorenie „nadstavby“ existujúceho indexu systému Egothor
- úspora miesta na disku

3.2 Charakterizácia cieľov

Z rozboru východiskovej situácie už vyplývajú omnoho konkrétnejšie ciele, ktoré možno charakterizovať nasledovne: cieľom tejto diplomovej práce je vybudovanie modulárneho systému, ktorý by umožňoval spracovať výsledky vyhľadávania poskytované rôznymi typmi dokumentografických informačných systémov, tieto výsledky porovnávať, interpretovať a následne poskytovať používateľovi. Z hlavného cieľa vyplývajú tri čiastkové ciele:

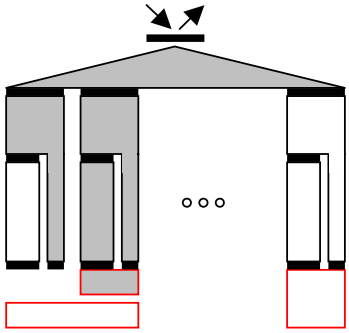
- **Všeobecný index DIS** - teoretická analýza indexových súborov DIS a špecifikácia rozhrania v jazyku Java pre prácu s takýmto všeobecným indexom. Cieľ pokrýva aj teoretický rozbor potrieb jednotlivých modelov DIS so zameraním sa na vektorový a boolský model. Ďalej si práca kladie za cieľ návrh organizácie a štruktúry súborov indexu pre vektorový model DIS.
- **DIS vo vektorovom modeli** – vytvorenie DIS vo vektorovom modeli (FRC Finder). Táto diplomová práca sčasti naväzuje na existujúci DIS v rozšírenom boolskom modeli (Egothor) a preto novovytvorený DIS vo vektorovom modeli využíva už existujúce indexové súbory. Index systému Egothor je však rozšírený o súbory umožňujúce rýchlejší prístup k informáciám, ktoré sú dôležité práve pre vektorový model DIS.
- **Moduly KONVERTOR a DISTRIBÚTOR** – teoretický rozbor a vytvorenie modulov KONVERTOR a DISTRIBÚTOR. Modul DISTRIBÚTOR zabezpečuje prijatie používateľského dotazu v rôznych formátoch (vektor termov, logická formula termov, text, odkaz na zaindexovaný dokument ...), jeho prvotné naformátovanie a distribúciu jednotlivým DIS. Najdôležitejšou funkciou modulu DISTRIBÚTOR je však porovnanie a vyhodnotenie výsledkov vyhľadávania jednotlivých DIS. Moduly KONVERTOR slúžia na konverziu dotazu do podoby prijateľnej príslušným DIS. Prirodzenou súčasťou práce je aj špecifikácia rozhraní medzi jednotlivými modulmi.

Nasledujúci obrázok jednoducho a schematicky znázorňuje architektúru celého systému. Moduly, ktoré sú vyznačené sivou farbou sú analyzované a tvorené v rámci tejto diplomovej práce.


I5
DISTRIBÚTOR
KONVERTOR 1
KONVERTOR 2
KONVERTOR n
I4
I3
DIS 1 (Egothoh)
DIS 2 (FRC Finder)
DIS n
INDEX (FRC Finder)
INDEX n
I1
I2
INDEX (Egothor)

4. Index DIS

4.1 Motivácia



Každý dokumentografický informačný systém si za svoj prvoradý cieľ kladie rýchle a čo najpresnejšie vyhľadávanie vo veľkých kolekciách dokumentov a následné sprístupnenie sekundárnych informácií o nájdených dokumentoch zahŕňajúce autora, umiestnenie dokumentu, rok vydania. Nájdené dokumenty si potom môže používateľ vyžiadať systémom dodania dokumentov a následne prečítať žiadané informácie. Vzhľadom na potenciálne veľké množstvo dokumentov uložených v databáze, by ich prechádzanie pomocou fulltextového prehľadávania a porovnávanie s dotazom bolo neúnosne časovo náročné. Preto sa dokumenty indexujú – vytvára sa štruktúrovaný model dokumentu vhodný na porovnávanie s dotazom.

Na index dokumentografického informačného systému sú kladené vysoké požiadavky, pretože od rýchlosti odozvy indexu na dotazy závisí rýchlosť odozvy celého systému. Tento fakt je daný tým, že indexované informácie o dokumentoch sa v dôsledku svojej veľkosti do operačnej pamäte všeobecne vojsť nemôžu a musia byť preto uložené na pevnom disku počítača. Ako je všeobecne známe, tak prístup na pevný disk je rádovo pomalší ako prístup do operačnej pamäte počítača. Časovo najkritickejšou operáciou je teda získanie dát uložených v indexe systému.

4.2 Očakávané správanie indexu DIS

Z istého nadhľadu je možné povedať, že vstupom (dotazom) každého dokumentografického informačného systému je množina termov. Vo vektorovom modeli DIS je to vektor ohodnotených termov, prípadne odkaz na dokument, ktorý sa zaindexuje (ak ešte nie je) a získa sa z neho zasa len vektor ohodnotených termov. V boolskom modeli DIS je dotazom logická formula, kde termy vystupujú v roli atómov, prípadne ohodnotená logická formula v rozšírenom boolskom modeli, čo je v svojej podstate zasa množina termov. Z uvedeného vyplýva, že primárne sa do indexu bude pristupovať práve cez termy získané z dotazu používateľa.

Určite nie je efektívne, aby sa zadaný dotaz porovnával so všetkými zaindexovanými dokumentmi v databáze pomocou sekvenčného porovnávania, ale iba s dokumentmi, ktoré sú potenciálne vyhovujúce dotazu. Za dokument, ktorý je potenciálne vyhovujúci vzhľadom k dotazu sa považuje taký dokument, kde váha aspoň jedného termu z dotazu je nenulová. Táto definícia „zdegenerovane“ platí aj pre boolský model DIS, kde buď term dokument popisuje, má teda jednotkovú váhu, alebo nepopisuje, má nulovú váhu. Index by teda mal byť schopný k zadanému termu, prípadne skupine termov, vrátiť množinu potenciálne vyhovujúcich dokumentov v čo najkratšom možnom čase. K týmto dokumentom by mal byť schopný poskytnúť presnejšie informácie (váha a normalizovaná váha všetkých termov dokumentu, výskyty daného termu v dokumente), podľa ktorých sa v zvolenom modeli spočíta celková podobnosť (vhodnosť) dokumentu vzhľadom k dotazu. Na záver, keď je už

jasné, ktoré dokumenty sa poskytnú používateľovi ako výsledok, sa od indexu očakáva poskytnutie metadát týchto dokumentov.

4.2.1 Index vektorového DIS – problematické miesta

Pri podrobnejšom skúmaní práce indexu pri vyhľadávaní v kolekcii dokumentov je možné naraziť na slabé miesta, ktoré sú špecifické práve pre vektorový model DIS, no sčasti sa týkajú aj rozšíreného boolského modelu. Tieto miesta súvisia s tým, že vektorový model DIS môže počítať podobnosť dotazu a dokumentu pomocou rôznych podobnostných funkcií. Najpoužívanejšími a najštudovanejšími sú skalárny súčin, Diceov koeficient, Jaccardova miera, kosínusová miera. Ďalšie podobnostné funkcie možno nájsť napr. v [1D].

- skalárny súčin: $sim(\vec{q},\vec{d}_i) = \sum_{j=1}^{m} q_j \cdot w_{i,j}$
- Diceov koeficient: $sim(\vec{q},\vec{d}_i) = \frac{2 \cdot \sum_{j=1}^{m} q_j \cdot w_{i,j}}{\sum_{j=1}^{m} q_j + \sum_{j=1}^{m} w_{i,j}}$
- Jaccardova miera: $sim(\vec{q},\vec{d}_i) = \frac{\sum_{j=1}^{m} q_j \cdot w_{i,j}}{\sum_{j=1}^{m} q_j + \sum_{j=1}^{m} w_{i,j} - \sum_{j=1}^{m} q_j \cdot w_{i,j}}$
- kosínusová miera: $sim(\vec{q},\vec{d}_i) = \frac{\sum_{j=1}^{m} q_j \cdot w_{i,j}}{\sqrt{\sum_{j=1}^{m} q_j^2} \cdot \sqrt{\sum_{j=1}^{m} w_{i,j}^2}}$

Výraz $sim(\vec{q},\vec{d}_i)$ vyjadruje podobnosť vektora dotazu $\vec{q} =< q_1, q_2, ..., q_m > \in < -1,1 >^m$ a vektora *i*-tého dokumentu $\vec{w}_i =< w_{i,1}, w_{i,2}, ..., w_{i,m} > \in < 0,1 >^m$.

Normalizácia váh termov

Niektoré podobnostné funkcie, napríklad skalárny súčin, vyžadujú, aby váha termu poskytnutá indexom bola normalizovaná, prípadne vyžadujú také informácie, aby mohla byť normalizácia váh vykonaná pri samotnom výpočte. Dôvodom je to, že veľkosť vektoru váh termov v dokumente vplýva na výslednú podobnosť dotazu a dokumentu, pričom „dlhšie“ dokumenty (dokumenty, ktorých vektor váh má väčšiu dĺžku) sú zvýhodňované. Normalizácia, teda prevod vektoru váh na jednotkovú dĺžku, sa musí zabezpečiť buď pri vytváraní indexu a index ju bude priamo poskytovať, alebo počas vyhľadávania. Prvá možnosť má zásadnú nevýhodu v nutnosti dávkového prepočítavania pri pridávaní dokumentov do indexu, keďže výsledná váha termu (*w*) závisí od termovej frekvencie termu v dokumente (*TF* prípadne *NTF*), od celkového počtu dokumentov (*m*) a aj od počtu dokumentov kolekcie obsahujúcich daný term (viď výpočet inverznej dokumentovej frekvencie *IDF*).

$w = NTF \cdot IDF$, kde $IDF = \log(m / DF) + 1$

Druhá možnosť, kedy sa vektor váh termov normalizuje počas vyhľadávania, spomaľuje odozvu dokumentografického informačného systému.

Potreba dodania váh všetkých termov dokumentu
Aj toto slabé miesto indexu súvisí s výberom podobnostnej funkcie. Niektoré funkcie totiž vyžadujú pre výpočet podobnosti dotazu a dokumentu poznať celý vektor váh termov v dokumente. Ide napríklad o kosínusovú mieru alebo Diceov koeficient. Vektorový model DIS teda vyžaduje rýchly prístup k dátam nielen smerom od termov k dokumentom, ale aj naopak, kedy po zadaní dokumentu potrebuje získať všetky jeho relevantné termy. Pre objektivitu je nutné poznamenať, že v niektorých prípadoch je pre potreby podobnostných funkcií, namiesto poskytnutia celého vektoru váh termov dokumentu, postačujúce len dodanie veľkosti vektora dokumentu.

4.2.2 Špecializácia indexu

Vo všeobecnosti je možné povedať, že index boolského modelu DIS musí poskytovať a spravovať iné informácie ako index vektorového modelu DIS. Rozdiel je napríklad v termových frekvenciách, ktoré sú potrebné pre vektorový, no nepotrebné pre boolský model DIS. Opačne je to s pozíciami výskytov termov, ktoré sú využívané v boolskom modeli aj pri dotazovaní, zatiaľ čo vo vektorovom len na generovanie kontextovej ukážky. Z toho vyplýva aj fakt, že je efektívne tvoriť index danému modelu DIS „na mieru“. S užšou špecializáciou sú totiž spojené optimalizácie typické pre daný model. Tým sa zvyšuje efektívnosť a znižuje čas odozvy celého DIS.

4.3 Štruktúra a organizácia indexových súborov DIS

Štruktúra a organizácia sú z pohľadu tejto práce dva rôzne pojmy a je ich potrebné striktne rozlišovať. Organizácia súborov je charakterizovaná početnosťou, druhom súborov a vzájomnými vzťahmi medzi súbormi, teda skôr ich vonkajšími charakteristikami, zatiaľ čo štruktúra hovorí o vnútornom usporiadaní dát (polí, kľúčov, hodnôt a ukazovateľov) v rámci každého súboru. Táto práca bude tieto dva pojmy chápať vo význame **vnútorná štruktúra** a **vonkajšia organizácia,** napriek tomu, že v literatúre sa používa slovo organizácia aj vo význame vnútornej organizácie – napr. „súbor je organizovaný index-sekvenčne“.

Na štruktúru a organizáciu indexových súborov DIS sa je možné pozerať z dvoch rôznych pohľadov: fyzického a logického. Logický pohľad možno porovnať s pohľadom „dátového modelára“ na funkčný, respektíve logický dátový model, ktorý je súbormi tvorený (modelovaný). Vyjadruje teda logické umiestnenie dátových položiek v súboroch, a tiež vzájomné vzťahy medzi jednotlivými položkami a súbormi bez ohľadu na to, ako budú fyzicky reprezentované. Fyzický pohľad na štruktúru a organizáciu zahŕňa spôsoby fyzickej implementácie súborov, a je preto závislý na type dátového média, na ktorom budú uložené dáta. Je prirodzené, že súborová organizácia vhodná pre rýchle vyhľadávanie na pevnom disku nemusí byť najvhodnejšia pre magnetické pásky. Fyzický pohľad sa zameriava na fyzický spôsob uloženia dát (poradie položiek, umiestnenie položiek ...) na pamäťovom médiu.

4.3.1 Logická organizácia a štruktúra – teoretický rozbor

Kapitola rozoberá možnosti organizácie a štruktúry súborov indexu DIS so zameraním na vektorový model. Zameriava sa na počet, druh súborov a ich primárny obsah. Vnútorná fyzická štruktúra súborov tu nie je popisovaná.

Základný údaj – váha termu v dokumente
Zo špecifických princípov fungovania vektorového DIS vyplýva aj organizácia dátových súborov indexu. Každý vektorový DIS potrebuje na výpočet podobnosti

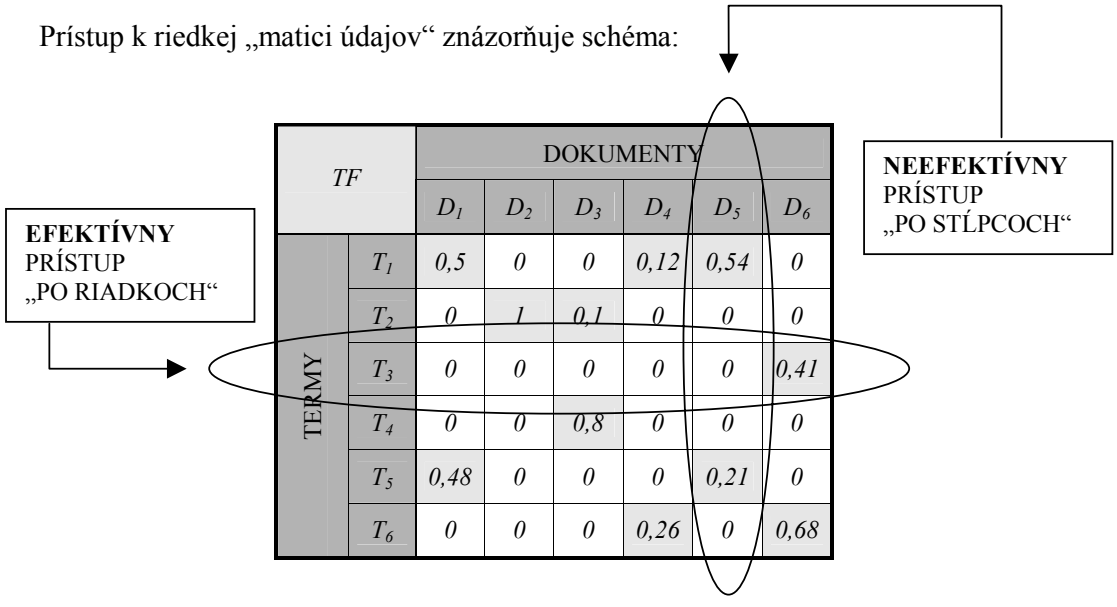
dokumentu a dotazu poznať váhy termov v danom dokumente. Základnou úlohou indexu je teda poskytovať váhu každého termu pre každý dokument v dostatočne krátkom čase. Tento fakt priamo vedie k myšlienke reprezentovať jadro indexu DIS ako maticu, kde na pozícii $[i, j]$ je váha i-tého termu v j-tom dokumente. Pre reálne termy a dokumenty je však väčšina váh nulová. Dôvodom je napríklad neexistencia termu v dokumente, alebo z dôvodu malého významu termu v danom dokumente, čím sa problém reprezentácie indexu transformuje na problém efektívnej implementácie riedkej matice. Vysoká efektivita implementácie je potrebná hlavne z dôvodu nutnosti spočítať podobnosť dotazu so všetkými potenciálne vyhovujúcimi dokumentmi pri každom zadanom dotaze – štatisticky je získanie váhy termu v dokumente najčastejšie vykonávaná operácia.

Spôsob prístupu k údajom

K spomínanej „matici údajov“ (dokument x termy) sa každý z modelov DIS správa inak:

- *Boolský model* DIS v zásade potrebuje k zadaným termom (vstupu) nájsť všetky relevantné dokumenty (výstup), a preto pristupuje k tejto „matici údajov“ len cez termy, čo sa dá výhodne priestorovo aj časovo implementovať pomocou invertovaných zoznamov termov. Je samozrejmé, že boolský model DIS nepotrebuje poznať skutočné váhy termov v dokumentoch, ale na výpočet mu stačia len identifikátory vyjadrujúce, či sa daný term v dokumente vyskytuje alebo nie (1/0, true/false).
- Problém nastáva pri použití *vektorového modelu* DIS. Ako už bolo spomenuté, na výpočet podobnosti dokumentu a dotazu totiž vektorový model môže v závislosti na použitej podobnostnej funkcii potrebovať hodnoty všetkých váh termov daného dokumentu, čo znamená, že k zadaným dokumentom (vstupu) je potrebné dodať všetky termy obsiahnuté v dokumente (výstup). Ide teda o prístup k „matici údajov“ cez dokumenty. Aby však nebolo nutné prechádzať a porovnávať všetky dokumenty uložené v indexe s dotazom, čo je časovo neúnosné, a teda prakticky nepoužiteľné, ale iba tie, v ktorých majú termy z dotazu nenulovú váhu, je potrebné pristupovať k „matici údajov“ zo strany termov. To odpovedá získavaniu potenciálne vyhovujúcich dokumentov. Pre efektívne fungovanie vektorového modelu je teda potrebný efektívny prístup z oboch smerov – po riadkoch a aj po stĺpcoch.

Prístup k riedkej „matici údajov“ znázorňuje schéma:



Je známe, že bežne používané súborové štruktúry dokážu efektívne pristupovať k akejkoľvek matici údajov, ako súboru dát, len v jednom smere a to zvyčajne po riadkoch. Dobre viditeľné to je na príklade index-sekvenčého súboru, kde je sekvenčné čítanie blokov („po riadkoch“) efektívnejšie ako čítanie blokov pri prístupe cez index („po stĺpcoch“), pretože tak nastávajú časové straty spôsobené posunom hlavičiek disku a jeho rotačným omeškaním (podobné omeškania je možné nájsť aj v prípade iného pamäťového média).

Ponúkajú sa dve riešenia. Jedným je zdvojenie tejto matice, pričom raz budú v riadkoch, na ktoré je rýchly prístup, termy a raz dokumenty. Obrovskou nevýhodou je zdvojená priestorová náročnosť jadra indexu, ktorá sa však dá čiastočne eliminovať vhodným rozložením údajov. Druhým riešením je použitie jednej matice údajov, pričom bude štruktúrovaná tak, aby sa minimalizoval počet čítaných blokov z vonkajšieho pamäťového média – nevýhodou je isté spomalenie odozvy dané fyzikálnymi vlastnosťami pamäťového média.

Spôsob uloženia váhy termu v dokumente

V spomínanej matici (dokumenty x termy) nemôže byť uložená priamo výsledná váha termu v dokumente ($w_{i,j}$), pretože tá je závislá jednak od počtu a dôležitosti výskytov daného termu v dokumente ($NTF_{i,j}$), a jednak od počtu dokumentov, v ktorých sa term vyskytuje (DF_j) a celkového počtu dokumentov v kolekcii (m). Závislosť ukazuje nasledujúci vzťah:

$$w_{i,j} = NTF_{i,j} \cdot IDF_j \text{, kde } IDF_j = \log(m / DF_j) + 1$$

Počet dokumentov, v ktorých sa term vyskytuje a celkový počet dokumentov je v dynamických kolekciách dokumentov premenlivý. S každým pridaným a odobraným dokumentom je preto nutné jednorázovo prepočítať všetky váhy termov. Táto operácia je časovo náročná a v dynamickom prostredí, akým je internet, s veľkým množstvom (aj objemných) dokumentov by svojím častým behom zbytočne zaťažovala systém. V matici (dokumenty x termy) budú teda uložené iba termové frekvencie (*TF*, prípadne *NTF*), ktoré závisia len na termoch daného dokumentu a sú teda nezávislé na iných dokumentoch. V prípade zmeny obsahu dokumentu je nutné prepočítať iba termové frekvencie termov v danom dokumente a to len pre zmenený dokument.

Kritériá uloženia položiek v indexe DIS

Pre zistenie výslednej váhy termu v dokumente je potrebné poznať aj počet dokumentov, v ktorých sa term vyskytuje. Je intuitívne, že ak sa term vyskytuje v každom dokumente z kolekcie, nie je pre rozlíšenie dvoch dokumentov podstatný. Tento fakt zohľadňuje inverzná dokumentová frekvencia termu (*IDF*), ktorej hodnota klesá s rastúcim počtom dokumentov, kde sa term vyskytuje. Inverzná dokumentová frekvencia vyjadruje vzťah termu k celej kolekcii dokumentov, zatiaľ čo termová frekvencia vyjadruje vzťah jedného termu ku konkrétnemu dokumentu, čo je kvantitatívne niečo iné - údaje majú inú granularitu. Práve granularita informácie je jedným z najdôležitejších indikátorov jej uloženia v súboroch indexu DIS.

Ďalším kritériom, podľa ktorého je možné rozhodovať o umiestnení položky (informácie) v systéme súborov indexu DIS je jej význam. V zásade podľa významu rozlišujeme štyri typy položiek - identifikačné, existenčné, pozičné a pomocné.

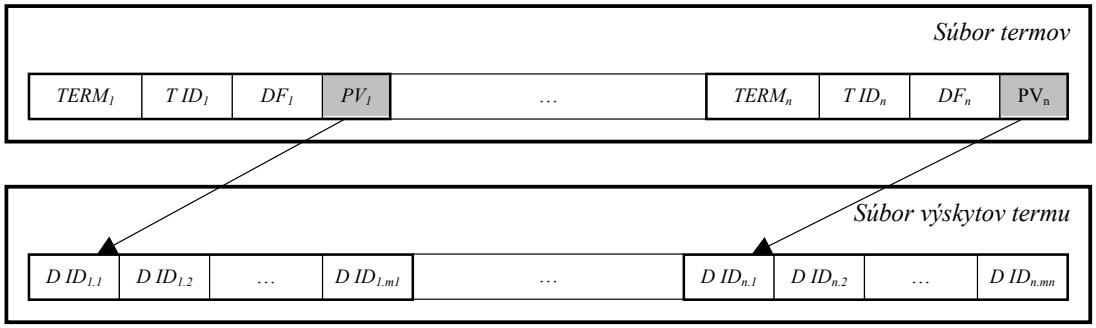
- *Identifikačné položky* jednoznačne identifikujú entity (termy, dokumenty, odstavce...). Identifikátory môžu mať rôznu podobu – textovú (term, URL dokumentu), číselnú (odstavec, veta), binárnu (signatúra termu alebo dokumentu).
- *Existenčné položky* vyjadrujú existenciu a mieru dôležitosti termu v dokumente. V praxi ide o položky typu boolean a štatistické údaje akými sú termová frekvencia, inverzná dokumentová frekvencia a dĺžka dokumentov.
- *Pozičné položky* vyjadrujú pozície, na ktorých sa entity vyskytujú v iných entitách. Používané pozičné položky sú hlavne pozície termu v celom dokumente, odstavci a vete, prípadne sú to údaje o poradí viet a odstavcov ako celkov.
- *Pomocné položky* sú hlavne ukazovatele do indexových súborov vo forme absolútnych a relatívnych pozícií (offset) v súbore.

Výsledná organizácia indexových súborov

Princípy a spôsob fungovania vektorového modelu DIS nedávajú príliš veľké šance na efektívnu implementáciu indexu. Hlavným dôvodom neefektívnosti je nutnosť uloženia matice údajov o termoch v dokumentoch a jej následné čítanie „po riadkoch“ a aj „po stĺpcoch“. Keďže v súčasnosti žijeme v dobe, kde „čas sú peniaze“, zdá sa rozumnejšie zvoliť takú implementáciu indexu, ktorá by mala čo najmenšiu odozvu a to aj za cenu ukladania redundantných informácií. Matica údajov sa preto v indexovej organizácii bude vyskytovať dvakrát, no zakaždým v inej štruktúre. Raz budú údaje (termové frekvencie) uložené tak, aby sa k nim dalo rýchlo pristupovať „cez doklumenty“ a druhý krát bude štruktúra optimalizovaná na prístup „cez termy“. Z princípu fungovania DIS vo vektorovom modeli vyplýva, že pri získavaní potenciálne vyhovujúcich dokumentov (prístup „cez termy“) nie je potrebné poznať váhy termov v dokumentoch, ale stačí len údaj o existencii, prípadne neexistencii termu v dokumente. Tento fakt je základom zníženia redundancie údajov. Konkrétne riešenie predstavuje nasledujúca schéma a jej rozbor.

Hlavným cieľom je teda čo najmenší počet prístupov na pevný disk. Tomu je podriadená organizácia aj štruktúra súborov. Z ohľadom na kritériá a obmedzenia naznačené v tejto kapitole je možné výslednú organizáciu a štruktúru naznačiť schematicky takto:

Prístup „cez termy“ (pokrýva 1.fázu vyhľadávania)



Prístup „cez dokumenty“ (pokrýva 2. a 3.fázu vyhľadávania)

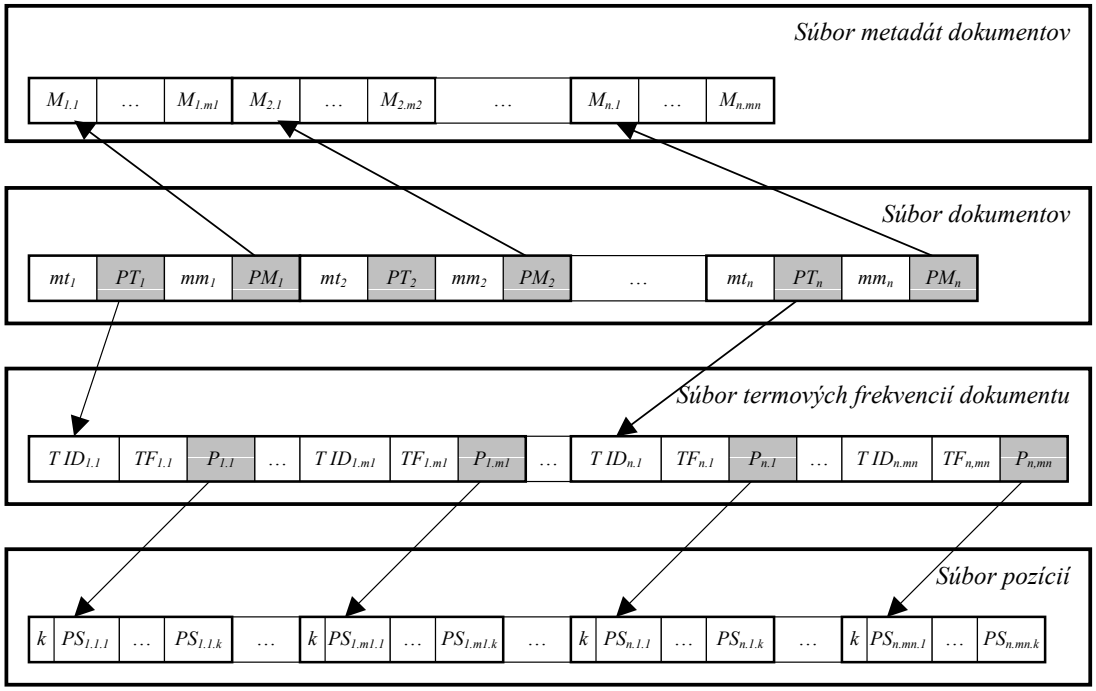


Schéma indexu vektorového DIS má niekoľko súborov:

- *Súbor termov* – súbor je zoznamom termov. Atómami sú štvorice – textová reprezentácia termu (*TERM*), technická číselná reprezentácia termu (*T ID*), dokumetová frekvencia (*DF*) a ukazovateľ do *súboru výskytov termu* (*PV*). Súbor má štruktúru index-sekvenčného súboru, pričom atómy sú zoradené podľa textovej reprezentácie termu. To dáva možnosť na úspornejšie uloženie samotného termu pomocou sufixovej kompresie. Pre efektívne fungovanie DIS by nad týmto súborom mali existovať dva indexy. Jeden na rýchly prístup k textovým identifikátorom termov, ktorý je potrebný pri samotnom vyhľadávaní a tvorbe indexu. Druhý index by mal byť vytvorený nad technickým identifikátorom termov a mal by slúžiť pri dotazoch referenčným dokumentom. Tento súbor je štartovacím bodom procesu vyhľadávania, a teda aj štartovacím bodom prvej fázy.
- *Súbor výskytov termu* – súbor ku každému termu zo *súboru termov* obsahuje zoznam dokumentov, v ktorých má term nenulovú termovú frekvenciu. Počet dokumentov, v ktorých sa term vyskytuje s nenulovou *TF* je uložený v položke *DF* v *súbore termov*. Atómami sú n-tice identifikátorov dokumentov (*D ID*). V rámci jedného atómu sú identifikátory zoradené vzostupne a je preto možné použiť rozdielovú kompresiu (namiesto identifikátora dokumentu sa ukladá rozdiel medzi identifikátorom predchádzajúceho a identifikátorom ukladaného dokumentu). Ak by tento súbor obsahoval aj termové frekvencie termu v dokumentoch, bol by to plnohodnotný invertovaný súbor termov používaný v implementáciách boolských DIS.
- *Súbor dokumentov* – je štartovacím bodom druhej fázy vyhľadávania. Jeho atómy sú tvorené dvoma pomocnými a dvoma identifikačnými položkami – počtom termov a ukazovateľom do *súboru termových frekvencií dokumentu* (*PF*), počtom metadát dokumentu a ukazovateľom do *súboru metadát dokumentov* (*PM*). Keďže

dokumenty sú jednoznačne identifikované číselným identifikátorom, je možné uložiť atómy tak, že atóm príslušný k dokumentu s identifikátorom *n* je *n*-tým atómom v súbore. Pri veľkosti ukazovateľa 4Byte to znamená, že *n*-tý atóm je na pozícii 4*4**n* Bytov. Tento spôsob uloženia umožňuje veľmi rýchly prístup k ďalším údajom ktoréhokoľvek dokumentu.

- *Súbor termových frekvencií dokumentu* – je implementáciou riedkej matice termových frekvencií termov v dokumentoch. Ako je vidieť zo schémy, atómy sú tvorené číselným identifikátorom termu (*T ID*), samotnou termovou frekvenciou termu v príslušnom dokumente (*TF*) a ukazovateľom do *súboru pozícií*. Súbor je štruktúrovaný ako zoznam termov a termových frekvencií pre každý dokument a je používaný v druhej fáze vyhľadávania, kedy sa počíta podobnosť príslušného dokumentu a dotazu. V tejto fáze už nie je dôležité poznať textové identifikátory všetkých termov dokumentu. Stačí poznať len číselné identifikátory termov dotazu, čím je možné „spárovať“ termy dokumentu a termy dotazu. Keďže niektoré podobnostné funkcie pre svoj výpočet potrebujú termové frekvencie všetkých termov dokumentu a niektoré len termové frekvencie termov z dotazu, nie je pri implementácii *T ID* použitá rozdielová kompresia. Zároveň majú atómy súboru pevnú veľkosť, čo umožňuje použiť, v porovnaní s lineárnym prechodom, efektívnejšie algoritmy vyhľadávania potrebnej termovej frekvencie, akým je napríklad polenie intervalu. V prípade implementácie boolského DIS, by tento súbor neexistoval a termové frekvencie vy boli uložené v invertovaných zoznamoch termov – v tejto súborovej organizácii to odpovedá *súboru výskytov termu*.
- *Súbor pozícií* – využíva sa v tretej fáze vyhľadávania, kedy sú používateľovi zobrazené metadáta o dokumente a jeho krátka ukážka. Práve krátka ukážka je vygenerovaná podľa pozícií výskytov hľadaných termov, aby používateľ videl kontext, v ktorom sú hľadané termy použité. V boolských systémoch sa *súbor pozícií* využíva aj pri dotazovaní. Atómy sú tvorené k-ticou pozícií výskytov termu v dokumente (*PS*). Prvý údaj atómu (*k*) hovorí o počte výskytov termu v dokumente. Je pravdou, že údaj *TF* zo *súboru termových frekvencií* dokumentu veľmi úzko súvisí s počtom výskytov termu v dokumente, no v *TF* môžu byť (a aj sú) zohľadnené rôzne ďalšie faktory (výskyt termu v nadpise, tučné písmo, normalizácia váh ...). Nejde teda o redundanciu.
- *Súbor metadát dokumentov* – podobne ako v prípade *súboru pozícií,* ide o súbor používaný v tretej fáze vyhľadávania. Atómy sú tvorené postupnosťou metadát o dokumente. Keďže typov metadát môže byť niekoľko a niektoré z nich dokonca ani nemajú význam pre všetky dokumenty (URL v prípade skutočného časopisu, počet strán v prípade HTML stránky), je každý typ označený svojím kódom. Jednotlivé položky atómov majú potom tvar: *kód-text*. Kódy môžu mať číselnú, alebo textovú podobu. Dôležité je, aby boli jednoznačné - unikátne. Tento spôsob umožňuje vysokú variabilitu a štruktúrovanie typov metadát.

Už na prvý pohľad je jasné, že nedochádza k duplicitnému uloženiu položky *TF*, pretože pri získavaní potenciálne vyhovujúcich dokumentov nie je tento údaj potrebný. Dochádza iba k redundandnému uloženiu existenčných údajov - identifikátorov termov a dokumentov v *súbore termov*, *súbore výskytov termu* a *súbore termových frekvencií dokmentu*. Väčšina redundantných údajov sú navyše číselné identifikátory, ktoré je možné ukladať v komprimovanom tvare. Organizácia však plne akceptuje a podporuje spôsob výpočtu vektorového DIS. Výpočet začína vyhľadaním termu z dotazu v index-sekvenčnom *súbore termov*, ktorého atómy sú

zoradené práve podľa termu. Nad týmto súborom by mal existovať index uložený v pamäti umožňujúci priamy prístup k údajom. Po prečítaní príslušnej dokumentovej frekvencie (teda počtu dokumentov, kde má term nenulovú váhu) zo *súboru termov* sa pomocou odkazu do *súboru výskytov termu* získa množina relevantných dokumentov. V prvom kroku vyhľadávania sa do dočasných vnútorných štruktúr musia uložiť aj technické identifikátory termov (*T ID*), ktoré budú použité pri výpočte podobnosti dotazu a dokumentu.

Je vhodné uvedomiť si fakt, že pri postupe práce s indexom popísaným na začiatku kapitoly 4.2 a v podkapitole 4.3.2 nie je potrebné vyhľadávať v indexe podľa používateľského identifikátoru (URL, názov knihy, názov článku) dokumentu. Dokonca aj v systéme dodania metadát dokumentu sa do indexu pristupuje cez technický identifikátor dokumentu, ktorý je výsledkom vyhľadávania DIS. Preto sa vo všetkých indexových súboroch používajú technické identifikátory dokumentov v podobe celých, obvykle kladných, čísel, čo umožňuje využiť niektoré metódy redukcie veľkosti dát a zároveň nemá žiadne ďalšie vedľajšie negatívne dôsledky.

Horšie je to s identifikáciou termov. Termy vyseparované z dotazu sú identifikované svojou reťazcovou (textovou) podobou a primárne tak aj vstupujú do procesu vyhľadávania. V indexových súboroch sa termy vyskytujú na dvoch miestach (v *súbore termov* a *súbore termových frekvencií dokumentu*), čo by mohlo byť dôvodom pre zavedenie kratších číselných identifikátorov termov, ktoré by boli použité v oboch súboroch. Tým by sa síce ušetril diskový priestor, no pri každom vyhľadávaní by sa musel k termu reprezentovanému reťazcom nájsť jeho jednoznačný číselný identifikátor. Ako vhodné sa javí riešenie, kde sa do indexu pristupuje primárne pomocou textových identifikátorov. Navyše je v *súbore termov* aj jednoznačný číselný identifikátor, čím súbor slúži ako prevodník medzi dlhými textovými a krátkymi číselnými identifikátormi. Číselný identifikátor termu je „naostro“ použitý len v *súbore termových frekvencií dokumentu*. Zmyslom je to, že v *súbore termových frekvencií dokumentu* sa každý term môže vyskytovať niekoľkokrát, čím vzniká značná úspora diskového priestoru. Zároveň to nekladie žiadne ďalšie nároky v procese vyhľadávania. Proces tvorby a údržby indexu sa tým mierne skomplikuje.

V druhom kroku sa prechádzajú potenciálne vhodné dokumenty, ku ktorým sa podľa potreby získavajú termové frekvencie termov. So získanými údajmi je už možné porovnávať dokument s dotazom pomocou akejkoľvek podobnostnej funkcie.

4.3.2 Vyhodnotenie dotazu – teoretický rozbor

Vyhodnotenie vektorového dotazu je v navrhovanej organizácii trojfázové.

Prvá fáza

Cieľom prvej fázy je hlavne korektné vytvorenie vektoru termov a ich požadovaných váh (či už separáciou termov z dotazu, alebo ich nájdenie v indexe) a následné nájdenie relevantných dokumentov k vyseparovaným termom.

Postup je do značnej miery závislý od spôsobu zadania dotazu. Dotaz môže byť okrem základného tvaru (vektor termov a ich požadovaných váh) zadaný aj odkazom na iný, už zaindexovaný dokument (referenčným dokumentom), prípadne aj úplne novým dokumentom.

V prípade zadania dotazu vo forme nezaindexovaného dokumentu je nutné daný dokument zaindexovať pomocou štandardného algoritmu a získať tak vektor termov, ktorý je už možné použiť ako vstup do procesu vyhľadávania.
V prípade, že je dotazom referencia na zaindexovaný dokument, závisí postup od formy odkazu. Používateľ systému si totiž môže vybrať z dokumentov ponúkaných systémom v nejakom výberovom menu, prípadne vo výsledku predchádzajúceho vyhľadávania, kedy systém pozná technické identifikátory ponúkaných dokumentov. Ohodnotený vektor termov sa získa jednoduchým prístupom cez *súbor dokumentov* do *súboru termových frekvencií dokumentu*. Ak však používateľ systému zadá odkaz na dokument pomocou názvu, prípade URL je situácia horšia. Bez ujmy na všeobecnosti predpokladajme, že bude dokument zadaný svojím URL. URL dokumentu je ako všetky ostatné metadáda uložené v *súbore metadát dokumentov*, pričom zo schémy organizácie je jasné, že z neho nevedú žiadne odkazy do *súboru dokumentov*, pretože ich vzájomná väzba je opačná. Riešením je len sekvenčné vyhľadávanie zadaného URL v *súbore metadát dokumentov*, ktoré je v prípade nájdenia URL nasledované sekvenčným prehľadávaním *súboru dokumentov*, kde je potrebné nájsť odkaz na nájdené URL. Je evidentné, že tento spôsob je neefektívny. Nahradiť by sa dal vytvorením indexu nad *súborom dokumentov*. Nepríjemné je, že pre každý typ metadát by bol pravdepodobne potrebný samostatný indexový súbor, čo tiež nie je úplne optimálne. Toto je miesto, ktoré by mohlo byť predmetom ďalšieho skúmania. Je dobré si uvedomiť, že referenčným dokumentom je možné získať len vektor identifikátorov termov (*T_ID*) a ich termových frekvencií. Kvôli rýchlemu prístupu do *súboru termov* je preto potrebné, aby existovali dva indexy nad *súborom termov* – jeden podľa textových a druhý podľa technických identifikátorov termov (viď popis *súboru termov* pod schémou v kapitole 4.3.1.) Fakticky je koverzia používateľského dotazu na systémový dotaz úlohou modulu KONVERTOR. Ten však na to potrebuje informácie z indexových súborov, konverzia je preto zahrnutá v prvej fáze vyhľadávania (práce s indexom).

V každom z troch uvedených prípadov zadania dotazu je v konečnom dôsledku dotazom vektor ohodnotených termov. Niekedy je potrebné tieto termy lemmatizovať – nájsť ich základný morfologický tvar. Potom sa termy z vektora separujú a postupne pomocou indexu vyhľadávajú v *súbore termov*. Odtiaľ sa pomocou odkazu prečítajú všetky potenciálne vyhovujúce dokumenty. Výsledkom prvej fázy je množina dokumentov, ktoré sú potenciálne vyhovujúce systémovému dotazu. Ide teda o zjednotenie potenciálne vyhovujúcich dokumentov jednotlivých termov dotazu.

Druhá fáza
V druhej fáze vyhľadávania sa získané dokumenty pomocou príslušnej podobnostnej funkcie porovnávajú so zadaným dotazom a zoraďujú sa podľa podobnosti s dotazom. Cieľom je teda nájdenie *x* dotazu najpodobnejších dokumentov, prípadne všetkých dokumentov, ktorých podobnosť s dotazom je aspoň sim_{min}. Fáza sa začína prístupom do *súboru dokumentov*. Tento prístup je rýchly vďaka blokovej štruktúre a prístupu pomocou technického identifikátoru dokumentu (dáta *n*-tého dokumentu sú uložené na pozícii 4*4**n* Bytov). Odtiaľ vyhľadávanie pokračuje do *súboru termových frekvencií dokumentu*, kde sú uložené termové frekvencie všetkých termov daného dokumentu.

Práve táto fáza vyhľadávania poskytuje dostatok priestoru na rôzne optimalizácie. Ako bolo uvedené v popise *súboru termových frekvencií dokumentu*, atómy majú

konštantnú veľkosť. Počet atómov dokumentu odpovedajúci počtu zaindexovaných termov dokumentu systém pozná zo *súboru dokumentov* (položka *mm*). Za predpokladu, že atómy sú v súbore zoradené vzostupne podľa položky *T ID*, je možné využiť na vyhľadanie konkrétneho atómu (termu) algoritmus polenia intervalu. Tým sa zníži počet prístupov do súboru z lineárneho počtu (sekvenčné prehľadávanie) na logaritmický počet z počtu atómov. Táto optimalizácia sa javí ako opodstatnená, pretože niektoré podobnostné funkcie požadujú len termové frekvencie termov, ktoré boli zadané v dotaze, ostatné sú pre ne irelevantné. Naopak, niektoré podobnostné funkcie potrebujú pre výpočet všetky termové frekvencie termov dokumentu – v tomto prípade sa využije sekvenčné čítanie súboru.

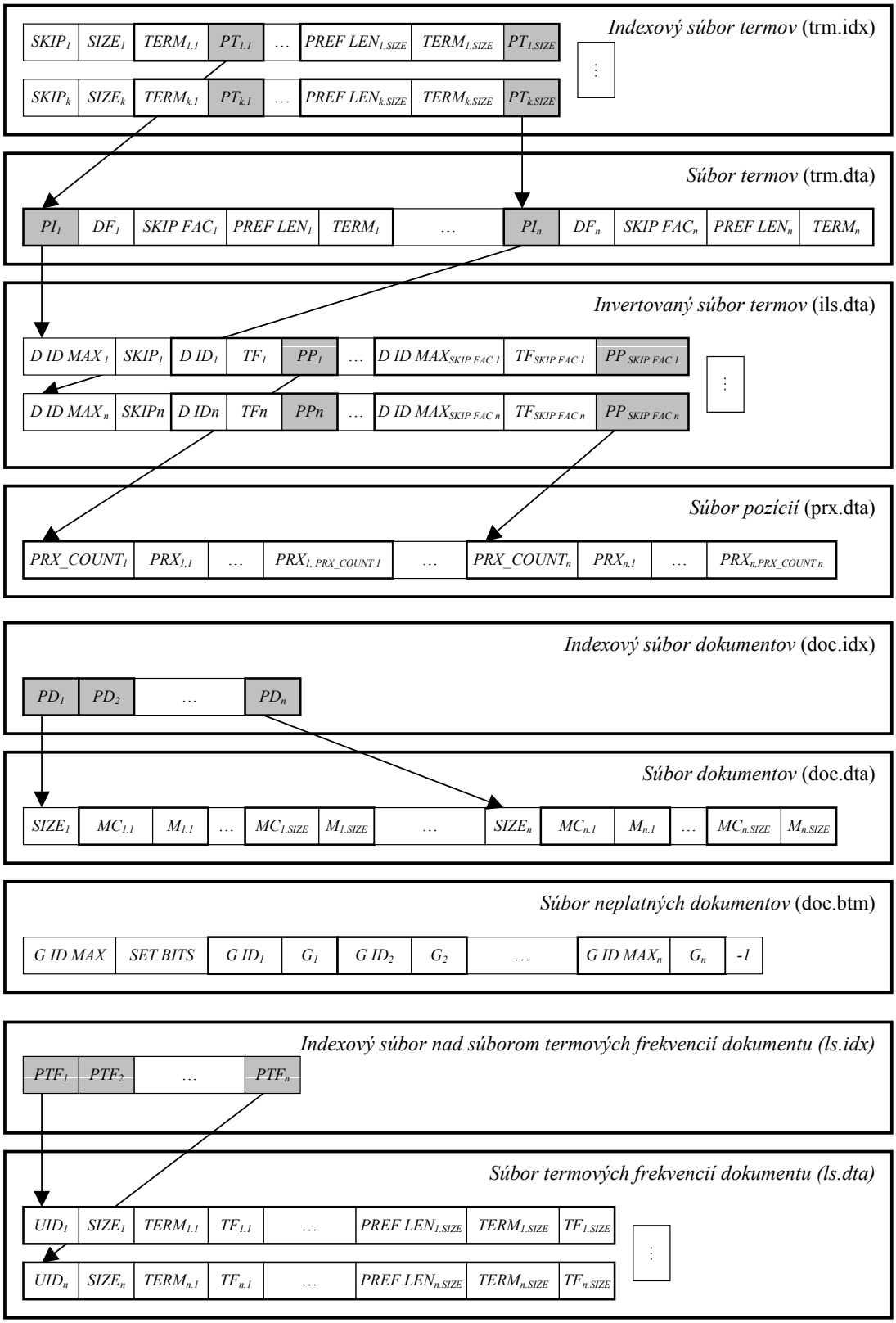
Za zmienku stojí aj možnosť zoradenia atómov v súbore podľa klesajúcej hodnoty *TF* a využiť systémové optimalizácie popísané v kapitole 4.4.1. Ide o možnosť parciálne usporiadať dokumenty už počas vyhľadávania a ušetriť tak niekoľko prístupov do indexu. Táto možnosť je však vhodnejšia pre boolské DIS, pretože jedným z primárnych predpokladov účinnosti je jednotková váha termov v dotaze, čo vlastne znamená, že ju systém neberie do úvahy.

Tretia fáza
Úlohou tretej fázy vyhľadávania je zrozumiteľné poskytnutie výsledku používateľovi. Výsledok by mal obsahovať základné metadáta dokumentov, ktoré najviac vyhovujú dotazu a prípadne aj ich krátku kontextovú ukážku. Na tieto účely sa využívajú dva súbory: *súbor metadát dokumentu* a *súbor pozícií*. Do prvého z nich sa je možné dostať priamo zo *súboru dokumentov*, čo nie je časovo veľmi náročné. Horšie je to z generovaním kontextovej ukážky, pretože na zistenie pozície termu v danom dokumente je potrebné najprv vyhľadať term z dotazu v *súbore termových frekvencií dokumentu* a až potom prečítať pozície príslušného termu v dokumente zo *súboru pozícií* (viď schéma). Je jasné, že tento postup v sebe skrýva niekoľko prístupov do rôznych súborov indexu. Takýmto spôsobom je možné zistiť, ktorá časť dokumentu by sa používateľovi mala predložiť ako ukážka. Problém však spočíva v tom, že systém nemá v indexových súboroch uložený samotný dokument, a ani ho na základe údajov uložených v indexe nedokáže vygenerovať. Systém by preto mal mať niekde (mimo indexové súbory) uložené aspoň neformátované zdrojové texty indexovaných dokumentov. Bez nich totiž nie je možné poskytnúť používateľovi ukážku. Pri rozsiahlejších kolekciách dokumentov však môže byť objem zdrojových textov príliš veľký. Tieto zdrojové texty nemôžu byť uložené na pamäťových médiách s veľkou časovou odozvou, pretože generovanie ukážky by bolo v porovnaní so samotným vyhľadávaním pomalé. Implementácia systému Egothor kontextovú ukážku poskytuje v svojom HTML rozhraní, no novo vytvorený DIS túto možnosť v tejto verzii neposkytuje. V jeho výsledku je uvedený len sumár dokumentu tvorený najvýznamnejšími termami dokumentu.

4.3.3 Použitá organizácia indexových súborov a vyhodnotenie dotazu

V predchádzajúcej kapitole bol vytvorený návrh organizácie indexu so zameraním na vektorový model DIS. Keďže táto práca naväzuje na už funkčný DIS v rozšírenom boolskom modeli (Egothor), organizácia súborov je po dôkladnom preštudovaní prebratá. Prebraté sú všetky indexové súbory systému Egothor, ku ktorým sa jednorázovým spustením vytvorenej rutiny (súčasť systému FRC Finder) zreorganizujú potrebné informácie o termoch dokumentov. Prakticky sa vytvoria dva nové súbory: *Súbor termových frekvencií dokumentu* a *Index nad súborom termových*

frekvencií dokumentu. V podstate ide o invertovaný súbor dokumentov a index nad invertovaným súborom dokumentov. Schematicky možno použitú organizáciu znázorniť nasledovne:



Celkovo vyzerá organizácia nasledovne:

- *Indexový súbor termov* (trm.idx) – súbor je indexom nad dátovým *súborom termov,* a práve preto sa v ňom začína celý proces vyhľadávania. Je organizovaný po blokoch (skupinách) termov, pričom každý blok začína svojím abecedne posledným (lexikograficky najväčším) termom. Ostatné termy bloku sú uložené v abecednom poradí. Spôsob uloženia urýchľuje prístup k dátam termu. V tomto indexe sa vyhľadáva tak, že sa postupne prechádzajú polohovo prvé termy blokov (abecedne posledné) dovtedy, kým hľadaný term nie je lexikograficky menší ako práve načítaný prvý term bloku. Ak sa tak stalo, znamená to, že hľadaný term sa môže vyskytovať už len v tomto bloku termov. Potom sa postupne prechádzajú termy bloku, až kým aktuálne čítaný term z bloku nie je lexikograficky menší alebo rovný ako hľadaný term. Ak už je, tak sa skontroluje, či je rovný (a teda term v indexe existuje) alebo menší (term sa v indexe nenachádza). Výsledkom vyhľadávania v *indexovom súbore termov* je odkaz na informácie o terme v dátovom *súbore termov*. Každý blok termov má svoju hlavičku a telo. Hlavička bloku je tvorená dvoma položkami: počtom termov bloku vrátane prvého termu (*SIZE*) a veľkosťou bloku v Bytoch (*SKIP*), ktorá sa používa na rýchly prechod na nasledujúcu skupinu termov. Telo bloku je tvorené skupinou *SIZE* atómov, pričom jednotlivé atómy majú nasledujúce položky:
    - *PREF LEN* (prefix length) – položka určuje dĺžku spoločného prefixu s termom v predchádzajúcom atóme. Prvý term (atóm) bloku túto položku nemá uloženú, pretože je uložený kompletne. Termy v ostatných atómoch bloku sú uložené pomocou prefixovej kompresie.
    - *TERM* – textová reprezentácia termu – presnejšie sufixu termu, ktorý ho odlišuje od termu v predchádzajúcom atóme.
    - *PT* – odkaz na informácie o terme v *súbore termov*.
- *Súbor termov* (trm.dta) – dátový súbor termov. Uchováva základné informácie o termoch v granularite term x kolekcia dokumentov. Pre každý term tento súbor obsahuje hlavne počet dokumentov (*DF*), v ktorých sa vyskytuje s nenulovou váhou (z tohto údaju je možné odvodiť inverznú dokumentovú frekvenciu). Je dôležité, aby informácie v granularite term x dokument a term x kolekcia dokumentov boli v súborovej organizácii uložené v rozdielnych súboroch (viď kapitola 4.3.1 o implementácii indexových súborov DIS). Atómy súboru obsahujú nasledujúce položky:
    - *PI* – ukazovateľ do *invertovaného súboru termov*. Na pozícii *PI* v *invertovanom súbore termov* je uložený blok dokumentov a príslušných termových frekvencií termu.
    - *DF* – počet dokumentov, v ktorých sa term vyskytuje s nenulovou váhou. Z *DF* je možné odvodiť dokumentovú frekvenciu termu.
    - *SKIP FAC* (skip factor) – položka slúži na optimalizáciu počtu prístupov na disk pri paralelnom čítaní dvoch alebo viacerých zoznamov z *invertovaného súboru termov*. Určuje veľkosť blokov, z ktorých je poskladaný invertovaný zoznam termu.
    - *PREF LEN* (prefix length) – položka určuje dĺžku spoločného prefixu s predchádzajúcim termom. Je to implementácia prefixovej kompresie.
    - *TERM* – textová reprezentácia termu. Keďže termy sú uložené pomocou prefixovej kompresie, ide len o časť termu (sufix), ktorá je odlišná od predchádzajúceho termu.

- *Indexový súbor dokumentov* (doc.idx) – súbor je indexom nad dátovým *súborom dokumentov* a umožňuje rýchly prístup k dátam dokumentov. Atómy súboru obsahujú jedinú pomocnú položku – ukazovateľ do *súboru dokumentov* (*PD*). Je organizovaný tak, že na pozícii *CONST***D ID* (*D ID* je jednoznačný identifikátor hľadaného dokumentu v rámci danej časti indexu – barelu a *CONST* je veľkosť dátového typu long) je uložený ukazovateľ do *súboru dokumentov* (*PD*) určujúci polohu dát dokumentu s identifikátorom *D ID*. Súbor je teda postupnosťou pozícií v dátovom súbore dokumentov.
- *Súbor dokumentov* (doc.dta) – dátový súbor dokumentov. Obsahuje hlavne metadáta o dokumentoch (názov dokumentu, umiestnenie dokumentu, URL, počet strán...). Metadáta sú uložené v dvojiciach: kód metadát (*MC*), metadáta (*M*). Kód metadát je zvyčajne jedno písmeno (S-summany, T-title) a slúži ako identifikátor typu metadát. Pre každý dokument je v *súbore dokumentov* najprv uložený počet takýchto dvojíc (*SIZE*) a potom samotné dvojice. Vo výsledku je súbor organizovaný blokovo, kde jeden blok odpovedá dátam jedného dokumentu. Každý blok má svoju hlavičku a telo. Hlavička bloku obsahuje už zmienený počet metadát pre daný dokument (*SIZE*). Telo bloku obsahuje *SIZE* dvojíc (atómov) kód metadát (*MC*), metadáta (*M*).
- *Súbor neplatných dokumentov* (doc.btm) – Súbor slúži na identifikáciu dokumentov, ktoré sú už neplatné (boli odstránené). Obsahuje bitovú mapu, kde hodnota 1 znamená, že dokument bol odstránený a hodnota 0, že dokument je stále platný. Kvôli zníženiu priestorovej a časovej náročnosti je telo súboru organizované po skupinách (*G*). Pre lepšiu predstavu je možné skupinu dokumentov chápať ako riadok bitovej mapy. Jedna skupina ma veľkosť 4kB, čo znamená, že pokrýva 4096 dokumentov. V súbore nemusia byť uložené všetky skupiny (riadky) bitovej mapy. Ak sa skupina v súbore nevyskytuje, je chápaná ako skupina, z ktorej ešte nebol odstránený žiadny dokument (všetky jej dokumenty sú platné a majú teda nastavenú hodnotu 0). Každá skupina (*G*) je jednoznačne identifikovaná svojím indexom (*G ID*). Dokument s jednoznačným identifikátorom *D ID* sa nachádza v skupine s indexom $G\ ID = \lfloor D\ ID / 4096 \rfloor$ (indexy skupín začínajú 0), pričom v danej skupine bude na pozícii $U\ ID - (G\ ID * 4096)$. Pri zisťovaní platnosti dokumentu je teda potrebné nájsť príslušnú skupinu a zistiť nastavenie bitu na príslušnej pozícii (v prípade neexistencie skupiny je dokument platný). Aby bolo možné identifikovať koniec bitovej mapy, respektíve poslednú načítanú skupinu, a tým aj koniec súboru, je posledný bit nastavený na hodnotu –1. Súbor má hlavičku a telo. Hlavička súboru má nasledujúce položky:
  - *G ID MAX* – maximálny index skupiny bitovej mapy, ktorá je v súbore zapísaná.
  - *SET BITS* – počet jednotkových bitov v celej bitovej mape. Položka určuje celkový počet dokumentov, ktoré sú už neplatné.

  Telo súboru tvorí postupnosť atómov z nasledovnými položkami:
  - *G ID* – index skupiny bitovej mapy.
  - *G* – samotná skupina bitovej mapy. Fyzicky však v súbore nie sú uložené postupnosti znakov 0 a 1, ale hodnoty 0 a 1 sú reprezentované na úrovni bitov. To znamená, že v súbore sú uložené znaky vytvorené z príslušného bitového reťazca.
- *Invertovaný súbor termov* (ils.dta) – ako už napovedá názov súboru, sú v ňom uložené invertované zoznamy výskytov termov v dokumentoch. To znamená, že

pre každý term zo *súboru termov* je tu uložený zoznam dokumentov, v ktorých sa vyskytuje (s nenulovou váhou) spoločne s váhou termu v dokumente a s odkazom do *súboru pozícií*, kde je potom možné nájsť presné pozície výskytu termu v danom dokumente. Súbor je organizovaný ako zreťazenie invertovaných zoznamov, pričom odkazy na jednotlivé invertované zoznamy sú v dátovom *súbore termov*. Každý invertovaný zoznam je štruktúrovaný blokovo (podobne ako *indexový súbor termov*). Každý blok má hlavičku a telo. V hlavičke je uvedené číslo posledného dokumentu v bloku (*D ID MAX*) a počet bytov (*SKIP*), o ktoré je nutné posunúť ukazovateľ v *invertovanom súbore termov*, aby ukazoval na začiatok nasledujúceho bloku. V tele bloku sú potom uložené samotné dáta – atómy tvorené identifikátorom dokumentu (*D ID*), váhou termu v dokumente (*TF*) a ukazovateľom na zoznam výskytov termu v dokumente do *súboru pozícií* (*PP*). Ako už bolo spomenuté, organizácia súboru je podobná s organizáciou *indexového súboru termov* a umožňuje rýchlejšie (v porovnaní s lineárnym prechodom) vyhľadanie váhy termu v požadovanom dokumente. Bloky rôznych invertovaných zoznamov nemajú rovnakú veľkosť, avšak bloky jedného zoznamu sú vždy rovnako veľké (s výnimkou posledného bloku zoznamu, ktorý môže byť menší). Veľkosť jedného bloku invertovaného zoznamu je určená položkou *SKIP FAC* (skip factor), z dátového *súboru termov*. V prípade, že *SKIP FAC* je rovný 0, nemá invertovaný zoznam bloky, a teda ani hlavičky blokov (obsahuje len jedno telo bloku). Poznámky a doplnenia k položkám:

  - *D ID MAX* - číslo posledného dokumentu v bloku uložené pomocou rozdielovej kompresie. Je to len inkrementálna hodnota voči predchádzajúcemu bloku v tom istom invertovanom zozname. Skutočné číslo sa preto vypočíta ako súčet *D ID MAX* predchádzajúceho bloku (prípadne 0, ak ide o prvý blok zoznamu) a aktuálne prečítanej hodnoty.
  - *SKIP* – veľkosť bloku v Bytoch.
  - *D ID* – jednoznačný identifikátor dokumentu. Podobne ako v hlavičke bloku je použitá rozdielová kompresia, čo znamená, že je tu uložená inkrementálna hodnota voči predchádzajúcej položke v tele bloku (môže to byť aj posledná položka v tele predchádzajúceho bloku toho istého invertovaného zoznamu).

- *Súbor pozícií* (prx.dta) – súbor výskytov termov v dokumentoch. Pre každý term a dokument (ak taká kombinácia existuje v invertovaných zoznamoch termov) obsahuje *súbor pozícií* zoznam všetkých výskytov (presnejšie polôh výskytov) termu v príslušnom dokumente. Súbor je organizovaný blokovo – jeden blok odpovedá výskytom jedného termu v jednom dokumente. Položky bloku:
  - *PRX_COUNT* – počet výskytov termu v danom dokumente.
  - *PRX* – poloha (miesto výskytu) termu v dokumente. Táto položka je zaznamenávaná pomocou rozdielovej kompresie, t.j. inkrementálne voči predchádzajúcemu výskytu v zozname (prvý výskyt termu je samozrejme uvedený absolútne).
- *Súbor termových frekvencií dokumentu* (ls.dta) – tento súbor nie je súčasťou indexu systému Egothor. Musí byť vygenerovaný jedným z modulov systému FRC Finder z už existujúcich súborov indexu. Logicky je „transpozíciou“ k *invertovanému súboru termov* a slúži na rýchlejší prístup k váham termov v danom dokumente. Súbor obsahuje zoznamy termov v dokumentoch, t.j. pre každý dokument je tu uložený zoznam termov a ich termových frekvencií, ktoré sa v ňom vyskytujú s nenulovou váhou. Vnútorná organizácia je bloková, pričom jeden blok odpovedá termom jedného dokumentu. Hlavička bloku obsahuje

jednoznačný identifikátor dokumentu v príslušnej časti indexu (barreli) (*UID*) a počet termov bloku (*SIZE*). V tele bloku sú uložené samotné termy (*TERM*) a ich termové frekvencie (*TF*). Termy dokumentu sú uložené v lexikografickom poradí, kedy je možné výhodne využiť prefixovú kompresiu.

- *Indexový súbor nad súborom termových frekvencií dokumentu* (ls.idx) – ide o súbor umožňujúci rýchlejší prístup k váham termov v dokumentoch. Pri jeho konštrukcii je využitý fakt, že dokumenty sú jednoznačne identifikované celým číslom *UID*. Odkaz do *súboru termových frekvencií dokumentu* na dáta dokumentu s identifikátorom *UID* sú uložené na pozícii *UID * 4* (veľkosť číselného typu integer).

4.3.4 Fyzická štruktúra

Fyzická štruktúra indexových súborov je silno závislá na type použitého pamäťového média. V prípade systému FRC Finder ide o klasický pevný disk. Keďže vyvinutý vektorový DIS je určený na používanie na bežných osobných počítačoch, kde je fyzická štruktúra disku skrytá za operačný systém, neuvažuje sa o nejakom špeciálnom využití parametrov pevného disku – teda o rozdelení indexových súborov na stopy a cylindre... Takéto rozdelenie, ako aj teoretický rozbor fyzického rozloženia dát sa nachádza mimo rámca práce.

Index systému tvoria klasické binárne súbory. Jednotlivé položky v indexových súboroch sú uložené v komprimovanom tvare, čím sa šetrí diskový priestor. Ide hlavne o ukladanie číselných položiek (Integer, Long) v 7 bitovom komprimovanom tvare a ukladanie série lexikograficky usporiadaných reťazcov pomocou prefixovej kompresie. Pri prefixovej kompresii sa ukladá dĺžka spoločného prefixu a len sufix, ktorou sa dva po sebe idúce reťazce líšia.

4.4 Optimalizácie

Optimalizácie dokumentografických informačných systémov možno podľa prístupu rozdeliť na systémové a technické. Systémové optimalizácie sú nezávislé na spôsobe implementácie dokumentografického systému a sú to skôr algoritmy, postupy a metódy, ktoré umožňujú rýchlejšie a efektívnejšie získanie odpovede na dotaz používateľa. Naopak, technické optimalizácie sú závislé na implementácii a sú to hlavne optimalizácie kódu daného programovacieho jazyka.

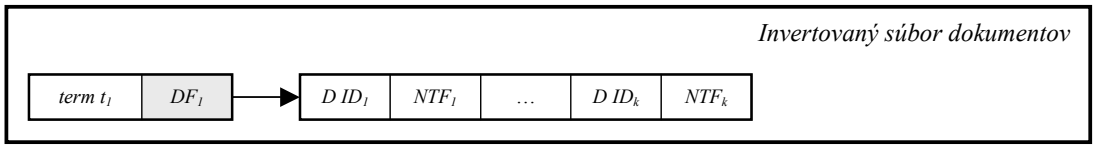
4.4.1 Systémové optimalizácie

Obmedzenie počtu preberaných dokumentov

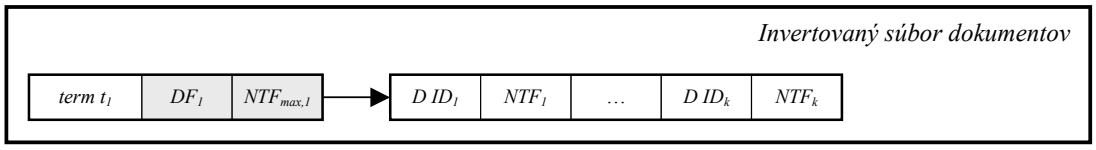
Naším cieľom vyhľadávania je nájdenie *x* dotazu najpodobnejších dokumentov, prípadne nájdenie všetkých dokumentov, ktorých podobnosť je väčšia ako určená hranica sim_{min}. V oboch prípadoch je nutné dokumenty počas výpočtu priebežne usporadúvať podľa ich podobnosti s dotazom. Pre presné vyhľadávanie (vo význame úplného usporiadania dokumentov) v obrovských kolekciách dokumentov je nutné porovnať s dotazom všetky potenciálne vyhovujúce dokumenty, ktorých môže byť hlavne pri dotazoch tvorených všeobecnejšími, nie však až nevýznamovými termami relatívne veľa. Existuje veľmi elegantný, no nie úplne všeobecne použiteľný spôsob, ktorým možno za cenu zníženia presnosti vyhľadávania zmenšiť počet preberaných dokumentov. Tým sa samozrejme zníži aj doba odozvy systému. Ide o využitie čiastočného usporiadania, ktorým je možné získať dokumenty, z ktorých len niektoré (nie nutne všetky) patria do množiny *x* najlepších.

V literatúre ([2D]) možno nájsť popisy niekoľkých heuristických metód (*L-metóda, W-metóda, SW-metóda*), ktoré počítajú podobnosť dokumentu a dotazu, nazývanú aj skóre dokumentu, inkrementálne. Ide o metódy používané hlavne v boolských modeloch DIS, keďže predpokladajú jednotkovú váhu termu v dotaze. Ich myšlienka sa dá s istými obmedzeniami využiť aj pri tvorbe vektorového modelu DIS. Na výpočet podobnosti dokumentu a dotazu sa používa skalárny súčin, ktorý je vplyvom jednotkovej váhy termov v dotaze zdegenerovaný na $\sum_{j=1}^{n} NTF_{i,j} \cdot IDF_j$, kde *n* je počet termov v dotaze a *j* je index termu v dotaze. K výpočtu sa pristupuje od termov dotazu k dokumentom. Postupne sa pre každý term dotazu inkrementuje skóre príslušných potenciálne vyhovujúcich dokumentov o hodnotu $NTF_{i,j} \cdot IDF_j$ (uvažovaný bol *j*-tý term dotazu a *i*-tý dokument). Je prirodzené, že čím rýchlejšie bude rásť skóre dokumentov, tým rýchlejšie a presnejšie bude možné určiť parciálne najdôležitejšie dokumenty. Práve maximalizácia rýchlosti rastu skóre je cieľom spomínaných metód. *L-metóda* určuje poradie spracovania termov dotazu podľa ich rastúcej *DF* (dokumentovej frekvencie), *W-metóda* určuje poradie podľa klesajúcej hodnoty súčinu: $IDF_j \cdot NTF_{\max,j}$, kde $NTF_{\max,j}$ je maximálna normaliovaná váha termu *j* v kolekcii dokumentov a *SW-metóda* navyše rozdeľuje časť invertovaného súboru termov na stránky (diskové bloky), pričom poradie spracovania termov dotazu je určované klesajúcou hodnotou súčinu $IDF_j \cdot NTF_{\max,j}$, kde $NTF_{\max,j}$ je maximálna normalizovaná váho termu *j* v rámci jednej stránky, pričom sa inkrementuje skóre dokumentov z príslušnej stránky – ide teda o „stránkovanú“ verziu *W-metódy*. Všetky tieto metódy dokonca kalkulujú s tým, že termy sú v tomto poradí uložené v súbore invertovaných zoznamov, čím sa šetrí aj počet prístupov na pevný disk. Na nasledujúcej schéme vidieť vždy spôsob uloženia jedného termu s príslušnými termovými frekvenciami.

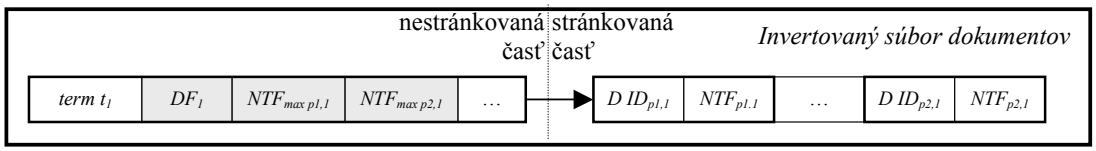
L-metóda



W-metóda



SW-metóda



Po každom kole inkrementácie je k dispozícii isté parciálne usporiadanie dokumentov kolekcie. Po nejakom počte inkrementácií je za určitých podmienok možné vyhlásiť,

že usporiadanie je konečné. Tieto konečné podmienky môžu mať v závislosti na použitej podobnostnej funkcii rôzne matematické vyjadrenie, no všetky majú rovnaký význam. Zabezpečujú, aby prírastok skóre dokumentov v nevykonaných inkrementáciách bol tak malý, aby nespôsobil zmenu, prípadne neovplyvnil poradie, tých najdôležitejších dokumentov.

Táto práca pojednáva o vektorovom DIS, ktorý svojimi indexovými štruktúrami nadväzuje na už existujúci boolský DIS Egothor, takže nie je možné ovplyvniť poradie uloženia termov v indexových súboroch. Je však možné implementovať princípy čiastočného usporiadania s koncovými podmienkami nasledujúcim spôsobom:

- termy dotazu sa usporiadajú a následne prechádzajú podľa zadanej váhy (dôležitosti) zostupne – najviac významné termy budú spracované na začiatku
- dokumenty sa berú postupne z invertovaných zoznamov termov
- skóre dokumentu sa nevytvára inkrementálne, ale naraz – vždy pri prvom výskyte nejakého termu dotazu v dokumente
- po spracovaní každého termu (teda všetkých relevantných dokumentov termu) sa pomocou koncovej podmienky prislúchajúcej k použitej podobnostnej funkcii skontroluje, či má ešte zmysel pokračovať, alebo bolo dosiahnuté konečné usporiadanie.

V prípade použitia podobnostnej funkcie skalárny súčin $sim(\vec{q},\vec{d}_i) = \sum_{j=1}^{m} q_j \cdot w_{i,j}$, je po spracovaní h-tého termu z dotazu maximálny možný inkrement rovný $m-h$. Maximálna váha termu v dotaze aj dokumente je 1. To znamená, že každý súčin $q_j \cdot w_{i,j}$ je rovný nanajvýš 1. Po spracovaní h-tého termu ostáva v dotaze maximálne $m-h$ súčinov. V tomto prípade je m rovné počtu termov v dotaze, takže ak sú hodnoty podobnosti všetkých dokumentov vo výsledku väčšie ako $m-h$, dostali sme úplné usporiadanie dokumentov vo výsledku.
Pre Diceov koeficient je podľa [2] maximálny možný inkrement po spracovaní h-tého termu z dotazu rovný $\frac{2 \cdot (|Q| - h)}{|Q| + \min(|D|)}$.

4.4.2 Technické optimalizácie

Použitie vyrovnávacích pamäťových štruktúr - bufferov (cache)
Typický dôvod na použite vyrovnávacej pamäte je zníženie časovej náročnosti programu (algoritmu) na úkor jeho pamäťovej náročnosti. Zvyčajne sa pomocou vyrovnávacej pamäte nahradzuje opakované vykonávanie časovo alebo zdrojovo náročnej časti programu tak, že po prvom vykonaní príslušnej časti programu sa do vyrovnávacej pamäte vložia výsledky výpočtu spoločne s príslušnými vstupnými parametrami. Pri každom nasledujúcom spustení príslušnej časti programu sa najprv algoritmus pokúsi nájsť výsledok podľa aktuálnych vstupných parametrov vo vyrovnávacej pamäti a až v prípade, že sa tam výsledok nenachádza pustí príslušnú časť programu, ktorá výsledok spočíta. V prípade, že sa výsledok vo vyrovnávacej pamäti už nachádza, je vrátený a nie je potrebné spúšťať časovo náročnú časť programu.

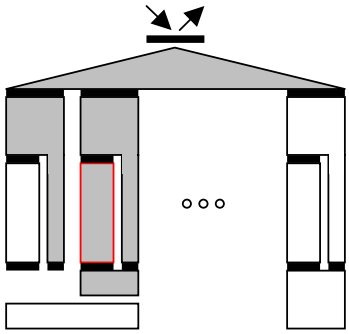
Z uvedeného je jasné, že vyhľadávanie vo vyrovnávacej pamäti sa vykonáva pri každom spustení príslušnej časti programu a z časového hľadiska by malo teda byť oproti samotnému výpočtu zanedbateľné. Zo spôsobu použitia vyplýva, že najčastejšie používané operácie budú dotazy na existenciu danej položky v pamäti a jej získanie. Ako optimálne, a aj najčastejšie používané, pamäťové štruktúry sa javia rôzne typy asociatívnych polí a hashovacích tabuliek.

Termy
Ako už bolo spomenuté, vstupom dokumentografického systému je množina termov, ku ktorým systém vyhľadáva čo najrelevantnejšie dokumenty. Keďže používatelia systému často zadávajú rovnaké alebo aspoň podobné dotazy je výhodné uložiť do medzipamäte aspoň parciálne výsledky prislúchajúce k najčastejšie kladeným termom. Z dôvodu dynamickosti prostredia, z ktorého systém čerpá dokumenty nie je možné ukladať si k termu zoznam všetkých relevantných dokumentov, pretože ten je premenlivý. Do vyrovnávacej pamäte je však možné uložiť odkaz na miesto v indexovom súbore, kde sa príslušný zoznam nachádza (týmto spôsobom je to riešené v systéme Egothor). Týmto spôsobom sa ušetrí niekoľko prístupov na pevný disk – vyhľadanie daného termu v index-sekvenčnom súbore termov a posun a čítanie v dátovom súbore termov. Takéto použitie vyrovnávacej pamäte je veľmi efektívne, pretože termov jazyka je obmedzený počet, čo platí dvojnásobne o najpoužívanejších termoch. Pri vhodne zvolenej veľkosti medzipamäte trafí používateľ systému na term vo vyrovnávacej pamäti s veľkou pravdepodobnosťou.

Dokumenty
Dokumentov, ktoré môže pojať dokumentografický informačný systém môže byť teoreticky nekonečne veľa, preto ich nemá zmysel ukladať do medzipamäte.

5. FRC Finder – DIS vo vektorovom modeli

FRC Finder je dokumentografický informačný systém založený na princípoch vektorového modelu. Je alternatívou k existujúcemu DIS Egothor. Pre potreby tejto práce však nebol naimplementovaný kompletný DIS, pretože zdroj dát, indexové súbory, sú prebraté zo systému Egothor. Implementovaná je teda kompletná vyhľadávacia časť systému a modul, ktorý dopĺňa indexové súbory systému Egothor o údaje potrebné pre prácu DIS vo vektorovom modeli.

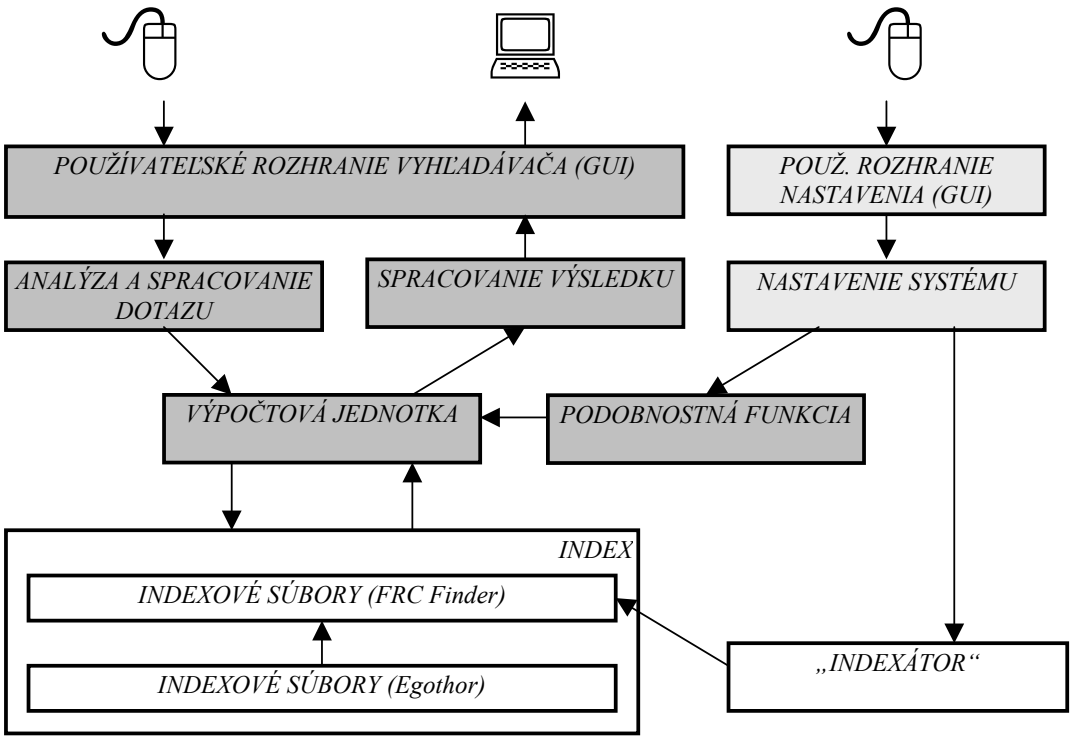


5.1 Architektúra

Pri vytváraní architektúry systému bol braný ohľad na nasledujúce faktory:

- efektívne rozdelenie funkčnosti celého systému do modulov – zabezpečuje, aby mohli byť jednotlivé moduly samostatne vyvíjané a hlavne zameniteľné
- architektúra systému Egothor – dôvodom je hlavne využitie indexových súborov
- maximálna možná robustnosť a konfigurovateľnosť riešenia

Výsledkom analýzy je modulárna architektúra, ktorá schematicky vyzerá takto:



Na schéme dobre vidieť tri skupiny modulov (odlíšené odtieňom pozadia), ktoré plnia základné funkcie DIS: vyhľadávanie (tmavo sivá), nastavenie systému (svetlo sivá) a indexáciu, uloženie a poskytovanie informácií o dokumentoch (biela). Z porovnania schémy architektúry všeobecného DIS v kapitole 2.1 a schémy systému FRC Finder je možné vyčítať aj spomínanú neúplnosť indexačnej jednotky. Tá vôbec nepracuje

s dokumentmi a textami v pôvodnom (zdrojovom) tvare, ale využíva už spracované dáta v súboroch systému Egothor. Doplnenie súborov Egothoru zabezpečuje modul „INDEXÁTOR“. Úvodzovky sú použité zámerne, pretože funkcia, ktorú vykonáva nie je indexáciou v pravom slova zmysle.

5.2 Funkcionalita a implementácia modulov

5.2.1 Indexácia, uloženie a poskytovanie dokumentov

Indexácia

Rozšírenie indexu zabezpečuje modul „INDEXÁTOR“, ktorý je implementovaný triedou `frc_finder.indexer.IndexBarrel`. Algoritmus „indexácie“ je veľmi jednoduchý. Sekvenčne sa prechádza invertovaný súbor termov systému Egothor (ils.dta). V pamäťových štruktúrach sa tak postupne vytvárajú transponované zoznamy – teda ku každému dokumentu vyskytujúcemu sa v invertovanom súbore termov sú priradené termy, ktoré daný dokument reprezentujú. Výsledok sa po prečítaní uloží do súboru termových frekvencií dokumentu (ls.dta) a vytvorí sa nad ním index (ls.idx). Index systému Egothor je rozdelený na menšie časti – barely, pričom každý barel má takú veľkosť, že je ho možné naraz umiestniť do pamäte. Pri procese tvorby transponovaných zoznamov sa rozširuje každý barel osobitne, čo znamená, že proces sa skutočne deje v pamäťových štruktúrach. Dodatočne vytvorené súbory (ls.dta a ls.idx) slúžia na rýchly prístup do indexu „cez dokumenty“, teda vo fáze výpočtu podobnosti dokumentu a dotazu. Presná štruktúra vytvorených súborov je popísaná v kapitole o indexe DIS 4.3.3.

Poskytovanie informácií indexom

Prístup k súborom indexu je dvojvrstvový.

- *Prvá „fyzická“ vrstva* pracujúca priamo s binárnymi súbormi, implementovaná triedami `libs.io.FileIn` a `libs.io.FileOut`, poskytuje možnosť otvorenia a uzavretia súboru a čítanie/zápis primitívnych dátových typov (reťazec, long, integer, bit). Metódy, ktoré čítajú a zapisujú primitívne dátové typy sú vzhľadom na spoločné indexové súbory prebraté zo systému Egothor. Niektoré z nich boli mierne modifikované vzhľadom na odlišnú organizáciu tried systému FRC Finder.
- To je však pre prácu DIS nepostačujúce a hlavne neefektívne, a preto existuje ešte *druhá „logická“ vrstva* poskytujúca operácie s entitami DIS (dokumenty, termy, výskyty termov, metadáta). Funkčnosť logickej vrstvy je rozdelená na prácu s termami (`frc_finder.store.Terms`) a prácu s dokumentmi (`frc_finder.store.Documents`). Logická vrstva skrýva organizáciu a štruktúru súborov, nad ktorými pracuje, a preto je fakticky rozhraním medzi INDEXOM a VÝPOČTOVOU JEDNOTKOU. Z globálnejšieho pohľadu ide dokonca o rozhranie medzi DIS a INDEXOM. Jednotlivé entity systémov Egothor a FRC Finder sú reprezentované a aj používané iným spôsobom, takže implementácia „logickej“ vrstvy nie je prevzatá zo systému Egothor.

Z dôvodu potreby prístupu na pevný disk je poskytovanie informácií indexom časovo najnáročnejšia operácia vyhľadávania. To vytvára priestor na optimalizácie zamerané na zníženie počtu prístupov na pevný disk. Index systému FRC Finder využíva vhodne zvolené dočasné medzipamäte (cache, buffer) na oboch vrstvách prístupu.

- Na prvej „fyzickej“ vrstve sa využíva fakt, že z indexových súborov sa v rámci jedného vyhľadávania číta sekvenčne, napríklad pri čítaní relevantných dokumentov jedného termu, alebo pri čítaní výskytov termu v dokumente.

Podobne je to so zápisom, ktorý je tiež sekvenčný. Triedy `libs.io.FileIn` a `libs.io.FileOut` preto obsahujú medzipamäť, ktorej veľkosť je určená konštantou `libs.Constants.BUFF_IO_SIZE`. Do medzipamäte sa mapuje aktuálna časť súboru. Pri zápise sa všetky zmeny vykonávajú práve v spomínanej medzipamäti a až keď je to potrebné, napríklad požiadavkou na zmenu mimo mapovanej oblasti súboru, sa zmeny uložia na pevný disk a namapuje sa ďalšia časť súboru. Pri čítaní je to podobné – do pamäte sa namapuje vždy časť súboru, z ktorej sa číta sekvenčne, a až keď je potrebné čítať z nenamapovnej oblasti súboru, pristupuje sa na pevný disk a namapuje sa ďalšia oblasť.

- Na úrovni „logickej" vrstvy sa využíva iný prístup. Keďže vrstva pracuje s entitami DIS, je možné do medzipamäte ukladať informácie o celých entitách. Do medzipamäte je však dobré ukladať len entity, pri ktorých je vysoká pravdepodobnosť, že sa ich čítanie bude opakovať. Ide teda o entity, ktorých je obmedzený počet, čo v prípade DIS znamená hlavne termy, pretože slov daného jazyka je len určitý počet, zatiaľ čo dokumentov je potenciálne neobmedzene. Efektívne je teda ukladať informácie o najpoužívanejších termoch, respektíve termoch, na ktoré sa používatelia najviac dotazujú. Vyhľadávanie potom prebieha tak, že pri každom dotaze systém najprv zistí, či žiadaný term nie je v zásobníku a až v prípade, že tam nie je, číta informácie zo súborov na pevnom disku. Otázne je, ktorú zo stratégií výmeny termov v zásobníku použiť. Možností je niekoľko:
  - *LRU – least recently used* – zo zásobníka sú v prípade potreby mazané termy, ktoré boli naposledy použité veľmi dávno
  - *LFU – least frequently used* – zo zásobníka sú v prípade potreby mazané termy, ktoré boli najmenej často použité
  - *SIZE* – zo zásobníka sú v prípade potreby mazané termy, ktorých dáta zaberajú najviac miesta v pamäti (zabezpečuje sa tým čo najväčší počet entít v pamäti)

  Ako najefektívnejšie riešenie sa v danom prípade javí kombinácia metód LRU a LFU, ktorá je použitá aj v implementácii indexu vyhľadávača FRC Finder. Hlavným kritériom výmeny termu v medzipamäti je frekvencia dotazovania sa na daný term, pričom sa frekvencie termov v zásobníku periodicky delia dvoma. Tým sa zohľadní aj časové hľadisko používania termu. Teda termy, ktoré boli v minulosti často používané a v súčasnosti už nie sú používané, sú z medzipamäte vytláčané.

5.2.2 Vyhľadávanie

FRC Finder je DIS vo vektorovom modeli a jeho natívnym formátom dotazu je ohodnotený vektor termov. V prípade systému FRC Finder je použitý rozšírený formát vektora termov, v ktorom má každý term okrem číselného ohodnotenia aj príznak, či je povinný – teda či ho dokument vo výsledku musí obsahovať. Táto drobná zmena významne rozširuje možnosti používateľa pri kladení dotazov (viac o rozšírenom formáte a dôvode pre jeho zavedenie v kapitole 7.2). Dotaz teda vyzerá nasledovne: $[T1:0{,}2:\mathit{true}];[T2:0{,}8:\mathit{false}];[T3:0{,}95:\mathit{false}];[T4:0{,}1:\mathit{true}]$. Takýto dotaz je možné zadať prostredníctvom grafického (`frc_finder.execs.Find_GUI`), prípadne textového rozhrania (`frc_finder.execs.Find_TXT`). Po zadaní je dotaz v textovom formáte predaný modulu ANALÝZA A SPRACOVANIE DOTAZU. Tento je implementovaný triedou `frc_finder.core.find.Query`, ktorá zabezpečuje syntaktickú kontrolu a separáciu jednotlivých častí dotazu (termy, ohodnotenia termov a povinnosti termov) do vnútorných štruktúr. Takto štruktúrovaný dotaz je parametrom VÝPOČTOVEJ JEDNOTKY implementovanej triedou

`frc_finder.core.find.FindBarrel` (názov triedy sa pridržiava „názvoslovia“ systému Egothor, pretože je absolútne výstižný). Ďalšími parametrami výpočtového modulu sú PODOBNOSTNÁ FUNKCIA a inštancia INDEXU. Výpočet prebieha v dvoch fázach:

1. získanie *množiny potenciálne vyhovujúcich dokumentov dotazu* – využíva sa princíp vyhľadávania boolských systémov, kedy sa ku každému termu z dotazu nájde pomocou invertovaného zoznamu termov množina relevantných dokumentov. Na množiny relevantných dokumentov jednotlivých termov sa podľa povinnosti a váhy termu v dotaze aplikujú množinové operácie podľa nasledujúcich pravidiel (algoritmu):
    - termy sa rozdelia na *4 množiny*: NOT – termy, ktorých ohodnotenie je záporné a zároveň sú povinné, AND – termy, ktorých ohodnotenie je kladné a sú povinné, OR – termy, ktorých ohodnotenie kladné a nie sú povinné a REST – zvyšné termy (záporné ohodnotenie a zároveň nepovinné).
    - vytvorí sa množina *pozitívne relevantných dokumentov,* a to nasledovne:
        - ak je množina AND neprázdna, množina pozitívne relevantných dokumentov je *prienikom* množín relevantných dokumentov termov AND množiny
        - ak je množina AND prázdna, množina pozitívne relevantných dokumentov je *zjednotením* množín relevantných dokumentov termov OR množiny
    - vytvorí sa množina *negatívne relevantných dokumentov* ako *zjednotenie* množín relevantných dokumentov termov NOT množiny
    - výsledná množina *relevantných dokumentov dotazu* je množinovým rozdielom množín *pozitívne* a *negatívne relevantných dokumentov*

    Množina relevantných dokumentov dotazu intuitívne obsahuje dokumenty, ktoré sú relevantné k všetkým povinným termom. Ak nie je žiadny term povinný, tak obsahuje všetky potenciálne relevantné dokumenty. Výsledok určite neobsahuje žiadny dokument obsahujúci term, ktorý bol zadaný ako povinný so zápornou hodnotou, no môže obsahovať dokumenty obsahujúce termy, ktoré boli zadané ako nepovinné so zápornou hodnotou.

    Prvá fáza pristupuje do indexu „cez termy“ – k jednotlivým termom sa získavajú ich potenciálne vyhovujúce dokumenty.
2. výpočet podobnosti dotazu a potenciálne vyhovujúcich dokumentov – všetky potenciálne vyhovujúce dokumenty sa ohodnotia použitím PODOBNOSTNEJ FUNKCIE. Implementované sú najpoužívanejšie podobnostné funkcie:
    - skalárny súčin (`frc_finder.sim_function.ScalarProduct`)
    - Diceov koeficient (`frc_finder.sim_function.DicesCoef`)
    - Jaccardova miera (`frc_finder.sim_function.JaccardsCoef`)
    - kosínusová miera (`frc_finder.sim_function.CosineCoef`).

    Ďalšie funkcie je možné doplniť implementovaním rozhrania `frc_finder.sim_function.SimFunction`. Podľa potrieb podobnostnej funkcie sa k ohodnocovanému dokumentu z indexu získavajú váhy popisujúcich termov. Závisí na funkcii, či pre výpočet potrebuje váhy všetkých popisujúcich termov dokumentu, alebo iba niektoré. V princípe sa však v tomto kroku do indexu pristupuje „cez dokumenty“.

Ohodnotené dokumenty sa vložia do výsledku a zoradia zostupne podľa podobnosti s dotazom. Pri vkladaní dokumentov (hitov) do výsledku sa berie do úvahy používateľom nastavená minimálna požadovaná podobnosť dokumentu a dotazu a tiež maximálne povolené množstvo dokumentov na výstupe. Modul SPRACOVANIA VÝSLEDKU je implementovaný triedou `frc_finder.core.find.Result`, ktorej základom je štruktúra halda implementovaná triedou `libs.util.Heap`. Halda je zvolená z toho dôvodu, že pri pridávaní dokumentu do výsledku je vždy potrebné porovnávať hodnotu podobnosti aktuálne pridávaného dokumentu s minimálnou hodnotou podobnosti dokumentu, ktorý je už vo výsledku (odpovedá typickej funkcii haldy `getMin()` s časovou náročnosťou `O(1)`). Halda zároveň poskytuje záverečné zoradenie dokumentov vo výsledku v čase `O(n.log(n))`. Použitá je implementácia haldy v poli.

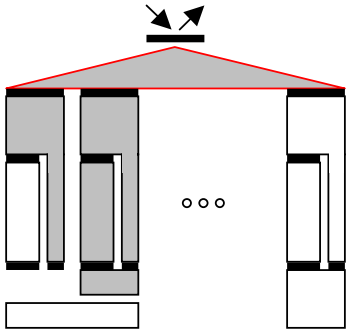
5.2.3 Nastavenie systému

Nastavenie systému dáva používateľovi možnosť ovplyvňovať chovanie systému V prípade DIS FRC Finder je realizované dvoma spôsobmi:

1. *grafickým rozhraním* – menia sa ním vlastnosti systému priamo ovplyvňujúce proces a výsledky vyhľadávania. Ide o nastavenia adresára s indexovými súbormi, zadanie minimálnej požadovanej podobnosti dotazu a dokumentu, maximálny počet dokumentov vo výsledku a pod. Grafické rozhranie je implementované v triede `frc_finder.final_objects.ConvertorV`.
2. *zmenou systémových konštánt* – ide o systémové nastavenia, ktoré sú realizované formou verejných statických konštánt a je ich teda možné meniť len pred vytvorením `*.class` súborov. Ide o nastavenia adresára s logmi, veľkosti vyrovnávacej pamäte ...

6. Modul DISTRIBÚTOR

6.1 Úloha modulu



Primárnou úlohou modulu DISTRIBÚTOR je distribúcia používateľského dotazu medzi všetky systémom zaregistrované DIS a spracovanie výsledkov vyhľadávania zaregistrovaných DIS do jedného uceleného prehľadu. Na prvý pohľad sa môže zdať, že funkcia tohto modulu je jednoznačne vymedzená a jednoducho a „bezducho" implementovateľná. Nie je to však úplne tak. Hlavne porovnávanie a sumarizácia výsledkov vyhľadávania jednotlivých DIS si vyžaduje hlbšiu teoretickú analýzu.

Systém, ktorý využíva výsledky vyhľadávania niekoľkých podriadených DIS sa podľa [1D] nazýva *systém optimálneho vyhľadávania*. Systém optimálneho vyhľadávania je systém zahŕňajúci niektorú z nasledujúcich možností:

- obsahuje viac indexačných algoritmov dokumentov. V tomto prípade systém obsahuje viac indexov, čo neúmerne zvyšuje priestorovú náročnosť.
- obsahuje viac indexačných algoritmov dotazu. Prakticky je to otázka použitia niekoľkých analyzátorov dotazu, prípadne využitia tezauru (využitie synoným termov v dotaze, ktoré zmenšuje problém predikcie).
- obsahuje viac vyhľadávacích algoritmov. Táto možnosť zvyšuje šancu, že niektorý z vyhľadávacích algoritmov bude bližší subjektívnym požiadavkám a potrebám používateľa, čo v konečnom dôsledku zvyšuje presnosť a úplnosť celého systému optimálneho vyhľadávania.

Nespornou výhodou použitia systému optimálneho vyhľadávania, ktorý využíva niekoľko nezávislých podriadených DIS, je minimalizácia problému kritéria predikcie, čím sa zvyšuje použiteľnosť a používateľský komfort celého systému. Problém kritéria predikcie sa minimalizuje v dôsledku diverzifikácie metód indexácie a spôsobu výpočtu jednotlivých DIS. Je evidentné, že každý z podriadených DIS má v svojom indexe uložené iné dáta. Tým sa zvyšuje pravdepodobnosť, že termy zvolené používateľom v dotaze sú zhodné s termami použitými na indexáciu daného dokumentu v niektorom z DIS.

Predvádzaná konfigurácia obsahuje dva DIS, boolský – Egothor a vektorový – FRC Finder, ktoré používajú spoločné indexové súbory. Ide teda o možnosť, kedy systém optimálneho vyhľadávania využíva viac vyhľadávacích algoritmov. Teoreticky by bolo možné, aby systém FRC Finder poskytoval niekoľko sád výsledkov, ktoré by odpovedali použitiu rôznych podobnostných funkcií. V implementácii je však poskytovaný práve jeden výsledok vyhľadávania odpovedajúci aktuálne nastavenej podobnostnej funkcii. Vplyv na minimalizáciu problému kritéria predikcie má teda len použitie rôznych vyhľadávacích algoritmov. Systém je však navrhnutý tak, aby bolo možné zaregistrovať aj úplne nezávislé DIS.

6.2 Distribúcia dotazu

V predloženej implementácii je distribúcia dotazu jednotlivým podriadeným DIS jednovláknová, a teda seriová. Čas odozvy celého systému je teda rovný súčtu časov odozvy jednotlivých DIS navýšený o čas potrebný na spracovanie výsledkov. Ideálne by však bolo, keby bola distribúcia dotazu paralelná. Čas odozvy celého systému optimálneho vyhľadávania by mohol byť teoreticky rovný času odozvy najpomalšieho zaregistrovaného DIS navýšeného o čas potrebný na spracovanie výsledkov.

6.3 Spracovanie výsledkov vyhľadávania

Najväčším problémom porovnávania a sumarizácie výsledkov jednotlivých DIS do jedného celku je fakt, že pri tom istom dotaze budú na výstupe z rôznych DIS rôzne dokumenty s rôznymi mierami podobnosti s dotazom. Dôvodov je hneď niekoľko:

- Rôzne metódy indexácie, a teda rôzny obsah indexov jednotlivých DIS. Nepríjemný je hlavne fakt, že ten istý dokument môže mať v rôznych indexoch inú reprezentáciu. To znamená, že pri jeho indexácii môžu byť v rôznych indexoch použité rozdielne váhy termov, ba dokonca aj rôzne termy.
- Rôzne spôsoby výpočtu podobnosti dokumentu a dotazu. Je zjavné, že spôsob výpočtu podobnosti dotazu a dokumentu výraznou mierou ovplyvňuje výslednú množinu dokumentov. Dokonca aj pri použití rôznych podobnostných funkcií vektorového modelu sú výsledky iné.
- Variabilita výstupov rôznych typov DIS. Princípy fungovania vektorového modelu DIS umožňujú ako výsledok vyhľadávania poskytnúť zoznam dokumentov, ktorý je zoradený podľa miery podobnosti dokumentu s používateľským dotazom. Podobný výsledok možno očakávať aj od DIS v rozšírenom boolskom modeli. Pri použití DIS v čisto boolskom modeli je však situácia iná. Na výstupe možno očakávať len množinu neohodnotených a neusporiadaných dokumentov. Isté vylepšenie umožňuje parciálne usporiadanie dokumentov, ktoré je založené na zaradení dokumentu do istej skupiny dokumentov.
- Iná identifikácia dokumentov. Systém optimálneho vyhľadávania nejakým spôsobom sumarizuje čiastkové výsledky jednotlivých DIS. Aby mohol identifikovať výskyt toho istého dokumentu vo výstupoch viacerých DIS a zohľadniť ho, musí existovať jednoznačný identifikátor dokumentu. Je to názov dokumentu, ISBN, URL, prípadne kombinácia viacerých identifikačných údajov. Pre potreby tejto práce, ktorá sa zaoberá vyhľadávaním v sieti internet bude jednoznačným identifikátorom dokladu jeho URL. O čosi zložitejší kľúč by bolo potrebné voliť v prostredí reálnych tlačených dokumentov (napr. knižnice).

V konečnom dôsledku ide v module DISTRIBÚTOR o zvýhodnenie, prípadne potlačenie niektorej vyhľadávacej metódy (celého DIS) voči ostatným podľa potrieb používateľa. Ide teda o nájdenie ideálnej kombinácie dostupných metód vyhľadávania na ponúknutom výsledku. To však nie je možné bez aktívnej interakcie s používateľom.

6.3.1 Odvodenie výberu optimálnej metódy – teoretický rozbor

Nech má systém optimálneho vyhľadávania k dispozícii k rôznych DIS (metód vyhľadávania). Nech i-tý DIS vrátil vo svojom výsledku n_i dokumentov, z ktorých r_i je pre daný používateľský dotaz relevantných. Nech celkový počet relevantných

dokumentov na používateľský dotaz je *r*. Celkový počet dokumentov je *n*, no nie je pre odvodenie vhodnosti metódy podstatný.

Nech $\vec{X} = (x_1, x_2, ..., x_n)$, kde x_j je rovné 1 ak *j*-tý dokument je relevantný, inak 0 – je to vektor relevantných dokumentov celej kolekcie. Nech $\vec{Y}_i = (y_{i,1}, y_{i,2}, ..., y_{i,n})$, kde $y_{i,j}$ je rovné 1 ak *j*-tý dokument vrátený vo výsledku *i*-tého DIS je relevantný, inak 0 – je to vektor relevantných dokumentov *i*-tého DIS. Na vyjadrenie vhodnosti použitej metódy je možné použiť štandardnú podobnostnú funkciu vektorového modelu DIS: $Sim(\vec{Y}_i, \vec{X}) = \vec{Y}_i \cdot \vec{X} = \|\vec{Y}_i\| \cdot \|\vec{X}\| \cdot \cos\varphi$. Z toto vzťahu vyplýva: $\cos\varphi = \frac{\vec{Y}_i \cdot \vec{X}}{\|\vec{Y}_i\| \cdot \|\vec{X}\|}$, kde $\vec{Y}_i \cdot \vec{X} = \sum_{j=1}^{k} y_{i,j} \cdot x_j = r_i$, $\|\vec{Y}_i\| = \sqrt{n_i}$ a $\|\vec{X}\| = \sqrt{r}$. Po dosadení za príslušné premenné: $\cos\varphi = \frac{r_i}{\sqrt{n_i} \cdot \sqrt{r}} = \sqrt{\frac{r_i \cdot r_i}{n_i \cdot r}} = \sqrt{\frac{r_i}{n_i} \cdot \frac{r_i}{r}} = \sqrt{P_i \cdot R_i}$. Z uvedených vzťahov vyplýva skutočnosť, že miera vhodnosti daného DIS odpovedá výrazu $\sqrt{P_i \cdot R_i}$.

Ide o súčin presnosti (P – precision = pravdepodobnosť, že dokument zaradený v odpovedi je relevantný) a úplnosti (R – recall = pravdepodobnosť, že relevantný dokument je zaradený v odpovedi) daného DIS. Okrem iného to znamená, že DIS s vysokou presnosťou a úplnosťou budú v rámci systému optimálneho vyhľadávania uprednostňované. Ideálny prípad nastáva ak *P=R=1,* a teda *P.R=1*. V ideálnom prípade sú v odpovedi práve všetky relevantné dokumenty z kolekcie (nie sú tam navyše žiadne nerelevantné). Pre reálne DIS však platí, že *P.R=const<1*. Je to dôsledok toho, že pri snahe zvýšiť presnosť *P* sa na výstup nedostanú všetky relevantné dokumenty, čím sa znižuje úplnosť *R*. Naopak, pri snahe zvýšiť úplnosť *R* sa na výstup dostane viac relevantných dokumentov, no aj viac nerelevantných dokumentov, čo spôsobuje zníženie presnosti *P*.

Na rozhodnutie o vhodnosti DIS pre daného používateľa však nie je potrebné poznať jeho charakteristiky *P* a *R*. Keďže vo výraze $\sqrt{P_i \cdot R_i} = \sqrt{\frac{r_i}{n_i} \cdot \frac{r_i}{r}}$ je *r* konštanta a odmocnina je rastúca funkcia, je pre vhodnosť DIS pre používateľa rozhodujúci výraz $\frac{r_i^2}{n_i}$, kde n_i je celkový počet vyhovujúcich dokumentov vrátených *i*-tým DIS a r_i je počet relevantných dokumentov vrátených *i*-tým DIS.

6.3.2 Postup získania optimálnej metódy v praxi

Je evidentné a prirodzené, že výber vhodnej kombinácie metód vyhľadávania, prípadne celých DIS je subjektívny pre každého používateľa. Práve preto musí používateľ do výberu nejako vstupovať.

Pri prvom použití systému sa dotaz vyhodnotí všetkými dostupnými DIS a výsledok sa spojí do jedného výsledku (napríklad jednoduchým zjednotením výsledkových množín). Zoradenie dokumentov je v tomto okamihu irelevantné. Používateľ označí dokumenty, ktoré považuje za relevantné, čím oznámi systému r_i pre všetky použité DIS (n_i systém pozná). Systém môže teda vyhodnotiť vhodnosť všetkých DIS podľa

teoreticky odvodeného kritéria $\frac{r_i^2}{n_i}$. Najviac vhodné DIS budú v odpovediach zvýhodňované, menej vhodné DIS znevýhodňované. Takýmto spôsobom je možné vytvoriť rôzne „používateľské profily“. Profil fakticky pozostáva z *koeficientov vhodnosti* (značí sa λ) jednotlivých DIS. Koeficient vhodnosti λ_i je reálne číslo z intervalu $\langle 0,1 \rangle$, ktorým sa násobia podobnosti dokumentu a dotazu vo výsledku *i*-tého DIS. Najvhodnejšie DIS pre daného používateľa majú hodnoty blízko hodnoty 1 a tie nevhodné sú zo spodnej časti intervalu. Jednotlivé koeficienty λ_i môžu byť odvodené napríklad vzťahom $\lambda_i = \frac{r_i^2}{n_i} \Big/ \frac{r_{\max}^2}{n_{\max}}$, ktorý zabezpečí, že najlepšia metóda nebude znevýhodnená $\lambda_{\max} = 1$ a koeficienty ostatných budú menšie ako 1, čím budú znevýhodnené. Profil používateľa je možné vďaka jednoduchosti koeficientov vhodnosti ukladať na pevný disk a kedykoľvek ho používateľovi ponúknuť.

Spôsob popísaný v predchádzajúcom odstavci vedie k vytvoreniu profilu, ktorý optimálne odpovedá potrebám používateľa. Spôsob vytvorenia profilu je navyše nenáročný a odtieňuje používateľa od princípov fungovania systému a jeho technických detailov. Na druhú stranu umožňuje vytvárať profily využívajúce len jeden konkrétny DIS, a tým využiť jeho prednosti (vhodné je to napríklad v prípade, keď je dotaz v tvare konjunktívnej disjunkcie – vo vektorovom modeli DIS je to neinterpretovateľný dotaz, a preto je vhodné použiť DIS v boolskom modeli).

6.3.3 Spracovanie výsledkov jednotlivých DIS

Ako už bolo spomenuté v úvode kapitoly 6.3, pri spracovaní výsledkov vyhľadávania jednotlivých DIS do uceleného zoznamu je potrebné prekonať niekoľko problémov a nezrovnalostí.

Normalizácia výsledkov DIS

Všeobecne možno predpokladať, že výsledok vyhľadávania DIS je množina dokumentov, ktoré sú ohodnotené mierou podobnosti s dotazom. Istý problém nastáva pri použití DIS v boolskom modeli, kedy nie je možné ohodnotiť, do akej miery vyhovuje dokument zvolenému dotazu – buď vyhovuje alebo nevyhovuje. V tomto prípade bude miera podobnosti dotazu a dokumentov jednotná (napr. 1). Rôzne DIS môžu dávať miery podobnosti z iných intervalov: nech *i*-tý DIS dáva hodnoty z „lokálneho“ intervalu $\langle l_{i,1}, l_{i,2} \rangle$. Aby nebol zvýhodnený žiadny z DIS, je nutné prevedenie výsledkov z „lokálnych“ intervalov do hodnôt zo spoločného „globálneho“ intervalu $\langle g_1, g_2 \rangle$. Proces prevodu sa nazýva *normalizácia*. Hodnoty sa prevádzajú lineárne podľa vzťahu: $G_{i,j} = (L_{i,j} - l_{i,1}) \cdot ((g_2 - g_1) / (l_{i,2} - l_{i,1})) + g_1$ (hodnota podobnosti *j*-tého dokumentu s dotazom $L_{i,j}$ z lokálneho intervalu *i*-tého DIS bola prevedená na *j*-tú hodnotu $G_{i,j}$ z globálneho intervalu). Obvykle sa interval $\langle g_1, g_2 \rangle$ berie $\langle 0,1 \rangle$. Takto sa získajú normalizované hodnoty bez zohľadnenia vhodnosti jednotlivých DIS pre daného používateľa. Po zohľadnení koeficientov vhodnosti λ_i jednotlivých DIS sa výsledky jednotlivých DIS nachádzajú v intervaloch $\langle g_1, g_1 + \lambda_i \cdot (g_2 - g_1) \rangle$.

Ďalšie úpravy
Jeden dokument sa môže vyskytovať vo výsledkoch rôznych DIS, pričom v každom bude mať po normalizácii pravdepodobne inú globálnu mieru podobnosti s dotazom. S rastúcim počtom výskytov dokumentu v rôznych výsledkoch DIS rastie aj pravdepodobnosť, že je dokument k dotazu relevantný. Matematicky sa však jednoduchšie interpretuje ekvivalentné tvrdenie, že s rastúcim počtom výskytov dokumentu klesá pravdepodobnosť, že je dokument nerelevantný. Tento fakt interpretuje vzťah $G_j = 1 - \prod_{i=1}^{k} (1 - G_{i,j})$, kde $G_{i,j}$ je podobnosť *j*-tého dokumentu s dotazom vrátená *i*-tým DIS po normalizácii a zohľadnení koeficientu vhodnosti λ_i. G_j je teda výsledná miera podobnosti *j*-tého dokumentu s dotazom, ktorá je finálne poskytnutá používateľovi po zohľadnení všetkých faktorov.

6.4 Implementácia

Modul DISTRIBÚTOR je v svojej podstate jadrom celého systému optimálneho vyhľadávania. Z tohto dôvodu by mala byť jeho implementácia dynamická v zmysle možnosti pridania (zaregistrovania), prípadne odobratia niektorého DIS za behu systému. Celý modul je implementovaný v jazyku Java, ktorý poskytuje dynamické pripájanie zásuvných modulov – plugins. Jednotlivé podriadené DIS je teda možné dynamicky spúšťať prostredníctvom tzv. Java Reflection API (viď napr [2J]). Každý DIS však musí spĺňať (implementovať) verejné rozhranie `DISIface` definované v balíčku `frc_distributor.intrfaces`.

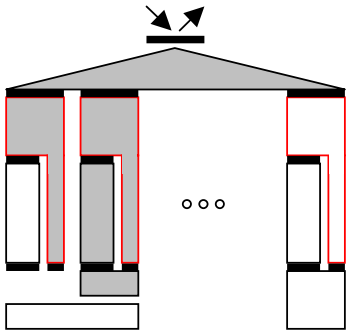
Implementácia modulu DISTRIBÚTOR je grafická a nachádza sa v triede `frc_distribuor.execs.Distributor_GUI`. Všetky operácie, hlavne zadávanie dotazov v rôznych formátoch a správu jednotlivých podriadených DIS, je teda možné vykonávať prostredníctvom grafického rozhrania.

Distribúcia dotazov jednotlivým DIS je (v tejto verzii) zabezpečovaná sériovo v jednom vlákne. Je však možné (a aj efektívnejšie) implementovať systém distribúcie dotazu podobným spôsobom ako to robí systém Egothor pri distribúcii dotazu jednotlivým častiam indexu (tzv. Barelom). Distribúciu tam zabezpečuje špeciálny modul `Distributor` (`balík org.egothor.distributor`), ktorý má dve rôzne implemenácie – multivláknovú (trieda `MTE`) a jednovláknovú (trieda `STE`). Prirodzene, multivláknová implementácia prináša isté zrýchlenie celého systému a to dokonca aj na jednoprocesorových počítačoch. Dôvodom sú prístupy DIS na pevný disk (kým jeden DIS čaká na prístup na pevný disk, iný zatiaľ počíta podobnosti dotazu a dokumentov...).

Teoreticky by dokonca mohli jednotlivé DIS bežať na rôznych počítačoch a celý systém by sa tak stal distribuovaný.

Na zlučovanie výsledkov jednotlivých DIS je použitý algoritmus popísaný v teoretickej časti (kapitola 6.3.3). Všetky potrebné metódy sú implementované v triede `libs.core.Result`. Podobne je aj tvorba multiplikačných konštánt – koeficientov vhodnosti jednotlivých DIS implementáciou algoritmu, ktorý je popísaný v teoretickej časti (kapitola 6.3.1 a 6.3.2).

7. Modul KONVERTOR



V systéme optimálneho vyhľadávania môžu vystupovať DIS rôznych vlastností, špecializácií a modelov. Používateľ systému však štandardne špecifikuje svoj dotaz vo formáte, ktorý najviac vyhovuje jeho požiadavkám a potrebám. Je zrejmé, že tento formát dotazu nemusí vyhovovať každému z DIS. Dobrým príkladom je nemožnosť vhodne reprezentovať typický boolský dotaz T_1 *AND* (T_2 *OR* T_3) *AND* T_4 (kde T_i sú termy) dotazom vo vektorovom modeli DIS. Práve na preklenutie rozdielov medzi používateľským dotazom a reprezentáciou dotazu, ktorá je prirodzená danému modelu (typu) DIS slúži modul KOVERTOR.

Systém optimálneho vyhľadávania podporuje 4 základné spôsoby dotazovania:

- **(ohodnotená) logická formula termov** – typický spôsob dotazovania na DIS v rozšírenom boolskom modeli. V podstate ide o obyčajnú formulu zloženú z termov, akurát sú jednotlivé atómy (termy) ohodnotené číselnou váhou, ktorá vyjadruje mieru dôležitosti termu pre používateľa. Všetky váhy a ohodnotenia termov dotazu sú štandardne z intervalu $\langle 0,1 \rangle$.

  Formát dotazu: $([0{,}2 : T1] \; OR \; [0{,}8 : T2]) \; AND \; [0{,}95 : T3] \; AND \; [0{,}1 : T4]$

- **ohodnotený vektor termov** – spôsob vhodný pre vektorový model DIS. Dotazom je ohodnotená množina (vektor) termov. Váhy termov dotazu sú na rozdiel od ohodnotenej formuly z intervalu $\langle -1,1 \rangle$, kde záporné hodnoty vyjadrujú fakt, že dokumenty s takýmto termom by sa vo výsledku nemali vyskytovať. V praxi podobnostných funkcií to však znamená, že dokumenty obsahujúce daný term sú znevýhodňované, takže vo výsledku sa môžu objaviť. Je to teda istá „mäkšia“ analógia boolského NOT.

  Formát dotazu: $[0{,}2 : T1]; [0{,}8 : T2]; [0{,}95 : T3]; [0{,}1 : T4]$

- **odkaz na už zaindexovaný dokument** – dotazom je odkaz na dokument, ktorý už prešiel procesom indexácie a DIS ho má uložený v svojom indexe. V systéme optimálneho vyhľadávania však vystupuje niekoľko nezávislých DIS, čo znamená, že ten požadovaný dokument nemusí byť zaindexovaný vo všetkých DIS. K vzniknutej situácii sa dá pristúpiť z dvoch strán. Nejakým spôsobom donútiť DIS, ktoré tento dokument nemajú zaindexovaný k jeho indexácii, alebo brať do úvahy len výsledky DIS, ktoré ho zaindexovaný majú. Navyše do popredia vystupuje otázka jednoznačnej identifikácie dokumentu. V kapitole o module DISTRIBÚTOR (6.3) je uvedený ako jednoznačný identifikátor dokumentu jeho URL, čo je jedno z metadát dokumentu.

  Niektoré DIS (viď Egothor) však majú indexové súbory organizované tak, že k metadátam dokumentu sa dá efektívne pristupovať len cez vnútorný technický identifikátor dokumentu. Je to danosť systému, ktorý sa snaží o minimalizáciu priestorovej náročnosti a prispôsobenie indexových súborov funkcii, na ktorú sú primárne určené. Riešením je zmena organizácie indexových súborov, prípadne vytvorenie indexu nad položkou URL v súbore metadát. Tento problém je tu

spomenutý vzhľadom na kontext diplomovej práce, ktorá naväzuje na existujúci DIS. V prvej verzii bude systém optimálneho vyhľadávania využívať výsledky práve spomenutého DIS Egothor a novovyvinutého DIS FRC Finder. Je samozrejmé, že všetky ďalšie DIS zapojené do systému optimálneho vyhľadávania musia spĺňať definovaný interface.

Zadanie dotazu odkazom na zaindexovaný dokument je možné rozšíriť. Jednak je možné zadať ako dotaz viac dokumentov a navyše ich ohodnotiť váhou. Výsledok by odpovedal predstave, že používateľ chce nájsť napríklad všetky dokumenty, ktoré sa podobajú dokumentu *D1* na 20% a dokumentu *D2* na 90% (myslené z hľadiska použitých termov, nie samotného obsahu a významu dukumentu). Tento prístup by mohol znížiť problém predikcie, pretože rôzni autori popisujú rovnakú problematiku rôznymi termami.
Formát dotazu: $[www.google.com : 0{,}9];[www.altavista.com : 0{,}4]$

- **ukážka (časť) textu** – pre používateľa nenáročný spôsob umožňujúci zadať ako dotaz nejakú časť textu (najlepšie Copy & Paste z existujúceho zdrojového dokumentu). DIS takýto text najprv spracuje indexačným algoritmom. Tým sa získa množina termov použitých v texte spolu s mierami významnosti použitých termov. Takto spracovaný dotaz sa dá jednoducho previesť na ohodnotenú logickú formulu (bude to disjunkcia vážených termov), prípadne na ohodnotený vektor termov. Tento spôsob zadania dotazu je svojou podstatou vhodný pre všetky modely DIS. Dotaz zadaný priamo textom sa používa hlavne v situáciách, keď chce používateľ získať dokumenty podobné nejakému zdrojovému dokumentu (zároveň sa predpokladá, že zdrojový dokument nebol, resp. nemusel byť zaindexovaný).
  Formát dotazu: $T1\ T2\ T3\ T1\ T4\ T5\ T3\ T1\ T6\ T7\ T2\ T8\ T9$

Pri zadaní dotazu časťou textu dokumentu, prípadne odkazom na zaindexovaný dokument, príde vhod tesnejšie naviazanie modulu KONVERTOR na DIS (v praxi to znamená širší interface s viacerými funkciami). Dôvod je jednoduchý. V oboch prípadoch zadania dotazu je na zostavenie finálneho dotazu potrebné aplikovať niektorý z algoritmov implementovaných v samotnom DIS – indexáciu textu a vyhľadanie dokumentu podľa jeho URL.

7.1 Konverzia dotazu: „vektorový“ -> „boolský“

Odpoveď DIS vo vektorovom modeli je závislá na použitej podobnostnej funkcii. Bežne používané podobnostné funkcie (skalárny súčin, Diceov koeficient, kosínusová miera...) však majú jednu vlastnosť spoločnú. Na to, aby bola podobnosť dokumentu a dotazu nenulová postačuje existencia jedného spoločného termu s nenulovou váhou v dotaze aj dokumente. Stačí si uvedomiť, že to môže byť ktorýkoľvek term dotazu a je jasné, že dotaz typický pre vektorový model DIS – ohodnotený vektor termov – je možné svojou podstatou chápať ako disjunkciu ohodnotených termov, čo je forma vhodná pre DIS v boolskom modeli. Koverzia spočíva v prevedení vektora termov na disjunkciu termov.

7.2 Konverzia dotazu: „boolský“ -> „vektorový“

Ohodnotený vektor termov obsahuje len množinu termov a ohodnotenie ich váh. Naproti tomu, ohodnotená logická formula termov má v sebe obsiahnutú aj informáciu o vzájomných vzťahoch medzi jednotlivými termami. Dôsledkom je fakt,

že do ohodnoteného vektora termov môže oproti ohodnotenej logickej formuli používateľ vložiť menej informácií. Preto nie je možné navrhnúť bezstratovú prevodovú funkciu, ktorá by každému dotazu v boolskom modeli DIS (ohodnotená formula termov) jednoznačne priradila dotaz vektorového DIS (ohodnotený vektor termov). Exaktne vyjadrené: neexistuje bijekcia medzi množinou dotazov boolského a vektorového modelu DIS (existuje len zobrazenie - surjekcia, ktoré je na celú množinu dotazov vektorového modelu). Aby však používateľ systému nebol konfrontovaný s nesprávnymi výsledkami vyhľadávania je nutné navrhnúť spôsob zadania dotazu v „boolskom tvare“ a jeho konverziu do „vektorového tvaru“.

Kontrola dokumentov na výstupe
Metóda predpokladá, že okrem štandardného výsledku vyhľadávania (dokument, URL, podobnosť s dotazom, kontextová ukážka) bude DIS poskytovať modulu KONVERTOR aj množinu termov, ktoré majú v dokumente nenulovú váhu. Pre každý dokument túto množinu termov modul KONVERTOR porovná so zadanou logickou formulou a vyhodnotí jej platnosť. Na celkový výstup sa dostanú len dokumenty spĺňajúce požiadavky používateľa. Popísaná metóda je síce korektná, no predpokladá existenciu špeciálneho výstupu (termy dokumentu), čo môže byť pri použití už existujúceho DIS nedosiahnuteľné. Navyše metóda mierne spomaľuje odozvu systému.

Alternatívou môže byť využitie tesnejšieho naviazania modulu KONVERTOR na samotný DIS, ktoré spočíva v možnosti modulu KONVERTOR priamo alebo nepriamo (prostredníctvom DIS) čítať indexové súbory a získavať z nich potrebné informácie. Samotný DIS by v tomto prípade nemusel vo výsledku vyhľadávania poskytovať s každým dokumentom aj zoznam relevantných termov. Modul KONVERTOR by si tieto termy mohol prečítať z indexových súborov sám. Touto metódou sa zmenší množstvo dát tečúcich cez rozhranie DIS-KONVERTOR. Na druhú stranu sa zvýši zložitosť konvergenčného modulu (čo je nežiadúce). V podstate ide stále o znovuspracovanie výsledkov DIS (v zahraničnej literatúre nazývané post-processing). Omnoho výhodnejšie je začleniť kontrolu splnenia podmienky dotazu už do spracovania dotazu.

Overenie podmienky dotazu pri spracovaní dokumentu
Metóda počíta s tým, že dotaz (ohodnotený vektor termov) bude rozšírený o jeden boolovský (bitový) atribút pre každý term. Pridaný boolovský atribút vypovedá o tom, či je term povinný – inými slovami, či dokument, ktorý bude poskytnutý na výstupe musí obsahovať daný term. Formát dotazu bude teda nasledovný: $[T1:0{,}2:true];[T2:0{,}8:false];[T3:0{,}95:false];[T4:0{,}1:true]$. Teraz si už len stačí uvedomiť, že vektor termov, kde žiadny z termov dotazu nie je povinný odpovedá disjunkcii termov a vektor, kde sú všetky termy dotazu povinné odpovedá konjunkcii termov. Formálne zapísané (spolu s váhami):

$$[T1:0{,}2:true];[T2:0{,}8:true];[T3:0{,}95:true] \approx [T1:0{,}2]\,AND\,[T2:0{,}8]\,AND\,[T3:0{,}95]$$

$$[T1:0{,}2:false];[T2:0{,}8:false];[T3:0{,}95:false] \approx [T1:0{,}2]\,OR\,[T2:0{,}8]\,OR\,[T3:0{,}95]$$

Každú logickú formulu je možné podľa [4D] previesť na disjunktívne konjunktívny tvar (DKT), teda na disjunkciu konjunkcií termov. To dopomôže k prevodu akéhokoľvek boolského dotazu na niekoľko rozšírených vektorov termov. Boolský dotaz – formula termov sa prevedie do DKT. Každá jedna konjunkcia termov odpovedá jednému rozšírenému vektoru termov, kde sú všetky termy povinné. Počet

rozšírených vektorov termov odpovedá počtu disjunkcií v DKT. Na výstup sa potom dostanú tie dokumenty, ktoré spĺňajú aspoň jednu konjunkciu v DKT. Prakticky to znamená, že jeden dotaz v boolskom modeli DIS odpovedá niekoľkým dotazom vektorového modelu DIS. Sumarizácia výsledkov niekoľkých dotazov je úlohou modulu KONVERTOR. Pre úplnosť teórie sa ešte treba vysporiadať s logickou spojkou NOT. Štandardne sa vo vektorových DIS zohľadňuje spojka NOT zápornou hodnotou váhy termu v dotaze. To však zvyčajne len znevýhodňuje dokumenty, ktoré obsahujú daný term. V žiadnom prípade záporná hodnota váhy termu nezaručuje, že sa na výstupe žiadny dokument s príslušným termom neobjaví. V rozšírenom vektorovom dotaze je však možné rozlíšiť „tvrdé“ a „mäkké“ NOT. „Mäkké“ NOT odpovedá chápaniu negácie klasickým vektorovým modelom, zatiaľ čo „tvrdé“ NOT znamená, že dokument obsahujúci daný term sa na výstupe neobjaví.

Pri rozklade boolskej formuly do DKT môžu vo výsledku vzniknúť dva typy rozšírených vektorových dotazov. Jeden, kde budú všetky termy povinné, čo odpovedá jednej konjunkcii v DKT a druhý, kde budú všetky termy nepovinné, čo odpovedá čisto disjunktívnemu boolskému dotazu. Znamená to, že atribút rozširujúci vektor termov nie je potrebné špecifikovať pre každý z termov dotazu samostatne, ale stačí spoločne pre celý vektor. Pre možnosť vytvárať priamo vektorové dotazy, kde bude niektorý z termov povinný je atribút ponechaný na úrovni termov dotazu.

Podmienkou fungovania načrtnutej teórie v praxi je akceptácia rozšíreného dotazu DIS vo vektorovom modeli. Implementovaný DIS FRC Finder túto možnosť podporuje. Implementácia postupu je jednoduchá, ak je použitý spôsob výpočtu popísaný v kapitole o indexových súboroch vektorového DIS (4.3.2). Ten v prvom kroku výpočtu vytvára množinu relevantných dokumentov, pre ktoré sa bude vyhodnocovať podobnosť s dotazom. V štandardnom pojatí dotazu žiadny z termov nie je povinný, čo odpovedá boolskému OR dotazu. Množina relevantných dokumentov dotazu je teda zjednotením množín relevantných dokumentov jednotlivých termov dotazu. Ak sú všetky termy povinné, množina relevantných dokumentov dotazu odpovedá prieniku množín relevantných dokumentov jednotlivých termov. „Mäkká“ negácia, nemá vplyv na výslednú množinu, zatiaľ čo „tvrdá“ negácia znamená použitie množinového rozdielu. T.j. z množiny relevantných dokumentov celého dotazu sa odoberú tie dokumenty, ktoré sú relevantné pre negovaný term. Implementácia použitia rozšíreného vektora je založená na použití základných množinových operácií nad množinami relevantných dokumentov jednotlivých termov.

Popísaný postup zabezpečuje konverziu akéhokoľvek zmysluplného boolského dotazu na sériu vektorových dotazov, pričom väčšina práce systému sa vykoná počas vyhľadávania. Zvýšené nároky sú kladené aj na modul KONVERTOR, ktorý pripravuje konverziu boolskej formule do disjunktívne konjunktívneho tvaru a finálne zlúčenie výsledkov jednotlivých čiastočných dotazov zo série. Zlúčenie je pomerne jednoduchá záležitosť, pretože ide iba o zjednotenie množín výsledkov jednotlivých čiastočných vyhľadávaní. V prípade, že je ten istý dokument vo viacerých výsledkoch, vezme sa výskyt s maximálnou podobnosťou s dotazom.

7.3 Implementácia

V práci boli implementované dva moduly KONVERTOR. Jeden v rámci budovania systému FRC Finder (`frc_finder.final_objects.ConvertorV`) a druhý ako

„nadstavba“ existujúceho systému Egothor, ktorá spĺňa rozhranie `frc_distributor.interfaces.DISIface` a tým zabezpečuje pripojitelnosť systému Egothor do systému optimálneho vyhľadávania (`org.egothor.final_objects.ConvertorB`). Oba moduly fakticky implementujú teóriu rozpracovanú v predchádzajúcich kapitolách.

7.3.1 Konvertor pre vektorový DIS – FRC Finder

Úlohou konvertora pre vektorový systém FRC Finder je konverzia všetkých typov dotazov do natívneho vektorového tvaru – rozšíreného vektora ohodnotených termov. Implementačne a algoritmicky najzaujímavejšia je konverzia z boolského na vektorový tvar.

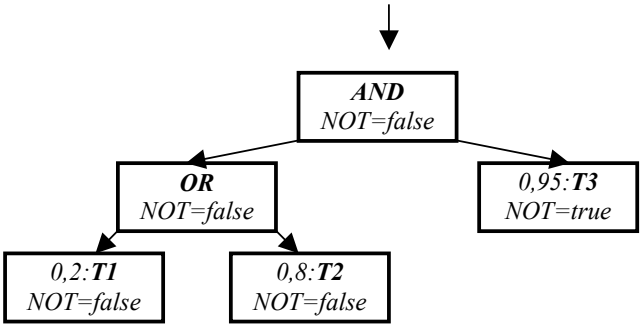
Konverzia dotazu z boolského na vektorový tvar

Tá má v podstate tri fázy. Prvá fáza zabezpečuje kontrolu dotazu (logickej formuly termov) a jeho prevod do vnútorných štruktúr, druhá fáza pokrýva prevod formuly na disjunktívne konjunktívny tvar a tretia fáza zabezpečuje vytvorenie množiny ohodnotených vektorov termov.

- *Prvá fáza algoritmu prevodu* - Vstupom do algoritmu je ohodnotená logická formula termov (napríklad $(0{,}2 : T1\ OR\ 0{,}8 : T2)\ AND\ NOT(0{,}95 : T3)$). V prvom rade je formula rozdelená na jednotlivé slová, kde oddeľovačmi slov sú medzera, ľavá zátvorka a pravá zátvorka (aj zátvorky sú slová). Po rozdelení dotazu na slová sa lineárnym prechodom skontroluje, či je počet ľavých a pravých zátvoriek totožný a či sú zmysluplne usporiadané. Navyše sú jednotlivým slovám priradené čísla vypovedajúce o hĺbke zanorenia slova v zátvorkách. Z takto predspracovaného dotazu je už možné vytvárať stromovú štruktúru jednoznačne reprezentujúcu formulu termov.

  Vnútorná stromová štruktúra je vlastne binárny strom, ktorého listy sú ohodnotené termy a vnútorné uzly reprezentujú logické spojky AND a OR. Všetky vnútorné uzly majú dvoch potomkov, ktorí reprezentujú ľavú a pravú podformulu príslušnej spojky. Všetky elementy stromu (listy aj vnútorné uzly) majú príznak NOT, ktorý hovorí o tom, že na term, prípadne na podstrom reprezentovaný logickou spojkou je aplikované logické NOT. Napríklad reprezentácia logickej formuly uvedenej v predchádzajúcom odstavci vyzerá nasledovne:

$$(0{,}2 : T1\ OR\ 0{,}8 : T2)\ AND\ NOT(0{,}95 : T3)$$



  Táto stromová štruktúra je implementovaná triedami `BQueryElement`, `BQueryTerm`, `BQueryORForm` a `BQueryANDForm` v balíčku `frc_finder.core.convertor_b2t`. Triedy tvoria objektovú hierarchiu, ktorej základom je abstraktná trieda `BQueryElement`. Binárny strom je vytváraný

rekurzívnym volaním metódy `BQueryElement` ***`makeStructure`***`(int word_from, int word_to, boolean apply_not)`. Metóda pracuje nad dotazom rozdeleným na slová, presnejšie nad slovami dotazu od pozície určenej parametrom *word_from* po pozíciu určenú parametrom *word_to*. Návratovou hodnotou je referencia na stromovú štruktúru, ktorá reprezentuje časť dotazu určenú parametrami. V prípade chybnej syntaxe dotazu je vrátená hodnota *null*.

- Pri svojom výpočte metóda najprv hľadá niektorú z binárnych spojok AND, OR alebo NOT a to medzi slovami s čo najmenším zanorením v zátvorkách. Vzhľadom na vzájomnú hierarchiu spojok, (bez použitia zátvoriek je spojka NOT najprioritnejšia, potom nasleduje spojka AND a nakoniec OR) je pri voľbe koreňa príslušného podstromu uprednostňovaná spojka OR, potom AND a nakoniec NOT – docieli sa tým, že prioritnejšie spojky budú v stromovej hierarchii nižšie ako menej prioritné. V prípade, že koreňom je niektorá binárna spojka (OR, AND), je metóda ***`makeStructure`***`()` raz zavolaná na slová vľavo od pozície nájdenej spojky (s reálnymi parametrami *word_from*, *pos-1*, *false*) a druhý krát na slová vpravo od nájdenej spojky (s reálnymi parametrami *pos+1*, *word_to*, *false*). V prípade, že za koreň bola vzatá unárna spojka NOT, je metóda ***`makeStructure`***`()` zavolaná na zvyšok výrazu (*word_from+1*, *word_to*, *true*).
- Ak medzi slovami s najnižším zanorením v zátvorkách neexistuje žiadna logická spojka, musí to byť nutne práve jeden term (prípadne term v zátvorkách). Inak ide o syntaktickú chybu.

Výsledkom je teda binárny strom, ktorý reprezentuje dotaz vo forme logickej formule termov.

- *Druhá fáza algoritmu prevodu* - Druhá fáza zabezpečuje prevod dotazu na disjunktívne konjunktívny tvar (DKT). Formálne algoritmus prevodu na DKT spočíva v aplikovaní niekoľkých transformačných pravidiel, ktoré bez zmeny pravdivostnej hodnoty formule modifikujú vnorenie spojok AND a OR tak, že logickú spojku AND vkladajú dovnútra OR výrazov (čo je cieľom). Prirodzene, transformačné pravidlá sú odvodené zo známych de Morganových pravidiel. V nasledujúcich pravidlách predstavujú *x, y* a *z* akékoľvek logické formule. Pravidlá sú napísané v prefixovom tvare (namiesto bežného infixového).

1. $NOT(NOT(x)) \rightarrow x$
2. $NOT(AND(x,y)) \rightarrow OR(NOT(x), NOT(y))$
3. $NOT(OR(x,y)) \rightarrow AND(NOT(x), NOT(y))$
4. $AND(OR(x,y),z) \rightarrow OR(AND(x,z), AND(y,z))$
5. $AND(x, OR(y,z)) \rightarrow OR(AND(x,y), AND(x,z))$

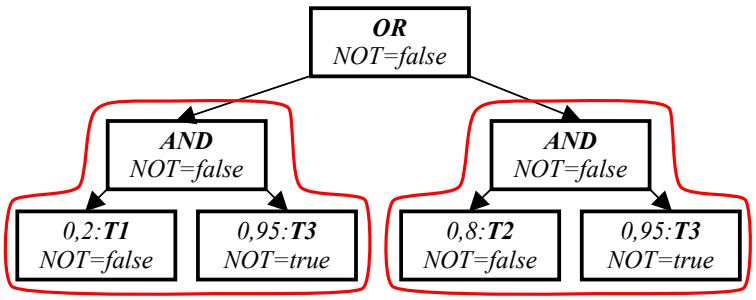
Pravidlá 1, 2 a 3 zabezpečujú odstránenie dvojitého záporu a distribúciu záporu na čo najnižšiu úroveň. Nosnými sú pravidlá 4 a 5, ktoré vyjadrujú distributívnosť spojky AND voči spojke OR, čím spojku AND presúvajú na nižšiu úroveň (bližšie k termom) na úkor spojky OR. Dá sa ukázať, že formula je v DKT práve vtedy, ak už nie je možné aplikovať žiadne zo systému pravidiel 1 – 5.

Implementácia aplikácie pravidiel je založená na práci so stromovými štruktúrami. Všetky transformačné pravidlá sú reprezentované binárnym stromom popísaným v predchádzajúcej kapitole, ale s tým rozdielom, že listy stromu nie sú termy, ale tzv. abstraktné elementy

(`frc_finder.core.convertor_b2v.BQueryAbstractElement`) zastupujúce akúkoľvek logickú formulu. Pri aplikácii niektorého z pravidiel sa najprv zistí, či ľavá (zdrojová) časť transformačného pravidla je podstromom transformovanej formuly. Ak áno, zistia sa (prehľadávaním do hĺbky) mapovania všetkých abstraktných elementov transformačného pravidla na podformule transformovanej formuly. Inými slovami, v transformovanej formule sa nájdu podformule odpovedajúce všetkým abstraktným elementom zdrojovej časti transformačného pravidla. Nakoniec sú abstraktné elementy v cieľovej časti transformačného pravidla na základe zisteného mapovania nahradené príslušnými podformulami a celý podstrom transformovanej formule odpovedajúci zdrojovej časti transformačného pravidla je nahradený cieľovou časťou transformačného pravidla.

- *Tretia fáza algoritmu* - Po prevode formule do DKT je už len potrebné nájsť všetky konjunkcie termov, ktoré budú použité ako vektorové dotazy. Použitý je rozšírený formát, kde všetky termy vektora sú povinné. Nájsť konjunkcie termov v strome reprezentujúcom formulu v DKT je jednoduché. V dôsledku použitia pravidiel sú spojky AND na nižších úrovniach a teda bližšie pri listoch (termoch) ako spojky OR. Strom sa preto prehľadáva od koreňa do hĺbky. Prvý uzol na každej ceste z koreňa do listu (listový alebo nelistový), ktorý nereprezentuje logickú spojku OR je koreňom podstromu, ktorého listy (termy) sú už len v konjunkcii.

  Formula použitá ako príklad v predchádzajúcej kapitole $(0{,}2 : T1\ OR\ 0{,}8 : T2)\ AND\ NOT(0{,}95 : T3)$ vyzerá po transformácii na DKT nasledovne: $(0{,}2 : T1\ AND\ NOT(0{,}95 : T3))\ OR\ (0{,}8 : T2\ AND\ NOT(0{,}95 : T3))$. Sú v nej dve konjunkcie termov: $(0{,}2 : T1\ AND\ NOT(0{,}95 : T3))$ a $(0{,}8 : T2\ AND\ NOT(0{,}95 : T3))$. V stromovej reprezentácii formule odpovedajú konjunkcie dvom vyznačeným podstromom. Korene týchto podstromov sú prvými uzlami na ceste z koreňa do listov, ktoré nereprezentujú spojku OR.



Ostatné konverzie dotazu (dokument zadaný textom a odkazom)
Ani jeden z týchto formátov nie je systémom FRC Finder podporovaný. Dôvod je náväznosť na index systému Egothor.

V prípade dotazu zadaného textom je dôvodom neexistencia indexačného algoritmu, ktorý k zadanému textu nájde jeho reprezentáciu vo forme ohodnoteného vektora termov. Tento algoritmus by bol súčasťou modelu INDEX v prípade, že by bol systém FRC Finder aj po stránke tvorby indexových súborov samostatným DIS. Prirodzene, takýto algoritmus mohol byť naimplementovaný aj samostatne, ale aby bol korektný,

mal by na rovnakých vstupoch dávať rovnaké výsledky ako indexačný algoritmus systému Egothor. Šlo by teda v podstate o duplicitnú a teda zbytočnú prácu.

V prípade dotazu odkazom na zaindexovaný dokument je príčinou organizácia indexových súborov Egothoru, kde neexistuje jednoduchá možnosť vyhľadávania reprezentácie dokumentu za základe jeho metadát (URL). Podrobnejšie je problém špecifikovaný v kapitole 4.3.2, konkrétne v podkapitole „Prvá fáza vyhľadávania“, ktorá platí aj pre indexové súbory Egothoru.

7.3.2 Konvertor pre boolský DIS – Egothor

Konvertor je implementovaný priamo v balíkoch systému Egothor. Konkrétne ide o balík `org.egothor.final_objects` a triedu `ConvertorB`. Plní dve úlohy. Základnú, ktorou je konverzia všetkých typov dotazov na ohodnotenú logickú formulu a premosťovaciu, ktorá zabezpečuje možnosť pripojenia systému Egothor do systému optimálneho vyhľadávania.

Ako základ pre vytvorenie konvertora je použitá trieda `org.egothor.apps.Search`, ktorá je vlastne implementáciou vyhľadávača Egothor. Funkcia `main()` triedy `Search` očakáva dva parametre: adresár s indexovými súbormi a samotný dotaz, ktorý má byť riešený. Na základe týchto údajov sú jednorázovo vytvorené vnútorné štruktúry konvertora (hlavne `Tanker`) a následne opakovane spúšťaný dotaz. V implementácii konvertora sú potrebné parametre (adresár s indexovými súbormi) dodané pomocou metódy `runConfig()` rozhrania `DISIFace` (viď popis rozhrania v kapitole 8.2). Nosná štruktúra `Tanker` je vytvorená len raz a to po dodaní potrebných parametrov a je používaná počas celej doby behu systému.

Algoritmus konverzie dotazu na natívny tvar systému Egothor je veľmi jednoduchý.

Konverzia dotazu z vektorového tvaru
Prevod spočíva v separácii termov (neohodnotených) z vektora a ich následné umiestnenie do disjunkcie.

Konverzia dotazu z boolského tvaru
V boolskom type dotazu sú v podstate len nahradené logické spojky NOT, AND a OR za výrazy „-“, „&&” a “||”. Dotaz z modulu DISTRIBÚTOR je však potrebné predspracovať, pretože Egothor vyžaduje správne uzátvorkovanie dotazu vo forme logickej formule (dotaz z modulu DISTRIBÚTOR nemusí byť uzátvorkovaný - uplatňuje sa priorita logických spojok). Predspracovanie je zabezpečené balíkom `org.egothor.final_objects.convaertor_b2e`, ktorý je zjednodušenou verziou balíka `frc_finder.core.convertor_b2v` systému FRC Finder. Odstránené boli všetky súčasti zabezpečujúce konverziu do vektorového tvaru, takže ostali len súčasti zabezpečujúce prevod dotazu vo forme reťazca do vnútorných štruktúr a ich následné zobrazenie vo formáte vhodnom pre Egothor.

Konverzia dotazu z textovej podoby
Na konverziu je použitý štandardný indexačný algoritmus systému Egothor, ktorého implementácia je spúšťaná v konštruktore triedy `org.egothor.html.HTMLDocument`. Trieda je naimplementovaná dostatočne všeobecne, čoho príkladom je aj parameter konštruktora. Je ním otvorený textový kanál (character stream). Pri indexácii súborov je použitý `java.io.FileReader`, pri indexácii dotazu vo forme textu je to

`java.io.StringReader`. Výsledky indexácie poskutuje metóda `getIList()`, ktorou je možné získať všetky termy a ich váhy. Z nich je už len poskladaný dotaz.

Konverzia z dokumentu zadaného odkazom
Nie je podporovaná z rovnakého dôvodu ako v konvertore systému FRC Finder (viď. 7.3.1).

8. Rozhrania medzi modulmi systému

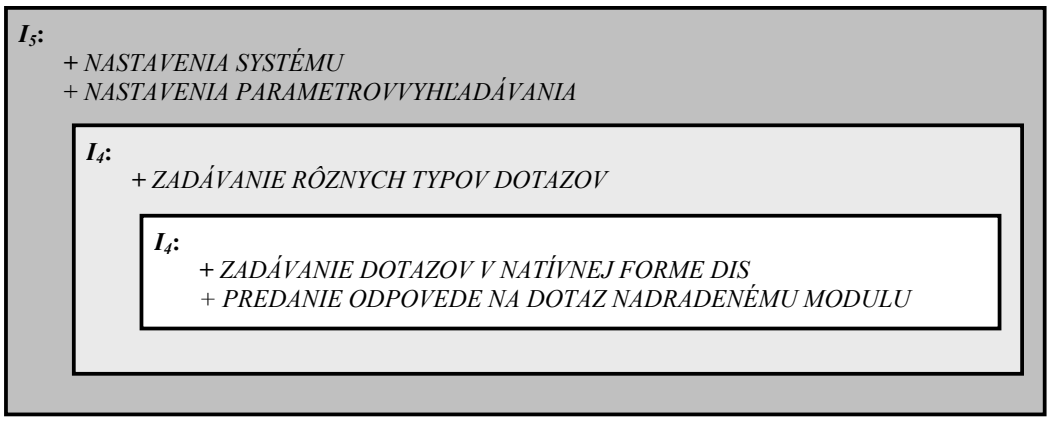
Cieľom tejto diplomovej práce je vybudovanie **modulárneho** systému, ktorý by umožňoval spracovať výsledky vyhľadávania poskytované rôznymi typmi dokumentografických informačných systémov a poskytnúť ich používateľovi.

Nespornou výhodou modulárnych systémov je ich jednoduchá modifikovateľnosť. V prípade potreby je totiž možné nahradiť niektorý z modulov úplne iným, a tak modifikovať funkčnosť a správanie celého systému. Aby bolo možné takýto modulárny systém zostaviť, je potrebné presne špecifikovať funkcie jednotlivých modulov a spôsob ich komunikácie s inými modulmi. Ide teda o presnú špecifikáciu rozhraní medzi modulmi. Systém optimálneho vyhľadávania má nasledovné typy modulov: DISTRIBÚTOR, KONVERTOR, DIS, INDEX. Nie všetky typy modulov však vzájomne komunikujú, pretože to nemá význam (komunikácia INDEX-DISTRIBÚTOR je zbytočná). Podľa schémy uvedenej v úvode práce je potrebné vyšpecifikovať nasledujúce „nepoužívateľské“ = „systémové“ rozhrania: I_1 = INDEX-DIS, I_2 = INDEX-KONVERTOR, I_3 = DIS-KONVERTOR, I_4 = KONVERTOR-DISTRIBÚTOR. Špeciálnym rozhraním je „používateľské“ rozhranie modulu DISTTRIBÚTOR – I_5, ktoré reprezentuje celý systém optimálneho vyhľadávania používateľovi. Kvôli istému používateľskému komfortu by malo byť grafické (GUI).

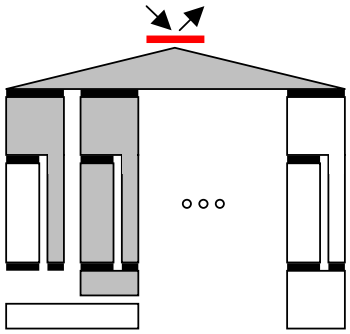
Niektoré rozhrania systému optimálneho vyhľadávania sú vo vzájomnom prirodzenom vzťahu, ktorý je daný funkčnosťou modulov, ktoré spájajú. Ide o rozhrania I_3, I_4, I_5, ktoré sú z pohľadu svojej šírky vo vzťahu $I_3 \subseteq I_4 \subseteq I_5$. Je to dané tým, že funkčnosť celého systému narastá zdola nahor.

- Modul DIS je zvyčajne dokumentografický informačný systém niektorého typu (vekorový, boolský, rozšírený boolský...). To znamená, že dokáže natívne spracovávať len dotazy jedného typu. Úlohou rozhrania I_3 je teda zadanie dotazu jedného typu a získanie odpovede.
- Modul KONVERTOR zabezpečuje konverziu rôznych typov dotazov (textom, logickou formulou, vektorom, odkazom na dokument) na typ, ktorý je natívny (prirodzený) „podradenému“ DIS. Cez rozhranie *I4* je preto potrebné predávať niekoľko rôznych typov dotazov a získavať na ne odpovede.
- „Na vrchole pyramídy“ je modul DISTRIBÚTOR, ktorý prijíma používateľské dotazy a distribuuje ich všetkým zaregistrovaným KONVERTOROM. Navyše, pri spracovávaní výsledkov berie ohľad na vhodnosť jednotlivých metód. Rozhranie I_5, ktoré je zvyčajne grafické, prípadne riešené konfiguračným súborom, plní niekoľko úloh. Hlavnou úlohou je poskytovať možnosť zadať používateľské dotazy rôznych typov a poskytovať (zobrazovať) výsledky vyhľadávania. Navyše musí rozhranie I_5 poskytovať možnosť nastaviť všetky ostatné parametre vyhľadávania a systému (koeficienty vhodnosti, ukladanie nastavení ...).

Nasledujúca schéme prehľadne ukazuje rastúce požiadavky na rozhrania I_3, I_4, I_5:

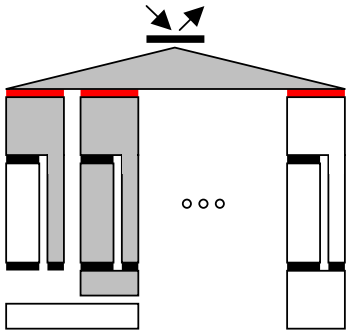


8.1 Rozhranie I_5 = Používateľské rozhranie DISTRIBÚTORa



Je to jediné „nesystémové“ rozhranie celého systému optimálneho vyhľadávania. Používateľ cez neho zadáva svoje dotazy v rôznych formátoch a konfiguruje celý systém. Rozhranie I_5 by teda malo byť čo najjednoduchšie a do istej miery používateľsky príjemné. Vhodnou formou je grafické rozhranie (viď implementácia systému), prípadne možnosť obsluhovať systém kombináciou konfiguračných súborov (konfigurácia systému) a príkazov v príkazovom riadku (dotazy). Odhliadnuc od možnosti konfigurácie systému je rozhranie I_5 formálne a funkčne totožné s rozhraním I_4.

8.2 Rozhranie I_4 = KONVERTOR-DISTRIBÚTOR



Je to nosné rozhranie systému optimálneho vyhľadávania. Podstata a funkčnosť systému optimálneho vyhľadávania je uložená v jedinom module – DISTRIBÚTOR. Modul je však sám o sebe len implementáciou algoritmu a nemá žiadne vlastné dátové zdroje. Dátovými zdrojmi sú pre neho práve jednotlivé dokumentografické informačné systémy, ktorých výsledky spracováva. Jednotlivé DIS komunikujú s modulom DISTRIBÚTOR práve cez rozhranie I_4 = KONVERTOR-DISTRIBÚTOR. V zmysle systému optimálneho vyhľadávania možno na toto rozhranie nahliadať ako na rozhranie DISTRIBÚTOR-„DIS“, pretože výsledky, ktoré dostáva DISTRIBÚTOR sú de facto výsledkami vyhľadávania celého DIS (bez ohľadu na vnútornú štruktúru DIS).

Rozhranie I_4 je z kaskády rozhraní I_3, I_4, I_5 ktoré sú prirodzene vo vzájomnom vzťahu $I_3 \subseteq I_4 \subseteq I_5$ najširším rozhraním. Zároveň je zo všetkých nepoužívateľských rozhraní systému optimálneho vyhľadávania aj najvoľnejším rozhraním, pretože na rozdiel od rozhraní I_1, I_2, I_3 neexistuje tesná väzba medzi naväzujúcimi modulmi.

Modul DISTRIBÚTOR jednoducho predá dotaz DISu v nejakej forme a očakáva výsledok v presne definovanej štruktúre. Spôsob jeho práce teda nijako nezávisí na spôsobe práce DIS.

V implementácii systému je rozhranie I_4 definované v balíku `frc_distributor.inferfaces` a konkrétne ide o rozhranie `DISIface`, ktoré odpovedá potrebám systému optimálneho vyhľadávania. Obsahuje nasledujúce metódy:

- `boolean` ***`runConfig`***`(Frame caller)` – Jednotlivé DIS sú do systému optimálneho vyhľadávania pridávané až „za behu“ cez grafické rozhranie, ktoré nedáva možnosť vstúpiť parametre cez príkazový riadok (nieže by to nebolo možné). Niektoré DIS však potrebujú pre svoje fungovanie nastaviť povinné parametre akými sú napríklad adresár s indexovými súbormi, logmi... Práve metóda `runConfig()` by mala spustiť formulár alebo textové okno, v ktorom by mohol používateľ nastaviť všetky parametre daného DIS. Funkcii je ako skutočný parameter predaná referencia na hlavné okno aplikácie, čo umožňuje DISu vytvoriť modálne konfiguračné okno. Návratová hodnota vypovedá o úspešnosti nastavenia všetkých povinných parametrov. Po úspešnom nastavení by aj metóda rozhrania `isReady()` mala vracať hodnotu `true`.

- `boolean` ***`isReady`***`()` – Metóda vracia status, respektíve pripravenosť DIS. V prípade, že je DIS nepripravený, nedáva žiadne výsledky vyhľadávania a je potrebné zavolať metódu `runConfig()`.

- `String` ***`runQuery`***`(String n_query, int format)` – Zavolaním tejto metódy predáva modul DISTRIBÚTOR jednotlivým „podradeným“ DIS používateľský dotaz v príslušnom formáte. Parameter `format` môže nadobúdať hodnoty `QT_BOOLEAN`, `QT_VECTOR`, `QT_TEXT`, `QT_URL_REFERENCE`, ktoré sú definované v samotnom rozhraní `DISIface` ako statické konštanty (QT je skratka od Query Type). Ich význam je názorný:
  - `QT_BOOLEAN` – dotaz je vo formáte logickej formule, kde sú použité logické operátory AND, OR a NOT, zátvorky a ohodnotené termy v tvare $[podobnosť : term]$. Priorita logických operátorov je štandardná: NOT > AND > OR, z čoho vyplýva, že dotaz nemusí byť uzátvorkovaný. Je to typický spôsob dotazovania na DIS v (rozšírenom) boolskom modeli.
    Príklad dotazu: *0,5:system AND (0,8:retrieval OR 0,2:information)*
  - `QT_VECTOR` – dotaz je vo vektorovom formáte. Bežný vektorový formát je rozšírený o príznak, či je daný term povinný alebo nie. Atómy majú tvar: $[podobnosť : term : povinný]$. Atómy sú oddelené znakom’;’. Bližšie je rozšírený formát popísaný v kapitole 7.2. Tento typ dotazu je typický pre vektorové modely DIS.
    Príklad dotazu: *0,5:system:T; 0,8:retrieval:F; 0,2:information:F*
  - `QT_TEXT` – dotazom je text, ktorý je považovaný za dokument. DIS by ho mal zaindexovať algoritmom, ktorý používa a výsledok indexácie následne použiť ako dotaz. Význam tohto spôsobu dotazovania spočíva v tom, že používateľ chce získať dokumenty, ktoré sa podobajú vloženému textu.
    Príklad dotazu: *information retrieval system shows all documents that ...*

- *QT_URL_REFERENCE* – ide o spôsob dotazovania, ktorý využíva už zaindexovaný dokument. Jednoznačným identifikátorom dokumentu na internete je jeho URL, preto je použitý ako referencia na dokument. DIS by mal v svojom indexe dokument s príslušným URL vyhľadať a jeho zaindexovaný tvar použiť ako dotaz.

Je prirodzené, že DIS nedokáže spracovať všetky typy dotazov. V tom prípade by mal byť výsledok vyhľadávania prázdny. Návratová hodnota typu `String` slúži na oznámenie akéhokoľvek chybového hlásenia (napríklad „Not supported query type“ v prípade, že DIS nepodporuje daný typ dotazovania). Všetky chybové hlásenia sú zobrazované modulom DISTRIBÚTOR.

- `DISResultIface` ***getResult()*** – Metóda vracia výsledok naposledy zadaného dotazu (metódou *runQuery*()) v akejkoľvek štruktúre spĺňajúcej verejné rozhranie `DISResultIface` (popísané v kapitole 8.2.1).

- `boolean` ***checkTerm***`(String` *n_term*`)` – Metóda kontroluje, či je term, zadaný ako parameter *n_term*, uložený v indexe systému. V prípade, že metóda vráti hodnotu *true*, mal by sa term vyskytovať aspoň v jednom dokumente a má teda zmysel dotazovať sa na neho. Metóda je využívaná pri kontrole existencie termov v prípade tvorby vektorového dotazu.

- `String` ***getInfo***`()` – Metóda vracia názov DIS a jeho krátky popis.

- `float` ***getMinSimilarity***`(), float` ***getMaxSimilarity***`()` – Metódy sú určené na zisťovanie minimálnej a maximálnej podobnosti dokumentu a dotazu, ktoré sú príslušným systémom poskytované. Tieto hodnoty sú potrebné pre normalizáciu výsledkov DIS na jednotné hodnoty podobností, čím sa zabraňuje zvýhodneniu výsledkov niektorého DIS. Musí platiť, že hodnota vrátená metódou *getMinSimilarity*() je ostro menšia ako hodnota vrátená metódou *getMaxSimilarity*().

- `void` ***delete***`()` – Metódou *delete*() oznamuje modul DISTRIBÚTOR príslušnému DIS fakt, že ho viac nebude používať. Zavolaním tejto metódy by mal DIS vyprázdniť všetky vyrovnávacie pamäte a zatvoriť všetky otvorené súbory.

8.2.1 Rozhranie DISResultIface

Rozhranie je definované v balíku *frc_distributor.interfaces*, konkrétne ide o rozhranie *DISResultIface*. V tomto rozhraní sú definované metódy, ktoré musí spĺňať štruktúra, ktorá je vrátená metódou getResult() rozhrania DISIface (metóda samozrejme vracia len referenciu, nie celú štruktúru) ako výsledok vyhľadávania. Rozhranie teda slúži na získanie výsledkov vyhľadávania od jednotlivých DIS vo forme štruktúry. Do úvahy prichádza aj možnosť, kedy by výsledky vyhľadávania boli predávané vo forme textového reťazca. Formát tohto reťazca by bol vzhľadom na množstvo a variabilitu informácií o jednom dokumente dosť komplikovaný a navyše by ho bolo treba na strane modulu DISTRIBÚTOR rozdeliť na jednotlivé zložky a spracovať. To samozrejme stojí nejaký čas. Ako efektívnejšie a spoľahlivejšie (v zmysle náchylnosti na chyby) sa preto javí predávanie výsledku referenciou na

štruktúru spĺňajúcu definované rozhranie. To dáva navyše možnosť implementovať túto štruktúru čo najefektívnejšie s ohľadom na potreby a povahu konkrétneho DIS (v prípade systému FRC Finder je rozhranie implementované haldou). Rozhranie má nasledovné metódy:

- `LinkedList<DISHitIface>` ***`getResultInList`***`()` – Je to najpoužívanejšia metóda rozhrania. Vracia zoznam dokumentov výsledku. Každý dokument výsledku (hit - zásah) je reprezentovaný istou štruktúrou, ktorá musí implementovať rozhranie DISHitIface (popis v kapitole 8.2.2). Je preferované, aby bol zoznam dokumentov usporiadaný zostupne podľa podobnosti dokumentu a dotazu (nie je to však nutná podmienka).

- `boolean` ***`containsHit`***`(String` *`n_location`*`)` – Metóda kontroluje, či sa vo výsledku vyskytuje dokument, ktorého URL je totožný so zadaným parametrom *`n_location`* (URL je v prostredí internetu jednoznačný identifikátor). V prípade, že sa požadovaný dokument vo výsledku vyhľadávania vyskytuje, návratová hodnota metódy je *`true`*, inak *`false`*.

- `int` ***`getSize`***`()` – Návratová hodnota metódy je počet dokumentov vo výsledku vyhľadávania. Počet dokumentov vo výsledku nesmie byť väčší ako hodnota *`MAX_RESULT_SIZE`* definovaná v rozhraní *`DISResultIface`* (v prípade väčšieho počtu dokumentov sú nadbytočné dokumenty ignorované).

- `void` ***`multiplySimilarities`***`(float` *`multiplicator`*`)` – Metóda násobí podobnosti všetkých dokumentov vo výsledku zadanou hodnotou *`multiplicator`*. To znamená, že ak by bol vo výsledku dokument, ktorého podobnosť s dotazom je *`sim`*, po volaní tejto metódy bude jeho podobnosť rovná *`sim*multiplicator`*. Zvýšenie podobností dokumentov je teda lineárne. Metóda je využívaná pri uplatnení koeficientov vhodnosti DIS pre daného používateľa.

- `void` ***`normalizeSimilarities`***`(float l_min, float l_max, float g_min, float g_max)` – Metóda normalizuje podobnosti dokumentov vo výsledku z lokálneho intervalu (interval, v ktorom sú podobnosti dokumentov pred normalizáciou) $\langle l\_\min, l\_\max \rangle$ na globálny interval (interval, v ktorom by mali byť podobnosti dokumentov po normalizácii) $\langle g\_\min, g\_\max \rangle$. Štandardne sa používa lineárny prevod z hodnoty L na hodnotu G podľa vzťahu $G = (L - l\_\min) \cdot ((g\_\max - g\_\min)/(l\_\max - l\_\min)) + g\_\min$, no nie je to nutné. Každý DIS si môže prispôsobiť spôsob normalizáie hodnôt podobností podľa svojich potrieb, napríklad v závislosti na použitej podobnostnej funkcii. Dôležité je, aby hodnoty podobností dokumentov vo výsledku po zavolaní metódy *`normalizeSimilarities`*`()` boli z intervalu $\langle g\_\min, g\_\max \rangle$.

- `float` ***`getMinSimilarity`***`(), float` ***`getMaxSimilarity`***`()` – Metódy vracajú hodnotu najmenšej a najväčšej podobnosti dokumentu vo výsledku vyhľadávania. V prípade, že je výsledok vyhľadávania prázdny, metódy vracajú hodnoty *`Float.MAX_VALUE`*, prípadne *`-Float.MAX_VALUE`*.

8.2.2 Rozhranie DISHitIface

Podobne ako rozhranie `DISResultIface`, aj toto slúži na odovzdanie informácie o dokumentoch, ktoré spĺňajú požadovaný používateľský dotaz. Zatiaľ čo `DISResultIface` odpovedalo pohľadu na celú množinu dokumentov, rozhranie `DISHitIface` odpovedá pohľadu na jeden dokument výsledku – teda zásah – anglicky „Hit". Aj v tomto prípade bolo možné informácie o dokumente predávať v nejakej definovanej textovej podobe, no to by kládlo vyššie nároky na spracovanie textu čím by vzrástla aj chybovosť systému. Údaje o dokumentoch sú preto predávané referenciou na štruktúru implementujúcu rozhranie `DISHitIface`, ktoré má definované nasledujúce metódy:

- `String` ***`getLocation`***`()` – Metóda vracia jednoznačný identifikátor dokumentu – jeho URL. Ak majú dva dokumenty z rôznych výsledkov rovnaké URL, sú brané ako ten istý dokument.

- `float` ***`getSimilarity`***`()` – Metóda vracia podobnosť dokumentu a dotazu.

- `DISDocMetaDataIface` ***`getMetaData`***`()` – Metóda vracia metadáta dokumentu v akejkoľvek štruktúre implementujúcej rozhranie `DISDocMetaDataIface` (bližší popis v kapitole 8.2.3).

Rozhranie je uložené v balíku `frc_distributor.interfaces`, konkrétne v súbore `DISHitIface`.

8.2.3 Rozhranie DISDocMetaDataIface

Rozhranie slúži na predávanie informácií o metadátach dokumentu. Keďže metadát dokumentu je mnoho druhov (URL, autor, kľúčové slová ...) sú organizované v asociatívnom poli – t.j. ako dvojice kľúč-hodnota (obe hodnoty sú znakové reťazce). Kľúče najbežnejšie používaných typov metadát sú súčasťou rozhrania. Ide o konštanty:

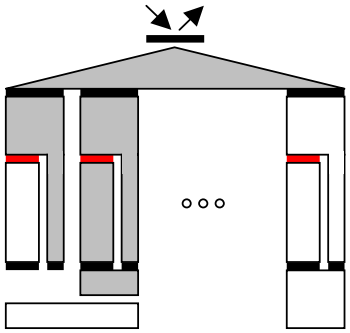
- `MD_LOCATION` – jednoznačná URL lokácia dokumentu
- `MD_BASE_URL` – doména, z ktorej je dokument.
- `MD_SUMMARY` – kľúčové slová dokumentu
- `MD_TITLE` – nadpis dokumentu

Ako všetky ostatné verejné rozhrania je aj toto súčasťou balíka `frc_distributor.interfaces`. Jeho súčasťou sú nasledujúce metódy:

- `String` ***`get`***`(String key)` – Metóda vracia hodnotu metadát v závislosti na parametri key, ktorý môže nadobúdať niektorú z definovaných hodnôt (`MD_LOCATION`, `MD_BASE_URL`, `MD_SUMMARY`, `MD_TITLE`), prípadne iný kľúč. Ak zadaný parameter `key` nie je v metadátach obsiahnutý, návratová hodnota je prázdny reťazec.

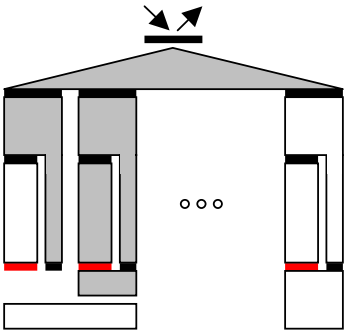
- `Set<Map.Entry<String, String>>` ***`entrySet`*** `()` – Každý zo systémov môže používať iné kľúče pre metadáta. Preto nie je možné bez znalosti kľúčov zistiť všetky metadáta, ktoré sú v dokumente uložené. Práve na získanie všetkých metadát dokumentu slúži metóda `entrySet()`. Podľa typu návratovej hodnoty je jasné, že metóda vracia množinu dvojíc kľúč-hodnota – teda množinu všetkých metadát dokumentu.

8.3 Rozhranie I_3 = DIS-KONVERTOR



Rozhranie I_3 je funkčným zúžením rozhrania I_4. Hlavný rozdiel spočíva v možnosti rozhrania I_4 predávať dotazy v rôznych formátoch, čo je pre rozhranie I_3 úplne zbytočné, pretože dotazy predáva priamo modulu DIS, ktorý dokáže spracovať len dotazy v jeho natívnej podobe. Pre praktické použitie by však bolo zbytočné definovať úplne nové formálne rozhranie, pretože by sa stratila možnosť napojiť modul DIS priamo pod modul DISTRIBÚTOR (čo znamená vynechať modul KONVERTOR). Formálne sú teda rozhrania I_3 a I_4 totožné, funkčne je I_3 zúžením I_4. Fakticky je rozdiel len v správaní sa metódy `runQuery()`, ktorá reaguje len na ten typ dotazu, ktorý je natívny pre daný DIS. Pre ostatné typy vracia chybové hlásenie (napr. „Not supported query type.“).

8.4 Rozhranie I_1 = INDEX-DIS



Ide o rozhranie, ktoré spája moduly INDEX a DIS. Z hľadiska výkonnosti celého systému optimálneho vyhľadávania je rozhodujúca práve miera spolupráce modulov INDEX a DIS. Dôvodom je to, že modul INDEX pracuje s údajmi, ktoré sú uložené na pevnom disku počítača, čo znamená, že dodanie údajov modulom INDEX je časovo najnáročnejšia operácia celého procesu vyhľadávania (aplikovanie podobnostnej funkcie, dodanie a spracovanie výsledku, atď. sa dejú už v operačnej pamäti a sú preto neporovnateľne rýchlejšie). Je preto nesmierne dôležité, aby modul DIS rešpektoval spôsob uloženia dát a teda aj spôsob práce INDEXu a informácie vyžadoval vo vhodnom poradí a forme. Väzba medzi modulmi DIS a INDEX by mala byť preto čo najtesnejšia.

Zúžením väzby však fakticky zanikajú dva osobitné moduly a nahrádza ich jeden, ktorého funkčnosť pokrýva funkčnosť zaniknutých modulov. Význam rozhrania I_1 ako rozhrania modulov INDEX a DIS teda zaniká.

Ďalším faktom hovoriacim proti existencii formálneho rozhrania I_1 je to, že indexové súbory rôznych typov DIS môžu obsahovať iné dáta. Príkladom je termová frekvencia termov, ktorá je potrebná vo vektorovom modeli DIS, no nepotrebná pre výpočet rýdzo boolského modelu DIS. Je preto nemožné použiť vzájomne nekompatibilné moduly DIS a INDEX.

Napriek tomu je možné naznačiť formu rozhrania I_1, ktoré pokrýva väčšinu potrieb všetkých typov DIS. Za základ je považované rozhranie indexu boolského modelu, ktoré je obohatené o potreby vektorového modelu, s ohľadom na ďalšie možné rozšírenia (dostatočná robustnosť riešenia).

V popise rozhrania sa počíta s tým, že termy sú identifikované svojou textovou podobou a dokumenty majú vnútorný jednoznačný identifikátor typu `Long`. Popis

rozhrania pre indexové súbory všeobecného dokumentografického informačného systému:

- `Set<Long>` ***`getTermsDocuments`***`(Collection<String>` *`terms`*`)` – Metóda odpovedá vyhľadávaniu relevantných dokumentov termu v invertovanom zozname dokumentov. Ako vstupný parameter sa očakáva zoznam (množina) termov zadaných v textovej podobe. Na výstup vráti množinu identifikátorov dokumentov relevantných k vstupu, teda všetky dokumenty, ktoré obsahujú aspoň jeden term zo vstupnej kolekcie s nenulovou váhou.

- `Float` ***`getTermDocumentWeight`***`(String` *`term`*`, Long` *`doc_id`*`, Boolean` *`normalised`*`)` – Metóda vracia váhu termu *`term`* v dokumente *`doc_id`*. Vstupnými parametrami sú jednoznačný identifikátor termu, dokumentu a príznak, či výsledkom má byť normalizovaná alebo nenormalizovaná váha. Metóda vlastne vracia jednu hodnotu z matice údajov „termy x dokumenty".

- `Set<Map.Entry<String, Float>>` ***`getDocumentWeight`***`(Long` *`doc_id`*`, Boolean` *`normalised`*`)` – Metóda vracia váhy termov v dokumente *`doc_id`* v asociatívnom poli term – váha. Vstupnými parametrami sú jednoznačný identifikátor dokumentu a príznak, či výsledkom majú byť normalizované alebo nenormalizované váhy. Návratová hodnota metódy vlastne odpovedá jednému riadku z matice údajov „termy x dokumenty".

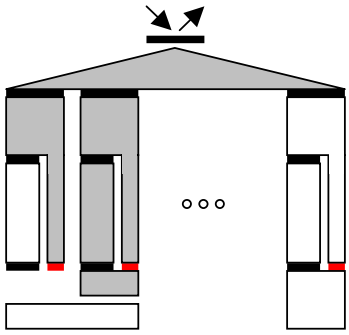
- `Set<Int>` ***`getTermDocumentProximities`***`(String` *`term`*`, Long` *`doc_id`*`)` – Úlohou metódy je poskytovať množinu výskytov termu *term* v dokumente *doc_id*. Pre rýchlejšie spracovanie by mali byť výskyty vo výsledku zotriedené vzostupne. Na základe výsledkov metódy je generovaná kontextová ukážka dokumentu a v boolských systémoch sa pomocou metódy rieši aj vyhľadávanie – ide o proximitné obmedzenia kladené na termy (napr. term T1 a T2 nesmú byť od seba vzdialené viac ako 20 slov...)

- `Set<Map.Entry<String, String>>` ***`getDocMetaData`***`(Long doc_id)` – Metóda poskytuje sekundárne informácie (metadáta) o dokumente zadanom jednoznačným číselným identifikátorom *`doc_id`*. Vrátené sú všetky dostupné metadáta dokumentu.

- `String` ***`getDocMetaData`***`(Long doc_id, String` *`type`*`)` – Metóda má podobnú úlohu ako funkcia *`getDocMetaData()`*, ale nevracia všetky metadáta dokumentu. Vracia hodnotu jedného metadáta s kľúčom *`type`* dokumentu *`doc_id`*. Keďže rozsah a typ metadát môže byť pre rozličné druhy dokumentov iný, kľúč asociatívneho poľa (parameter *`type`*) je identifikátor typu žiadanej informácie a je ho možné pružne rozširovať.

Ako už bolo spomenuté, význam rozhrania I_1 zaniká, čo sa odrazilo aj v implementácii vektorového modelu DIS, kde nie je rozhranie I_1 vôbec formálne definované. Avšak funkcie, ktoré boli spomenuté v popise rozhrania, systém skutočne používa v „logickej" vrstve obaľujúcej indexové súbory (viď kapitola o implementácii indexu FRC FIndera 5.2.1)

8.5 Rozhranie I_2 = INDEX-KONVERTOR



Existujú situácia, kedy je potrebné, aby modul KONVERTOR mohol pristupovať priamo k indexovým súborom. Ide o prípady, kedy je používateľským dotazom odkaz na zaindexovaný dokument, prípadne voľný text.

- V prípade dotazu *odkazom na dokument* je potrebné, aby modul INDEX požadovaný dokument našiel v príslušných súboroch a vrátil jeho vnútornú reprezentáciu vhodnú na dotazovanie.
- V druhom prípade, kedy je dotazom *voľný text*, je potrebné využiť indexačný algoritmus dokumentov. Od modulu sa teda očakáva, že zadaný text zaindexuje a výsledok indexácie (čo je vhodná forma na dotazovanie) vráti modulu KONVERTOR To všetko za predpokladu, že indexačný algoritmus je implementovaný modulom INDEX.

Formálne možno metódy rozhrania popísať nasledovne:

- `String` ***`getQueryDocument`***`(String` *`location`*`)` – Metóda hľadá dokument s jednoznačným identifikátorom *`location`* (reálnym parametrom je URL dokumentu) v indexových súboroch a vracia dotaz vo formáte, ktorý je natívny príslušnému DISu. Metóda pokrýva potreby modulu KONVERTOR v prípade dotazu vo forme odkazu na zaindexovaný dokument.

- `String` ***`getQueryText`***`(String` *`text`*`)` – Metóda slúži na transformáciu dotazu zadaného textom do formátu prirodzeného pre príslušný DIS. Transformácia sa dosahuje použitím indexačného algoritmu.

Vzhľadom na diskusiu o význame, respektíve bezvýznamnosti rozhrania I_1 v kapitole 8.4, v prípade splynutia tesne naviazaných modulov DIS a INDEX, splynú aj rozhrania I_2 a I_3. Z tohto dôvodu nie je rozhranie I_2 medzi modulmi INDEX a KONVERTOR formálne definované a jeho funkčnosť je prenesená do rozhrania I_3.

9. Výkonnostné a kvalitatívne charakteristiky

9.1 FRC Finder – DIS vo vektorovom modeli

Výkon implementovaného systému FRC Finder nebol testovaný na reálnych dátach, ale na dátach generovaných rutinou `Generator`, ktorá je súčasťou testovacieho balíčku systému Egothor. Dáta generované touto rutinou sú štatisticky podobné reálnym dátam, takže test výkonnosti systému simuluje reálnu situáciu.

Testy boli vykonané komparatívnym spôsobom – porovnávaný bol výkon Egothoru a FRC Finderu. Oba systémy používali rovnaké indexové súbory a bol im položený ten istý dotaz. Meraná bola odozva oboch systémov. Priemerná doba odozvy systému FRC Finder bola o 35% vyššia ako odozva systému Egothor (doba odozvy FRC Finderu = 135% * doba odozvy Egothoru). Tento výsledok sa dá odôvodniť spôsobom výpočtu vektorového DIS, ktorý na výpočet podobnosti dotazu a dokumentu potrebuje viac informácií a teda viac prístupov na pevný disk.

9.2 Systém optimálneho vyhľadávania

Systém optimálneho vyhľadávania má v predvádzanej konfigurácii zaregistrované dva podradené DIS – Egothor a FRC Finder. Oba DIS pracujú nad spoločnými indexovými súbormi, čo v konečnom dôsledku neznižuje problém kritéria predikcie, ale dáva vyššiu šancu, že výsledky niektorého zo systémov budú používateľovi subjektívne viac vyhovovať. Tým sa zvyšujú hodnoty veličín, ktorými je charakterizovaná kvalita DIS – presnosť a úplnosť. Do systému je však možné zaregistrovať akýkoľvek DIS, ktorý implementuje požadované rozhranie, čím môže narásť aj počet dátových zdrojov, a tým sa znížiť problém kritéria predikcie.

Čo sa týka výkonnostných charakteristík celého systému optimálneho vyhľadávania, sú závislé na charakteristikách podradených DIS. Nemá zmysel uvažovať o bežnej výkonnostnej charakteristike, ktorou je počet prehľadaných dokumentov za jednotku času, pretože jednotlivé DIS obsahujú rôzne počty dokumentov, ktoré sa navyše môžu opakovať. Prakticky má zmysel uvažovať hlavne o dobe odozvy na používateľský dotaz. Čisto teoreticky je možné dosiahnuť dobu odozvy najpomalšieho systému navýšenú o konštantný čas potrebný na porovnanie výsledkov jednotlivých DIS. Predpokladom je paralelný beh jednotlivých DIS bez vzájomného ovplyvňovania, t.j. v ideálnom prípade na rôznych počítačoch. Predložená implementácia DIS a systému optimálneho vyhľadávania teoretickú možnosť nespĺňa, pretože oba DIS bežia na jednom počítači a nad spoločnými súbormi. Navyše distribúcia dotazu jednotlivým DIS je jednovláknová a teda sériová. Doba odozvy je teda súčtom časov vyhľadávania jednotlivých DIS navýšená o čas potrebný na spracovanie a zobrazenie výsledkov.

10. Záver

Cieľom práce je podrobný teoretický rozbor a implementácia alternatívneho vyhľadávača k systému Egothor. V rámci práce bol naimplementovaný DIS založený na vektorovom modeli, ktorý však nie je úplne samostatný, pretože so systémom Egothor zdieľa indexové súbory. V teoretickej časti práce je však možné nájsť kompletný návrh organizácie a štruktúry indexových súborov pre vektorový model DIS. Možno skonštatovať, že cieľ bol dosiahnutý, čím sa paradoxne vytvoril priestor na ďalšiu prácu, testovanie a výskum. Ten sa dá nasmerovať na systém optimálneho vyhľadávania, ktorý bol naimplementovaný, aby bolo možné výsledky oboch systémov porovnávať a vyhodnocovať.

Systém optimálneho vyhľadávania je navrhnutý modulárne, čo poskytuje široké možnosti jeho konfigurácie. Do systému je možné dynamicky pridávať a odoberať DIS, ktoré implementujú presne definované rozhranie. Jednotlivé DIS zapojené do systému optimálneho vyhľadávania môžu byť úplne nezávislé, no nemusia. Príkladom vzájomnej závislosti DIS je spoločný index Egothoru a FRC Finderu. Vďaka modularite celého systému je možné jednotlivé moduly samostatne zdokonaľovať, prípadne zamieňať za iné.

Systém optimálneho vyhľadávania poskytuje štyri spôsoby zadania dotazu: ohodnotenou logickou formulou, ohodnoteným vektorom termov, časťou dokumentu (textom) a odkazom na zaindexovaný dokument (pomocou URL). V predvádzanej konfigurácii sú naplno využiteľné len prvé dva zo spomenutých spôsobov. Dotaz časťou dokumentu dokáže riešiť len systém Egothor, pretože obsahuje indexačný algoritmus, ktorým dokáže zadaný text korektne spracovať. FRC Finder vôbec neobsahuje indexačnú jednotku a teda ani indexačný algoritmus, a preto tento spôsob dotazu nedokáže korektne vyhodnotiť. Posledný zo spomenutých spôsobov zadania dotazu, odkazom na zaindexovaný dokument, je neriešiteľný v dôsledku organizácie indexových spôsobov, ktorá neumožňuje pristupovať k reprezentácii dokumentu v indexových súboroch pomocou žiadneho z metadát dokumentu a teda ani pomocou URL. Indexové súbory obsahujúce metadáta dokumentov majú v indexe Egothoru jednoducho inú úlohu.

V rámci práce bola otestovaná kofigurácia s dvoma DIS, ktorá potvrdila nárast subjektívnych kvalitatívnych veličín celého systému – presnosti a úplnosti. Vzhľadom na spoločný dátový zdroj Egothoru a FRC Finderu sa neprejavilo zníženie problému kritéria predikcie.

V priebehu písania diplomovej práce bola vydaná ďalšia verzia systému Egothor (1.3), ktorej indexové súbory sú modifikované. Systém FRC Finder však tieto zmeny nereflektuje a pracuje s verziou 1.2.6.

11. Zoznam použitej literatúry

11.1 Problematika DIS

[1D] Kopecký M. (2003): Dokumentografické informační systémy, materiály k prednáške DBI010, MFF UK, Praha.

[2D] Pokorný J., Snášel V., Húsek D. (1998): Dokumentografické informační systémy, Karolinum, Praha.

[3D] Pokorný J., Žemlička M. (2004): Základy implementace souborů a databází, Karolinum, Praha

[4D] Kůrka P. (2001): Úvod do matematické logiky a teorie množin, materiály s prednáške AIL063, MFF UK, Praha

[5D] C.J. van Rijsbergen (1979): Information Retrieval, Second edition, Butterworths, London

[6D] F.W. Lancaster (1968): Information Retrieval Systems: Characteristics, Testing and Evaluation, Wiley, New York

[7D] Vojtáš P. (2004): Kvantitativní datové modely a flexibilní vyhledávání, materiály k prednáške DBI021, MFF UK, Praha

[8D] Novák V. (1990): Fuzzy množiny a jejich aplikace, STNL, Praha

11.2 Problematika jazyka JAVA

[1J] Spell B. (2002): JAVA Programujeme profesionálně, Computer Press, Praha

[2J] Hnětynka P. (2005): JAVA, materiály k prednáške PRG013

[3J] Gosling J. (2005): Java Language Specification, Addison-Wesley, Sun Microsystems, Inc.

Príloha A – používateľská príručka systému optimálneho vyhľadávania

1. Inštalácia

Inštalácia programu pre bežné systémy (platforma Win32, UNIX) spočíva v skopírovaní kompletného inštalačného balíčka z CD ROM na pevný disk počítača. V prípade nejakého menej bežného operačného systému (OS/2) sa odporúča skompilovať zdrojové kódy podľa skriptu `build.xml`.

2. Spustenie programu a registrácia DIS

Spustenie programu
Inštalačný balík obsahuje v koreňovom adresári súbory *run.bat* a *run.xml*. Súbor `run.bat` je možné spustiť len na Win32 systémoch, zatiaľ čo súbor `run.xml` je možné predať utilite *ant* ako parameter, čo je použiteľné na všetkých platformách.

Registrácia DIS
Po spustení sa zobrazí grafické okno, v ktorom je päť záložiek. Prvou z nich je záložka *Config*, ktorá slúži na registráciu DIS a nastavenie multiplikátorov. Registrácia DIS má 3 kroky:

- V prvom rade je potrebné skopírovať `*.class` súbory požadovaného DIS do adresára `dises`. Ak DIS pre svoj beh potrebuje použiť nejaké externé knižnice, je ich možné skopírovať do adresára `lib`, ktorý je pri spúšťaní systému optimálneho vyhľadávania označený ako `classpath` – teda adresár, v ktorom sa nachádzajú triedy systému. Aj externé knižnice musia byť uložené ako `*.class` súbory.
- Do ktoréhokoľvek voľného poľa *DIS Class* v záložke *Config* vložiť plný názov triedy (vrátane balíčka), ktorá implementuje definované rozhranie `frc_distributor.interfaces.DISIface` a potvrdiť tlačidlom *Activate*. V tomto okamihu systém skontroluje, či zadaná trieda existuje a či implementuje požadované rozhranie. Prípadné chybové hlásenia sú zobrazené v poli *Info/Status*. Maximálny počet DIS, ktoré môžu byť zaregistrované v systéme je 8. Prázdne riadky na registráciu DIS je možné pridávať/odoberať tlačidlami +/- na pravej strane.
- Po úspešnej aktivácii triedy sa automaticky spustí konfiguračné okno zvoleného DIS. Konfiguračné okno je už súčasťou DIS a zvyčajne slúži na nastavenie cesty k indexovým súborom, prípadne nejakých iných špeciálnych nastavení. Po úspešnej aktivácii a konfigurácii by mal byť v poli *Info/Status* len čierny text „Ready“. Ak tomu tak nie je, je potrebné systém znovu konfigurovať a to tlačidlom *Config*.

Po registrácii je možné nastaviť multiplikátor, ktorým budú násobené výsledky príslušného DIS. Povolené sú hodnoty z intervalu $\langle 0,1 \rangle$.

Uloženie a načítanie konfigurácie zo súboru
Aby nebolo nutné stále vpisovať celé názvy tried a nastavovať multiplikátory všetkých používaných DIS, je možné tieto informácie uložiť do súboru a následne ich zo súboru načítať. Proces sa ovláda bežne používaným spôsobom – v hlavnom menu programu v podmenu *File* sú položky *Load config* a *Save config*. Nie je exaktne definovaná žiadna prípona konfiguračných súborov a ani adresár, v ktorom musia byť

uložené, no existuje adresár `configs`, ktorý je určený práve na ukladanie konfiguračných súborov. Po načítaní nejakého konfiguračného súboru je potrebné všetky DIS aktivovať a nakonfigurovať.

V priloženej distribúcii sú uložené preddefinované konfiguračné súbory: `egothor.sav`, `frc_finder.sav` a `both.sav`.

3. Vyhľadávanie

Systém poskytuje štyri spôsoby zadávania dotazu: ohodnoteným vektorom termov, ohodnotenou logickou formulou termov, textom a odkazom na zaindexovaný dokument. Tomu odpovedajú záložky *Vector query*, *Boolean query*, *Text query* a *URL reference query*. Vyhľadávať je možné len v prípade, že je v systéme aregistrovaný aspoň jeden DIS.

Dotaz ohodnoteným vektorom termov
Jeden prvok vektora má tri časti: term (pole *Term*), váha termu (pole *Value*) a nutnosť termu (pole *Necessary*). Ak zadaný term neexistuje v databáze žiadneho zo zaregistrovaných DIS, je pole *Term* zvýraznené červeným pozadím. Takýto term sa pri vyhľadávaní neberie do úvahy. Pole Necessary znamená, že takto dokument vrátený na výstupe musí obsahovať označený term.

Dotaz ohodnotenou logickou formulou termov
Dotaz môže obsahovať zátvorky, logické spojky AND, OR a NOT a ohodnotené termy. Ohodnotený term je dvojica $value:term$. Systém berie do úvahy priority logických spojok: NOT>AND>OR, ktoré sa využije v prípade, že nebudú použité zátvorky.
Príklad dotazu: *0,5:system AND (0,8:retrieval OR 0,2:information)*

Dotazovanie textom
Ako dotaz je použitá ukážka textu, ktorý sa týka problematiky, o ktorej chce používateľ získať informácie. Môže to byť časť dokumentu (Ctrl+C, Ctrl+V), prípadne výpis dôležitých termov.

Dotazovanie odkazom na zaindexovaný dokument
Dotazom je URL dokumentu, ktorý má byť použitý ako dotaz. Podobne ako v predchádzajúcom prípade je význam taký, že používateľ chce získať dokumenty, ktoré sa podobajú predloženému dokumentu.

Výsledky vyhľadávania sú pri všetkých typoch dotazov totožné. Zobrazené sú základné informácie o dokumentoch (poloha, nadpis, súhrn). V predloženej verzii sa negeneruje kontextová ukážka dokumentu. Dokumenty sú zoradené zostupne podľa podobnosti.

4. Nastavenie multiplikátorov DIS

Mulitiplikátory DIS slúžia na zvýhodnenie, prípadne znevýhodnenie výsledkov vyhľadávania niektorého DIS. Hodnota multiplikátora je z intervalu $\langle 0,1 \rangle$, kde 0 znamená, že výsledky DIS sa do celkového výsledku nezapočítavajú vôbec. To znamená, že ak sú hodnoty všetkých multiplikátorov okrem jedného rovné 0, na výstupe budú len výsledky toho jedného DIS. Používateľ nimi teda môže nájsť takú kombináciu vyhľadávacích metód, ktorá mu subjektívne najviac vyhovuje.

Multiplikátory jednotlivých DIS je možné nastaviť manuálne na záložke Config, prípadne využiť automatické nastavenie podľa výsledkov vyhľadávania. Autmatické nastavenie multiplikátorov sa spúšťa tlačidlom *Make DIS multiplicator*, ktoré sa zobrazí pod výsledkom vyhľadávania organizovanom ako štruktúra. Od používateľa sa očakáva, aby označil všetky dokumenty, ktoré považuje za relevantné a výber potvrdil spomínaným tlačidlom. Na základe tohto používateľského vstupu je možné upraviť multiplikátory tak, že metódy, ktoré vrátili práve dokumenty označené používateľom ako relevantné, sú zvýhodnené. A naopak, metódy, ktoré vrátili podľa používateľa nerelevantné dokumenty sú znevýhodnené.

Príloha B – CD ROM

Obsah CD ROM:

- Inštalačný balíky jednotlivých modulov systému
- Spoločný inštalačný balík vhodný na predvádzanie (obsahuje aj ukážkové dáta a konfiguračné súbory)
- Zdrojové kódy všetkých modulov systému
- Automatická programátorská dokumentácia – JavaDoc