VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMEDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

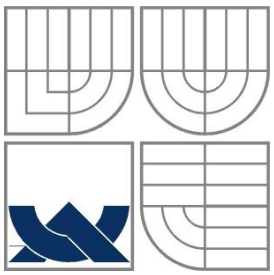
SÉMANTICKÁ BLÍZKOST PRO VĚDECKÉ ČLÁNKY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

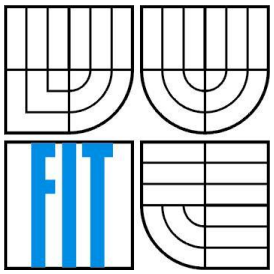
AUTOR PRÁCE
AUTHOR

Bc. ERIK DRESTO

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SÉMANTICKÁ BLÍZKOST PRO VĚDECKÉ ČLÁNKY

SEMANTIC RELATEDNESS OF SCIENTIFIC ARTICLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ERIK DRESTO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUBOMÍR OTRUSINA

BRNO 2011

Abstrakt

Hlavním cílem této práce je prozkoumat základní metody používající se k hledání sémantické blízkosti pro vědecké články. Jednotlivé metody budou podrobně vysvětleny, porovnány a ve výsledku ohodnoceny podle úspěšnosti. Na základě získaných znalostí bude navržena nová metoda pro výpočet podobnosti vědeckých článků, která by měla předčit ostatní dostupné metody tím, že spojí to nejlepší v dostupných algoritmech a přidá důležitý faktor pro podobnost, a to citace. Citace je důležitá z důvodu, že se jedná o statickou vazbu mezi články. Závěrem bude vytvořený algoritmus otestován na reálných testovacích datech a výsledky budou vyhodnoceny v porovnání s dostupnými metodami.

Abstract

The main goal of the thesis is to explore basic methods which can be used to find semantically related scientific articles. All the methods are explained in detail, compared and in the end evaluated by the standard metrics. Based on the evaluation, a new method for computing semantic similarity of scientific articles is proposed. The proposed method is based on the current state-of-the-art methods and adds the another important factor for computing similarity – citations. Using citations is important, since they represent a static bond between the articles. Finally, the proposed method is evaluated on the real data and compared with other described methods.

Klíčová slova

vědecké články, klíčová slova, sémantická blízkost, citace, random indexing, LSA, průnik termínů

Keywords

scientific articles, keywords, semantic similarity, citation, random indexing, LSA, term intersection

Citace

Dresto Erik: Sémantická blízkost pro vědecké články, diplomová práce, Brno, FIT VUT v Brně, 2011

Sémantická blízkost pro vědecké články

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Lubomíra Otrusinu.
Další informace mi poskytli členové projektu ReReSearch.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Erik Dresto
25. května 2011

Poděkování

Za odborné vedení a cenné rady děkuji vedoucímu diplomové práce Ing. Lubomíru Otrusinovi a taktéž bych se chtěl poděkovat všem členům, který se podílejí na projektu ReReSearch.

Obsah

1 Úvod

Vedecký článok je vedecko-náučný útvar, ktorý v písanej forme sprostredkúva originálne výsledky vedeckovýskumnej činnosti. Typicky sú publikované v angličtine, ale je možné nájsť mnoho publikácii aj v ďalších jazykoch. Jednotlivé publikácie sú najčastejšie zverejňované v špeciálnych vedeckých časopisoch, odborných knihách, v rámci rôznych akademických konferencii alebo na Internete. Práca popisuje spôsob návrhu systému, ktorý by uľahčil vyhľadávanie jednotlivých vedeckých článkov na základe ich sémantickej podobnosti.

Základom pre vyhľadávanie podobných vedeckých článkov sú kľúčové slová. Kľúčové slová sú v publikáciách použité pre jednoduchšiu orientáciu, alebo pre výstižné vyjadrenie obsahu. Majú typicky informačný charakter. Nie je pravidlom, že každá publikácia obsahuje ich explicitný zoznam. V prípade, že dokument neobsahuje kľúčové slová, je nutná ich ručná extrakcia z dostupného textu. Tento proces je časovo veľmi náročný a zároveň veľmi dôležitý. Presné, charakteristické a výstižné kľúčové slová zaručujú kvalitné výsledky. V ideálnom prípade sa jedná o zoznam základných a najdôležitejších pojmov, ktorými sa daný dokument zaoberá. Reálne sa medzi kľúčovými slovami vyskytujú pojmy, ktoré nie sú charakteristické pre článok samotný, ale napríklad pre jeho zaradenie do problematiky alebo oboru.

Dokumenty, ktoré budú v rámci systému porovnávané so vstupným článkom, je nutné nejakým spôsobom udržiavať. Najjednoduchším spôsobom je ich indexovanie. Je to proces, ktorý za pomoci zoznamu kľúčových slov alebo extrakcie kľúčových slov vie uložiť všetky dôležité informácie do tzv. indexu. Index je dátová štruktúra, ktorá obsahuje skomprimované dáta a umožňuje rýchle a efektívne vyhľadávanie nad nimi. V praxi si môžeme index predstaviť ako databázu, ktorá obsahuje všetky požadované informácie a na základe identifikátoru článku nám ich sprístupní.

Posledným krokom je samotné porovnávanie dokumentov a určenie tých, ktoré majú medzi sebou najväčšie sémantické väzby, a tým odpovedajú najpodobnejším. Úspešnosť tohto procesu záleží na zvolenom algoritme, ktorý určuje sémantickú podobnosť. Pokročilé algoritmy sú typicky náročné na požadované prostriedky, či už čas alebo potrebný výpočetný výkon. Cieľom práce je zanalyzovať jednotlivé dostupné metódy a na základe týchto poznatkov navrhnúť novú metódu, ktorá by vo výsledku mala mať lepšiu úspešnosť. Takisto je nutné jednotlivé metódy vzájomne porovnať. Je nutné zahrnúť nielen ich výsledky, ale aj potrebné prostriedky, ktoré boli použité počas testovania. Pre objektívne porovnanie výsledkov jednotlivých algoritmov je nutné vytvoriť testovaciu sadu, ktorej dokumenty obsahujú známe sémantické väzby.

Text diplomovej práce je rozdelený do siedmych hlavných častí. Prvou kapitolou, ktorú práve čítate, je úvod. V ďalšej kapitole je popísaný teoretický úvod do vyhľadávania sémanticky podobných dokumentov, proces predspracovania článkov pred samotným vyhľadávaním. Tretia kapitola popisuje jednotlivé dostupné algoritmy, ktoré tvoria základ pre mnou navrhnutý algoritmus. Záverom tejto

kapitoly je vysvetlená teória mnou navrhnutej metódy, ktorá je založená na citáciách. Ako štvrtá v poradí je kapitola, ktorá sa zaoberá návrhom systému pre vyhľadávanie sémanticky podobných článkov. Nasledovaná kapitolou, ktorá sa venuje samotnej implementácii. Šiesta kapitola sa zaoberá tvorbou testovacej sady, ktorá sa použije pri objektívnom testovaní a porovnaní implementovaných metód. Ďalej sa zaoberá návrhom testov, samotným testovaním a ich interpretáciou. Poslednou kapitolou je záver, ktorý hodnotí výsledky testov a zhodnocuje porovnanie jednotlivých spomínaných metód pre sémantické vyhľadávanie podobných vedeckých článkov.

Práca okrem zoznamu literatúry obsahuje aj tri prílohy. Prvou prílohou je príručka predspracovania dokumentov, ktorá popisuje spôsob prevodu vedeckých článkov do jednotného formátu, ktorý je akceptovaný systémom pre indexovanie. Druhou prílohou je príručka pre konfiguráciu a používanie systému pre indexovanie. Poslednou prílohou je kompaktný disk, ktorý okrem iného obsahuje zdrojové kódy systému pre indexovanie a predspracovania dokumentov. Súčasťou je aj vytvorená testovacia sada, programová dokumentácia vo forme *JavaDoc* a výsledky testov vykonaných v rámci tejto práce.

2 Predspracovanie dokumentov

Úvodom tejto kapitoly sú popísané používané praktiky pri rozpoznávaní a extrakcii kľúčových slov. Je nutné si uvedomiť, ktoré slová sú pre nás dôležité a budú zaradené medzi termíny. Často sa jedná o jednoslovné slová, tzv. unigramy, ktoré sú väčšinou podstatné mená v základnom tvare. Takisto medzi kľúčové slová patria aj viacslovné výrazy. Typicky sa jedná o rozšírenie jednoslovného termínu v základnom tvare o prídavné meno. Ďalšia časť objasňuje význam procesu indexácie a index samotný.

Táto kapitola popisuje praktiky, ktoré sú používané v procese predspracovania dát pre sémantické vyhľadávanie. Jeho vstupom je množina zdrojových dokumentov, ktoré sú typicky v digitálnom formáte *pdf* a výstupom je vytvorený index, ktorý obsahuje všetky potrebné dáta pre sémantické vyhľadávanie nad danou množinou dokumentov.

2.1 Rozpoznávanie a extrakcia termínov

V procese vyhľadávania podobných vedeckých článkov sú pre nás veľmi dôležité termíny. Termín je slovo alebo slovné spojenie, ktoré sa používa v špecifickom kontexte. Preto výber kľúčových slov, ktoré sú pre daný dokument najcharakteristickejšie je veľmi dôležitý. Na základe jednotlivých termínov môžeme určiť aj kontext dokumentu. Čím presnejšie a výstižnejšie termíny použijeme, tým charakteristickejší kontext pre článok získame. Samotné vyhľadávanie podobných článkov je založené na hľadaní dokumentov s rovnakým kontextom, čiže zaoberajúce sa rovnakou problematikou.

Vstupom rozpoznávacieho procesu je textový dokument. V prípade, že tento dokument je v inom formáte (napríklad *pdf*), je nutné ho najprv previesť do textovej podoby. Prvým krokom je lexikálna analýza, za pomoci ktorej prebieha identifikácia slov a viacslovných výrazov. Pre identifikáciu sa typicky používa proces tokenizácie, ktorý rozdeľuje text na menšie zmysluplné logické celky, v našom prípade slová alebo súslovia, ktoré sú označované ako tokeny. Typickým oddeľovacím znakom je medzera, ale to nemusí platiť vo všetkých prípadoch. Často je možné nájsť v dokumente vymenovanie dôležitých termínov, ktoré sú oddelené čiarkami alebo inými bežne zaužívanými znakmi. Takisto môže dochádzať k zlej interpretácii slov v prípade, ak sú oddelené spojovníkom. Nevieme určiť, či sa jedná o jedno alebo dvoj slovný výraz. Ďalej je jednoduché si pomýliť bodku ako znak konca vety s bodkou, ktorá ukončuje nejakú skratku. Pri rozpoznávaní viacslovných výrazov narážame na závažnejšie problémy ako u slov. Jedná sa hlavne o to, či musia jednotlivé slová nasledovať bezprostredne za sebou alebo môžu obsahovať medzery. Pre uľahčenie tohto procesu sa často využíva slovník súsloví. Výstupom tohto kroku je množina potenciálnych slov a viacslovných výrazov.

Nasledujúcim krokom je odstránenie slov, ktoré nenesú význam. Jedná sa hlavne o spojky, častice, citoslovcia a ďalšie výrazové prostriedky, ktoré obohacujú písaný text, ale pre nás sú nepodstatné. Toto filtrovanie je typicky založené na určovaní slovného druhu, ale môže byť ovplyvnené aj slovníkom, ktorý obsahuje nežiaduce slová, takzvaný stop-list. Výstupom je množina potenciálnych slov a viacslovných výrazov, ktoré nesú význam dokumentu.

Ako už bolo spomínané vyššie, kľúčové slová sú typicky podstatné mená v základom tvare. Avšak v bežnom písanom texte sa vyskytujú v rôznych pádoch alebo časoch. Množinu potenciálnych termínov je teda nutné previesť do nominatívu jednotného čísla za pomoci redukcie, ktorá sa nazýva lematizácia. Program, ktorý sa zoberá touto problematikou je označovaný ako lematizátor a môže používať rôzne metódy určovania základného tvaru.

Voliteľným krokom je porovnanie potenciálnej množiny termínov so slovníkom, kde dôjde k odstráneniu neexistujúcich slov, poprípade slov, ktoré sú označené ako slangové, vulgárne alebo iné. Vo vedeckých článkoch sa táto technika typicky nepoužíva, pretože by mohlo dôjsť k odstráneniu slov, ktoré označujú názvy metód a praktík.

V tomto bode procesu už je vytvorená solídna množina kľúčových slov v základnom tvare, ktorá vznikla ako výstup z posledného kroku analýzy potenciálnej množiny termínov. Teraz je nutné rozdeliť termíny na tie, ktoré sú významnejšie a viac súvisia s daným článkom, a tie, ktoré sa do užšieho výberu dostali z iných dôvodov. Toto rozdelenie sa nazýva ohodnotenie váhy a určuje relatívnu hodnotu významu daného výrazu. Existujú rôzne kritéria, podľa ktorých môžeme získať čo najpresnejšie výsledky. Jedná sa najmä o metódy založené na frekvencii výskytu alebo umiestnenia v texte. Správne kritérium je veľmi dôležité a závisí na použitých vstupných dátach. Ak máme napríklad neštruktúrovaný text, tak je zbytočné hľadať váhu na základe umiestnenia, ale lepšie je zvoliť frekvenciu výskytu.

Metóda založená na frekvencii výskytu termínov (term frequency – tf) patrí medzi základné kritéria pre určovanie váhy. Jedná sa o algoritmus, ktorý počíta výskyty termínov v dokumente a na základe tejto štatistiky určuje slová, ktoré sa najčastejšie používali [19].

Ďalším typickým zástupcom metód pre určovanie váhy je algoritmus založený na základe frekvencie výskytu termínov, ktorý ho rozširuje o vlastnosť ohodnotenia váhy slova pre daný dokument na základe aktuálneho korpusu. Táto metóda je označovaná ako tf-idf (term frequency – inverse document frequency). Hlavným princípom je, že dôležitosť slova stúpa priamo úmerne s počtom výskytov v rámci dokumentu, ale udáva sa ako ofset výskytu slova v korpuse [19].

Na konci tohto procesu je získaná množina kľúčových slov, ktorá je zoradená podľa váhy jednotlivých termínov. Typicky z tejto množiny je vybraných prvých N termínov, ktoré sú najviac charakteristické pre daný článok a na ich základe dôjde k vykonaniu vyhľadávania podobnosti.

2.2 Indexácia

Pre uľahčenie vyhľadávania kľúčových slov pre jednotlivé články je vhodné použiť index, ktorý funguje podobne ako databáza. Index je dátová štruktúra vytvorená na základe schémy. Schéma obsahuje informácie o jednotlivých dátach, ktoré budú spracovávané indexom. Analogicky s databázou si môžeme predstaviť, že schéma indexu obsahuje informácie o stĺpcoch tabuľky. Následne, takisto ako u databáz, umožňuje pomocou jednoduchých príkazov do indexu vkladať, editovať a mazať dáta.

Hlavným významom indexu je veľká optimalizácia na rýchlosť vyhľadávania informácii. Bez indexu by bolo nutné pri každom použití kľúčových slov vyhľadať súbor, v ktorom sú uložené, následne ho otvoriť a prečítať. Je známe, že vstupno-výstupné operácie sú veľmi pomalé, a tým by dochádzalo k veľkému zväčšeniu časovej náročnosti.

Samotný proces indexácie prebieha v niekoľkých krokoch. Prvým krokom je zostavenie príkazu na vytvorenie nového záznamu v indexe na základe dostupnej schémy. Pre každý dokument je vytvorený samostatný príkaz, ktorý bude obsahovať potrebné informácie ako názov článku, kľúčové slová, abstrakt, dostupný text a iné.

Ďalej dôjde k vykonaniu jednotlivých príkazov, kde sa najprv overí štruktúra schémy a následne sa spracujú potrebné dáta. Týmto ale ešte nedochádza k samotnej indexácii, iba k uloženiu dát do medzipamäte. Táto medzipamäť je používaná z dôvodu paralelizmu, aby nedochádzalo k sekvenčnému spracovaniu požiadaviek.

Poslednou fázou je takzvaný „Commit“, ktorý prevedie momentálny obsah medzipamäte do indexu. Celý proces je založený na sekvenčnom spracovaní jednotlivých položiek. Pri vzniku duplicitných dát môže podľa nastavenia dochádzať k zahadzovaniu nových informácii alebo ich prepisovaniu. Po úspešnom doplnení informácii dôjde k overeniu integrity, optimalizácii a kompresii. Výsledkom je index pripravený na použitie, ktorý na základe presných požiadaviek dokáže generovať odpovede v ráde milisekúnd.

3 Metódy pre výpočet sémantickej blízkosti

V tejto kapitole sú predstavené jednotlivé metódy pre výpočet sémantickej blízkosti vedeckých článkov. Objasňuje ich základné princípy a takisto aj predpokladané výsledky na základe dostupných informácii. Záverom kapitoly je popísaná mnou navrhnutú metódu vyhľadávania podobnosti, ktorá je založená na citáciách. Objasníme si dôvod, prečo boli zvolené práve citácie a takisto techniku počítania podobnosti na ich základe.

3.1 Prienik termínov

Prienik termínov je možné zaradiť medzi najjednoduchšie metódy pre vyhľadávanie sémantickej podobnosti. Tento algoritmus sa použije ako referenčný pre porovnanie voči ostatným metódam. Jeho princíp je veľmi jednoduchý, takisto náročnosť na výpočetný výkon a čas je nízka, má však nízku úspešnosť určovania sémantickej podobnosti.

Je založený na porovnávaní množín kľúčových slov pre jednotlivé dokumenty. Hlavnou myšlienkou je, že dokumenty obsahujúce rovnakú množinu kľúčových slov sa zaoberajú tou istou problematikou, a tým sa určuje sémantická podobnosť. Tu samozrejme vzniká veľká závislosť na zvolených termínoch. Ak sa nebude jednať o kvalitnú množinu, tak dôjde k negatívnemu skresleniu výsledkov. Takisto početnosť výskytov jednotlivých kľúčových slov hrá dôležitú rolu, pretože vo výsledku sa nepočíta len veľkosť množiny prieniku, ktorá môže byť u viacerých dokumentov zhodná, ale pre presnejšie určenie podobnosti sa využije pomer počtu výskytov a veľkosti množiny prieniku.

Medzi hlavné výhody tejto metódy je možné zaradiť najmä rýchlosť, ktorá je natoľko vysoká, že v niektorých prípadoch kompenzuje nižšiu úspešnosť určovania sémantickej podobnosti oproti algoritmom s vyššou náročnosťou. Ďalším pozitívom je výpočet sémantickej podobnosti v prípade modifikácie korpusu, kde stačí dopočítať podobnosť nového dokumentu s existujúcimi článkami a následne ho vložiť na správne miesto do zostupného lineárneho zoznamu zoradeného podľa podobnosti. Takisto na základe údajov o podobnosti s jednotlivými dokumentmi je pre tento nový článok vytvorený rovnaký zostupný lineárny zoznam podobnosti.

Hlavnú nevýhodu predstavuje nízka úspešnosť určovania sémantickej blízkosti. Je to spôsobené najmä tým, že táto metóda sa nezaoberá kontextom, v ktorom boli jednotlivé kľúčové slová použité, a tým môže dochádzať k skresleniu výsledkov.

3.2 Latentná sémantická analýza

Latentná sémantická analýza je technika, ktorá sa zaoberá analýzou vzťahov medzi dokumentmi a slovami. V kontexte aplikácie pre získavanie informácii často býva nazývaná ako latentná sémantická indexácia. Podľa tejto teórie sa slová, ktoré majú podobný význam, vyskytujú v rovnakom kontexte, z čoho vyplýva, že s rovnakými slovami [15].

Nejedná sa o tradičnú metódu spracovania prirodzeného jazyka, pretože nevyužíva slovníky, gramatiky, sémantické siete alebo morfológiu, ale len čistý text, ktorý je zložený zo slov rozdelených do zmysluplných častí ako vety alebo odseky. Využíva distribučnú štatistiku na vytvorenie mnohodimenzionálneho vektorového priestoru. Jednotlivé články sú reprezentované ako kontextové vektory. Kontextový vektor reprezentuje kontext článku na základe kontextov jednotlivých termínov, ktoré obsahuje. Pre určovanie podobnosti jednotlivých slov je použitá relatívna smernica vektoru, ktorá udáva mieru kontextu, v ktorej sa nachádzajú [15].

Základom je transformácia textu dokumentu na štruktúru, ktorá predstavuje maticu, kde každý riadok predstavuje jedinečné slovo a stĺpec jeho kontext. Za kontext považujeme nejaký úsek textu, typicky celý dokument. Jednotlivé bunky matice obsahujú frekvenciu výskytu slova v kontexte vyjadrenom stĺpcom. Nakoľko každý dokument má typicky inú dĺžku, čím dochádza k rozdielnym veľkostiam vektorového priestoru, môže byť táto matica normalizovaná. Táto kompenzácia je dosiahnutá jednoduchým podelením frekvencie výskytu s celkovým počtom slov v dokumente. Tým dochádza k vytvoreniu novej štruktúry matice tak, že každý riadok obsahuje vektor v mnohodimenzionálnom priestore. Obsah vektoru reprezentuje frekvenciu výskytu slov v rôznych kontextoch a počet dimenzii je rovný počtu kontextov, čiže stĺpcov matice. Takto upravený vektor vyjadruje kontext, v ktorom sa dané slovo nachádza.

Nastáva hlavný problém v tom, že testovacie sady obsahujú veľký počet článkov, z čoho vyplýva, že počet dimenzii vektora bude tiež veľký. Výpočet pre podobnosti vektorov s veľkým počtom dimenzii je veľmi časovo náročný, a tak potrebujeme znížiť počet dimenzii za pomoci transformácie na iné dimenzie, ktorých počet je pre náš výpočet prijateľný. Typicky matica kontextov obsahuje veľké množstvo hodnôt rovnajúcich sa nule, čiže môžeme o nej prehlásiť, že je riedka. To je spôsobené z dôvodu, že len malá časť slov sa vyskytuje vo veľkom počte kontextov. Na základe tejto skutočnosti môžeme redukovať počet dimenzii pomocou metódy singulárneho rozkladu (Singular Value Decomposition – SVD [12]).

Singulárny rozklad patrí medzi metódy lineárnej algebry, ktorej úlohou je zníženie dimenzii priestoru dokumentov. Umožňuje previesť túto redukciu tak, aby zachoval zhluky podobných dokumentov. Jeho definícia je: „Nech A je ľubovoľná štvorcová matica. Potom existujú ortogonálne matice U a V a diagonálne matice

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix}, \tag{3.1}$$

na ktorej diagonále sú vlastné čísla matice $\sqrt{A^T A}$ tak, že

$$A = U\Sigma V^T. \tag{3.2}$$

Tento rozklad sa nazýva singulárny rozklad a vlastné čísla matice $\sqrt{A^T A}$ sa nazývajú singulárne čísla matice *A*.“ [12]

Nakoľko matica *A* termínov v dokumentoch nie je štvorcová, ale obecne býva rádu $m \times n$, kde platí $m \neq n$. Z toho vyplýva, že *U* je ortogonálna matica $m \times m$, ktorej stĺpce definujú ľavé singulárne rozklady matice *A*, v našom prípade reprezentujú maticu vektorov. *V* je ortogonálna matica $n \times n$, ktorej stĺpce definujú pravé singulárne vektory *A* a reprezentuje maticu dokumentov. Σ je diagonálna matica $m \times n$ obsahujúca singulárne čísla $\sigma_1, \sigma_2, \dots, \sigma_{\min(m,n)}$ usporiadané zostupne na hlavnej diagonále tak, že platí $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)}$ [12].

$m > n$:

$$\begin{bmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ * & * & \dots & * \end{bmatrix} = \begin{bmatrix} * & * & \dots & \dots & \dots & * \\ * & * & \dots & \dots & \dots & * \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ * & * & \dots & \dots & \dots & * \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \dots & * \end{bmatrix} \tag{3.3}$$

$m < n$:

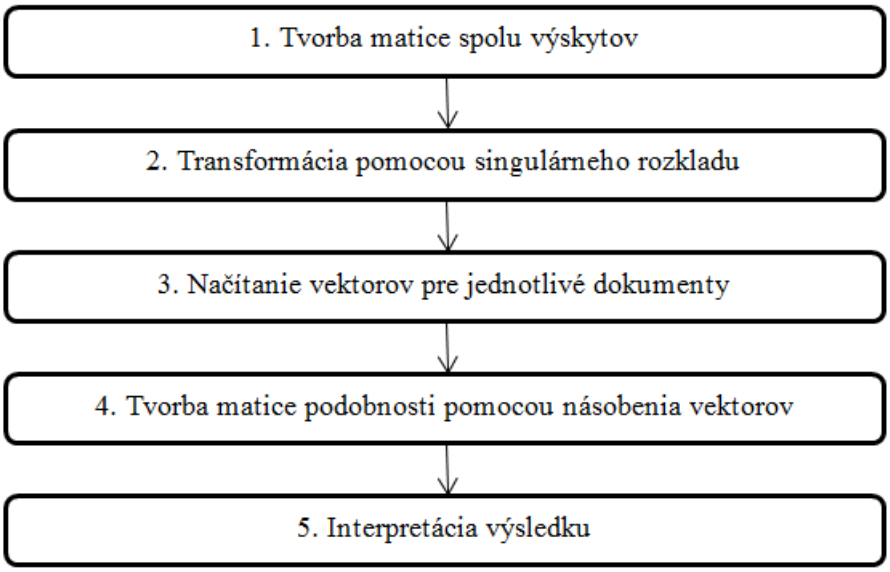
$$\begin{bmatrix} * & * & \dots & \dots & * \\ * & * & \dots & \dots & * \\ \vdots & \vdots & & & \vdots \\ * & * & \dots & \dots & * \end{bmatrix} = \begin{bmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & & \vdots \\ * & * & \dots & * \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma_n & \dots & 0 \end{bmatrix} \begin{bmatrix} * & * & \dots & \dots & * \\ * & * & \dots & \dots & * \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ * & * & \dots & \dots & * \end{bmatrix} \tag{3.4}$$

$$A(m \times n) \quad = \quad U(m \times m) \qquad \Sigma(\mathrm{m} \times \mathrm{n}) \qquad V^T(n \times n) \tag{3.5}$$

Toto zostupné radenie sa využíva v tom, že pre dostatočne presné výsledky nám stačí vypočítať prvých *k* najväčších singulárnych čísel. Takýmto spôsobom získame *k*-aproximáciu hodnosti matice *A*. Vhodnú hodnotu čísla k je nutné určiť experimentálne. Tým dochádza k zredukovaniu mnohodimenzionálneho priestoru na priestor s *k* dimenziou a zároveň dôjde k zachovaniu zhlukov podobných dokumentov a termínov.

Základom algoritmu latentnej sémantickej analýzy je vytvorenie matice spolu výskytov, kde riadky reprezentujú jedinečné slová a stĺpce reprezentujú kontexty, čiže početnosť výskytov v daných dokumentoch. Následne dôjde k prevedeniu singulárneho rozkladu, ktorý transformuje maticu spolu výskytov z dôvodu zníženia počtu dimenzii. Počet dimenzii záleží na nastavení užívateľa, typicky sa rovná počtu dokumentov v korpuse. Tým vzniknú tri matice *U*, Σ a V^T. *U* reprezentuje kontextové vektory termínov, Σ reprezentuje diagonálnu maticu, ktorá obsahuje singulárne čísla a V^T reprezentuje transponované kontextové vektory dokumentov. Následne získame výsledné kontextové

vektory dokumentov, ktoré predstavujú riadky výsledku násobenia matíc Σ a V^T zľava. Je nutné ich previesť do normalizovaného tvaru tak, aby jednotlivé dimenzie obsahovali hodnoty v rozmedzí -1 až 1. Na základe ktorých je možné vytvoriť maticu podobnosti, kde riadky a stĺpce reprezentujú dokumenty a jednotlivé bunky predstavujú hodnotu podobnosti, ktorá vznikne násobením vektorov korešpondujúcich dokumentov. Jednotlivé hodnoty matice podobnosti sú v rozmedzí -1 až 1, kde 1 predstavuje takmer totožný dokument a -1 úplne odlišný. Posledným krokom je interpretácia získaných výsledkov. Obrázok 3.1 reprezentuje diagram znázorňujúci popisovaný algoritmus.



Obrázok 3.1: Diagram znázorňujúci algoritmus latentnej sémantickej analýzy

Ako hlavná výhoda latentnej sémantickej analýzy je považované to, že je postavená na veľmi dobre matematicky popísanom vektorovom modele. To znamená, že existuje veľké množstvo matematických konštrukcii pre prácu s maticami alebo vektormi. Ďalšou podstatnou výhodou oproti konkurencii je, že sa dá veľmi dobre použiť pri hľadaní synoným. Je to spôsobené tým, že synonymá sa typicky spolu nevyskytujú, ale nachádzajú sa v rovnakom kontexte.

Medzi hlavné nevýhody patrí náročnosť na čas a výpočetný výkon. Jedná sa o jednu z náročnejších metód. Singulárny rozklad sa snaží tento aspekt vylepšiť, ale aj napriek tomu jeho zložitosť je rovná $O(\min(mn^2, m^2n))$ [6]. Napriek tomu ako pozitívum je hodnotené, že singulárny rozklad stačí vypočítať len raz, a potom s ním pracovať. Čo nás privádza k ďalšiemu problému, a to je pridávanie nových kontextov. V prípade, že je potrebné rozšíriť kontexty tak je nutné opätovne prepočítať kompletný singulárny rozklad. Toto veľké negatívum je v súčasnosti čiastočne vyriešené za pomoci použitia metód SVD-Updating [17], SVD-Updating za pomoci ortogonálnych rotácii [3] a Folding-in [23].

SVD-Updating je algoritmus, ktorý je založený na jednoduchom pridaní transponovaného riadku do už existujúceho singulárneho rozkladu. Následne je nutné dopočítať nové hodnoty na

diagonále matice Σ, čím sa získa modifikovaný singulárny rozklad bez toho aby bolo nutné prepočítať všetky kontexty. Táto metóda má však problém pri zaokrúhľovaní alebo orezaní jednotlivých hodnôt, pretože dochádza k poškodeniu ortogonality jednotlivých kontextových vektorov. Túto skutočnosť sa snaží riešiť metóda SVD-Updating za pomoci ortogonálnych rotácii. Riešenie je založené na základe parametrizácie jednotlivých pravých vektorov ortogonálnej matice jednoduchými rotáciami a v procese výpočtu nových diagonálnych hodnôt dochádza k modifikácii ich uhlov. Čím sú získané presnejšie údaje, ktoré nie sú ovplyvnené zaokrúhlením alebo orezaním.

Folding-in predstavuje jednoduchú metódu založenú na projekcii nových riadkov na koniec matice *U*, čím sú premietnuté do priestoru redukovaných dokumentov a termínov. Čo spôsobuje premietnutie stavu singulárneho rozkladu do týchto nových polí. Táto metóda má nevýhodu v tom, že nie je možné premietnuť stav nových vektorov do existujúceho rozkladu.

3.3 Random indexing

Random indexing bol vytvorený ako alternatíva k latentnej sémantickej analýze, ktorý predstavuje inkrementálny vektorový model. Je založený na práci Pentii Kanervu, ktorý sa zaoberal riedkymi distribučnými reprezentáciami [10]. Motiváciou pre základnú myšlienku tejto metódy bolo pozorovanie Roberta Hecht-Nielsena, ktorý demonštroval, že vo viacdimenzionálnom priestore sa typicky vyskytujú približne ortogonálne smery viac ako tie skutočné [5]. Tým prišiel Pentii Kanerva k záveru, že môžeme použiť náhodné smery, aby sme dosiahli vhodnú aproximáciu ortogonality [21].

Táto myšlienka bola základom pre rodinu techník, ktorých cieľom je redukcia počtu dimenzií. Medzi hlavných predstaviteľov tejto skupiny patria Random Projection, Random Mapping a Random Indexing. Všetky tieto metódy sú založené na Johnson-Lindenstraussovej lemme, ktorá hovorí, že: „Keď premietneme body vektorového priestoru na náhodne vybratý podpriestor s dostatočne veľkým počtom dimenzií, vzdialenosti medzi jednotlivými bodmi budú približne zachované.“ [21]. Preto je možné počet dimenzií matice *F* redukovať za pomoci vynásobenia s náhodnou maticou *R*:

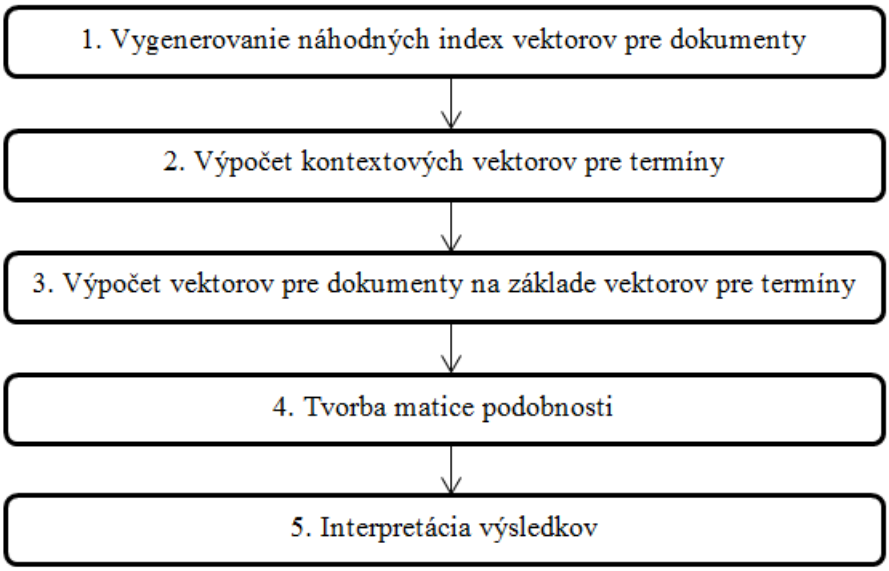
$$F_{w\times d}R_{d\times k} = F'_{w\times k} \tag{3.6}$$

Samozrejme výber náhodnej matice *R* je veľmi dôležitý. Ak by náhodné vektory, ktoré matica R obsahuje boli ortogonálne, čiže $R^T R = I$, tak by platilo, že $F = F'$. Poprípade ak by vektory boli približne ortogonálne, tak výsledkom by bolo $F \approx F'$. Najčastejšie sa používa náhodná matica, ktorá obsahuje náhodné vektory vytvorené pomocou Gaussovho rozloženia elementov. Avšak existuje jednoduchšia metóda, ktorú navrhol Dimitris Achlioptas. Hlavnou myšlienkou je, že takmer všetky elementy týchto náhodných vektorov budú nulové, čo znamená rozloženie s jednotkovou variáciou tak, aby to vyhovovalo podmienkam lemmy [21].

Základná technika random indexing pozostáva v dvoch krokoch. Prvý krok vytvára pre každý dokument unikátnu a náhodne vytvorenú reprezentáciu, ktorá sa nazýva index vektor. Index vektor je riedky, viacdimenzionálny a ternárny. Obsahuje malé množstvo náhodne rozložených 1 a -1, ostatné

prvky sú 0. Druhým krokom je postupné prechádzanie množiny termínov pre daný dokument a vždy, keď sa vyskytne v kontexte, je ku kontextovému vektoru pre daný dokument pripočítaný index vektor kontextu. To znamená, že dokumenty sú reprezentované kontextovými vektormi, ktoré sú v podstate súčtom index vektorov tých kontextov, v ktorých sa vyskytovali termíny. Kontexty, v ktorých sa vyskytujú termíny, sú typicky dokumenty alebo slová, ale samozrejme je možné využiť aj iné druhy kontextov [21].

V tejto práci je použitá pozmenená verzia random indexingu, ktorá vychádza z popisovanej základnej techniky, ale pracuje v troch krokoch. Táto nová verzia bola prezentovaná v roku 2008 ako súčasť knižnice semanticvectors [24]. Prvý krok je rovnaký, dôjde k vygenerovaniu náhodných index vektorov, ktoré budú reprezentovať jednotlivé dokumenty korpusu, kde náhodný index vektor predstavuje upravený nulový vektor tak, že na náhodných indexoch je vložená hodnota 1 alebo -1. Druhým krokom je výpočet kontextových vektorov pre jedinečné termíny tak, že pre každý výskyt termínu v článku dôjde k pripočítaniu súčinu index vektoru dokumentu a počtu jeho výskytov. V treťom kroku dôjde k prechádzaniu množiny kontextových vektorov pre jedinečné termíny tak, že ak článok obsahuje daný termín, dôjde k ovplyvneniu index vektoru dokumentu, pomocou pričítania súčinu kontextového vektoru termínu a počtu jeho výskytov. Tým získame kontextové vektory pre jednotlivé dokumenty, ktoré sú reprezentované súčtom náhodného index vektoru a kontextových vektorov obsiahnutých termínov. Je nutné ich previesť do normalizovaného tvaru tak, aby jednotlivé dimenzie obsahovali hodnoty v rozmedzí -1 až 1. Na základe týchto kontextových vektorov je možné vytvoriť maticu podobnosti, kde riadky a stĺpce reprezentujú dokumenty a jednotlivé bunky predstavujú hodnotu podobnosti, ktorá vznikne násobením korešpondujúcich vektorov. Jednotlivé hodnoty matice podobnosti sú v rozmedzí -1 až 1, kde 1 predstavuje takmer totožný dokument a -1 úplne odlišný. Posledným krokom je interpretácia získaných výsledkov. Obrázok 3.2 predstavuje diagram znázorňujúci popisovaný algoritmus.



Obrázok 3.2: Diagram znázorňujúci algoritmus random indexing

Tento prístup je presne opačný ako u latentnej sémantickej analýze, kde ako prvé dochádza k vytvoreniu matice výskytov a potom z nej sú extrahované jednotlivé kontextové vektory. Random indexing najprv vytvorí jednotlivé kontextové vektory a potom vytvorí maticu výskytov tak, že riadky matice predstavujú kontextové vektory. Takto vytvorená matica bude aproximáciou matice, ktorá by vznikla pomocou metódy latentnej sémantickej analýzy. Základný rozdiel týchto matíc je v počte dimenzií vektorového priestoru. Pri random indexingu závisí počet dimenzií vektorového priestoru na dimenzií index vektorov, ktoré bývajú v rádoch stoviek alebo tisícov. Latentná sémantická analýza závisí na počte kontextov, ktorých môžu byť rádovo až milióny. Tým pri random indexingu odpadá nutnosť použitia singulárneho rozkladu, ktorý je veľmi náročný. Výsledkom je podobná matica, ktorá by vznikla za pomoci metódy latentnej sémantickej analýzy.

Čo nás privádza k hlavným výhodám metódy random indexing. Ako prvou je získanie podobných výsledkov ako u latentnej sémantickej analýzy, ale za kratší čas a pri menších nárokoch na výpočetný výkon. Čo je spôsobené tým, že nie je nutné využívať náročnú metódu singulárneho rozkladu z toho dôvodu, lebo výsledná matica obsahuje prijateľný počet dimenzií. Ďalšou veľmi dôležitou výhodou je pridávanie nových kontextov bez toho, aby sme museli vykonávať náročné operácie ako je singulárny rozklad. Pri pridaní nového kontextu stačí vytvoriť index vektor pre daný kontext a pripočítať ho ku všetkým kontextovým vektorom, ktoré sa v danom kontexte vyskytujú. Takisto nedochádza k zvyšovaniu dimenzií pri pridávaní nových kontextov, nakoľko počet dimenzií závisí na index vektore, ktorého počet dimenzií sa nastavuje na začiatku výpočtu. Takisto využitie tejto metódy nie je limitované na určitý druh kontextu a je ju možné používať s ľubovoľným druhom kontextu.

Medzi hlavné nevýhody metódy random indexing patrí, že výsledný matica je len približne ortogonálna. Čiže v skutočnosti sa táto matica môže líšiť, ale na základe experimentov [9] sa podobá natoľko skutočnosti, že výsledky sú porovnateľné s výsledkami metódy latentnej sémantickej analýzy.

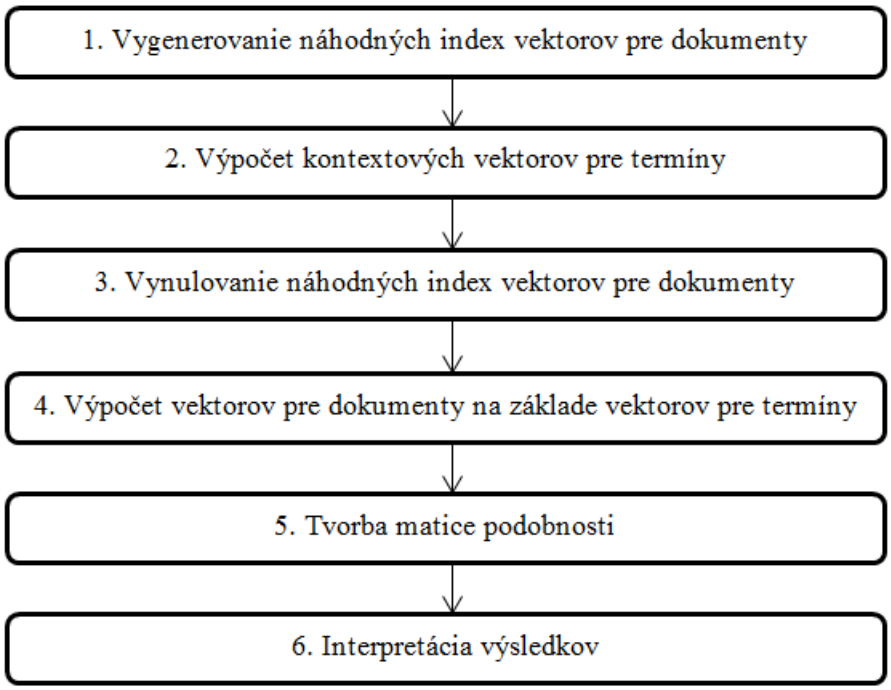
3.4 Modifikovaný random indexing

Táto metóda funguje na rovnakom princípe ako pozmenená verzia random indexingu obsiahnutá v knižnici semanticvetors. Z definície random indexingu vyplýva, že táto metóda funguje nielen pre dokumenty, ale pre ľubovoľné kontexty, ktoré môžu byť reprezentované napríklad pomocou kľúčových slov [21]. Modifikácia spočíva v spôsobe akým vytvára jednotlivé kontextové vektory pre články. Každý dokument je takisto reprezentovaný náhodným index vektorom, ale líši sa ich nasledovné spracovanie.

Tento princíp dokáže lepšie aproximovať sémantické vzťahy medzi jednotlivými kľúčovými slovami a dokumentmi. Každý článok je reprezentovaný náhodným index vektorom a každý termín je reprezentovaný vlastným nulovým kontextovým vektorom. Následne dochádza k vypočítaniu

kontextu jednotlivých kľúčových slov, ktorý reprezentuje kontext v rámci jednotlivých dokumentov na základe náhodného index vektoru pre článok, v ktorom sa nachádzajú, a ich početnosti výskytu. Takto vytvorené kontextové vektory je nutné upraviť do normalizovaného tvaru, aby jednotlivé dimenzie obsahovali hodnoty v rozmedzí -1 až 1. Tým sme vytvorili množinu vzťahov jednotlivých kľúčových slov a dokumentov na základe kontextu výskytu [21].

Následne je nutné prepočítať túto množinu na vzťahy medzi jednotlivými článkami, ktoré budú reprezentovať výsledok. Rozdielom oproti klasickému random indexingu je, že v prvom kroku dôjde k nahradeniu náhodných index vektorov za nulové kontextové vektory, čím dôjde k minimalizácii náhodnej zložky. Pokračuje sa tým, že je prechádzaná celá množina kontextových vektorov kľúčových slov a pre každý termín sa vyhľadá množina dokumentov, v ktorých sa nachádza. Následne dochádza k upraveniu kontextových vektorov pre jednotlivé články na základe kontextových vektorov termínov a ich početnosti výskytu v rámci daného dokumentu. Týmto procesom dochádza k získaniu výslednej množiny kontextových vektorov pre jednotlivé články, ktorá reprezentuje sémantické vzťahy medzi nimi vo viacdimenzionálnom priestore. V poslednom kroku musí byť táto množina normalizovaná tak, aby jednotlivé dimenzie obsahovali hodnoty v rozmedzí -1 až 1. Obrázok 3.3 predstavuje diagram znázorňujúci popisovaný algoritmus.



Obrázok 3.3: Diagram znázorňujúci algoritmus modifikovaného random indexingu

Hlavnou výhodou tejto metódy je, že dokáže presnejšie aproximovať výsledky metódy latentnej sémantickej analýzy ako klasický random indexing. Takisto medzi výhody môžeme zaradiť všetky vlastnosti klasického random indexingu, ako úprava korpusu, respektíve pridávanie nových

kontextov, vopred definovaný počet dimenzií, ktorý nevyžaduje použitie metódy singulárneho rozkladu.

Medzi nevýhody oproti klasickému random indexingu môžeme zaradiť vyššiu náročnosť na výpočetný výkon. Čo vyplýva z toho, že je nutné vytvárať novú nulovú množinu kontextových vektorov, ktoré sa použijú pre dokumenty. Ako nevýhodu je nutné opäť uviesť, že výsledná matica predstavuje len aproximáciu matice, ktorú by sme získali pomocou metódy latentnej sémantickej analýzy.

3.5 Navrhnutá metóda pre výpočet sémantickej blízkosti

Na základe získaných znalostí je možné prehlásiť, že pre výpočet sémantickej blízkosti na základe kontextu existuje dostatočné množstvo algoritmov, ktorých výsledky sú experimentálne overené [21]. Latentná sémantická analýza a random indexing boli overené na základe testu TOEFL, ktorého základný princíp je vyhľadávanie synoným. Subjekt, v našom prípade algoritmus vyhľadávania sémantickej podobnosti, vyberá správne synonymum pre požadované slovo na základe výberu zo štyroch poskytnutých slov. Úspešnosť latentnej sémantickej analýzy bola na základe testov Thomasa Landauera a Susan Dumaisovej z roku 1997 na úrovni 64.4% [14]. Skóre random indexingu na základe testov Magnusa Shalgrena a Jussiho Karlgrena z roku 2001 bolo na úrovni 64.5% až 67% [11]. Úspešnosť random indexingu sa podarilo dostať až na hranicu 82% na základe rozdelenia jednotlivých slov do kategórii, čo dokázal test z roku 2004, ktorý aplikovali Magnus Shalgren a Rickard Cöster [22].

Výsledky týchto algoritmov sú dostatočne úspešné a pri tvorbe novej metódy založenej na skúmaní kontextu by pravdepodobne nedošlo k rapídnemu zlepšeniu. Pretože ak by sa podarilo prekonať výsledky súčasných metód, tak by to bolo v zanedbateľných hodnotách. Lepším riešením je využiť metódu, ktorá už dosahuje veľmi dobré výsledky a vylepšiť ju pridaním nového aspektu, na základe ktorého dosiahne ideálne výsledky. Týmto novým aspektom budú citácie.

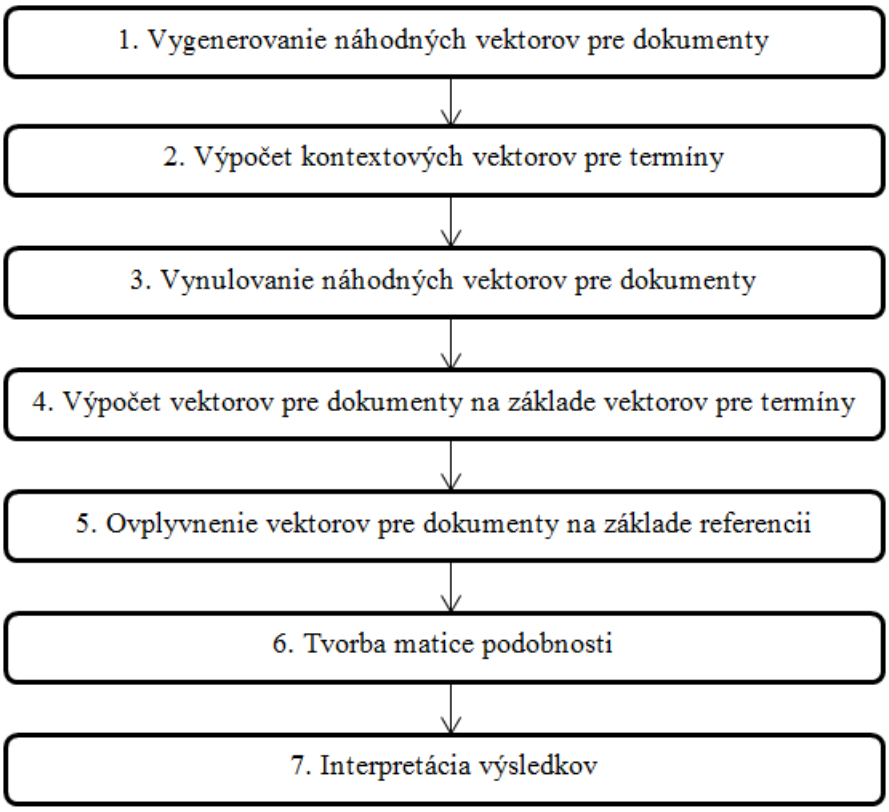
Analýza citácii má dlhú tradíciu v informačnej vede, kde je radená do odvetvia bibliometrie, teraz často nazývaná ako scientologická alebo informačná metrika. Od jej vzniku, keď Eugene Garfield vytvoril Science Citation Index na Ústave vedeckých informácii (Institute for Scientific Information) v roku 1960, sa citácie stali dôležitým prostriedkom hodnotenia vplyvu vedeckej práce. Počas vývoja a aplikácie týchto metrík vzrastalo množstvo kritík. Hlavným problémom a cieľom kritík bolo hodnotenie vedeckých prác na základe počtu citácii. Čím väčší počet referencii, tým kvalitnejšie bolo dané dielo označované. Z toho dôvodu došlo k hlbšej analýze citácii a skúmaniu ich kontextu. Vzniklo niekoľko metód, ktoré sa zaoberali skúmaním a klasifikáciou citácii, ale žiadna z nich si nenašla typické využitie v praxi [8].

Prečo práve citácie? Pretože citácia vytvára statickú väzbu medzi dvoma dokumentmi. Na základe tejto väzby je možné presnejšie určovať sémantické vzťahy jednotlivých článkov, pretože k jej tvorbe nedochádza nejakým nástrojom, ktorý automaticky tieto väzby vyhľadáva, ale je vytvorená autorom článku. Tento proces tvorby väzby dvoch dokumentov má výhodu v tom, že nedochádza k skresľovaniu vzťahov, pretože autor typicky cituje články, ktoré obsahujú informácie s rovnakou problematikou, poprípade ju rozširujú. Na základe toho môžeme predpokladať, že tieto články majú podobný kontext. Avšak táto väzba neurčuje na koľko je tento kontext podobný, bude preto nutné vedieť experimentálne odhadnúť váhu každej citácie.

Základným princípom je použitie modifikovaného random indexingu, ako metódy pre určovanie kontextu jednotlivých dokumentov a ich vzájomných vzťahov. Následne na základe citácii dôjde k ovplyvneniu týchto kontextov. Citácia je typicky jednosmerná väzba, a tým dôjde k tomu, že kontext citovaných dokumentov nebude ovplyvnený, ale ovplyvní sa kontext článkov, ktoré daný dokument citovali na základe jeho kontextu. Tým vzniká problém, čo ak citovaný dokument sa nenachádza v množine skúmaných dokumentov. Riešením je vytvorenie náhodného index vektoru, ktorý reprezentuje jeho kontext a je použitý pre ovplyvňovanie kontextu dokumentov, ktoré ho citovali. Určite je vhodné otestovať správanie algoritmu ak každú jednosmernú väzbu v podobe citácie prevedieme na obojsmernú a dôjde tak ovplyvneniu kontextu oboch dokumentov. Takisto je nutné experimentálne zistiť váhu, akú bude zohrávať citácia v ovplyvňovaní kontextu. Túto váhu môžeme odvodiť aj napríklad na základe počtu citovaných celkov z daného dokumentu, čím viac sa daný dokument cituje, tým by mal byť ich kontext podobnejší. Táto váha sa použije ako koeficient, ktorým sa budú násobiť jednotlivé dimenzie v procese ovplyvňovania kontextu, ktorý spočíva v súčte súčasnej hodnoty dimenzie a súčinu koeficientu s hodnotou rovnakej dimenzie citovaného článku.

Základom výpočtu je vytvorenie kontextových vektorov dokumentov pomocou metódy modifikovaného random indexingu. Postup tvorby týchto kontextových vektorov je popísaný v kapitole 3.4. Rozdielom je, že na konci tvorby kontextových vektorov pre dokumenty nedôjde k prevodu do normalizovaného tvaru. Je to spôsobené z dôvodu nasledovného výpočtu, ktorý bude ovplyvňovať jednotlivé kontexty na základe referencii. Následne dôjde k vytvoreniu množiny vektorov, ktorá obsahuje náhodné index vektory pre referencie, ktoré nie sú obsiahnuté v korpuse a kontextové vektory referencii nachádzajúcich sa v korpuse. Postupným prechádzaním množiny dokumentov dochádza k ovplyvneniu ich kontextu na základe pripočítania súčinu váhy podobnosti a vektoru reprezentujúceho referenciu. Tým vznikne jednosmerná väzba dokumentu a referencie, v prípade obojsmernej väzby dôjde k ovplyvneniu kontextu referencie na základe pripočítania súčiny kontextového vektoru dokumentu a váhy podobnosti. Tým sme získali výslednú množinu kontextových vektorov jednotlivých dokumentov, ktorú je nutné previesť do normalizovaného tvaru tak, aby jednotlivé dimenzie obsahovali hodnoty v rozmedzí -1 až 1. Následne je vypočítaná matica podobnosti, kde riadky a stĺpce reprezentujú dokumenty a jednotlivé bunky predstavujú hodnoty podobnosti. Táto hodnota je v rozmedzí -1 až 1, kde 1 predstavuje takmer totožný dokument a -1

úplne odlišný. Posledným krokom je interpretácia získaných výsledkov. Obrázok 3.4 predstavuje diagram znázorňujúci popisovaný algoritmus.



Obrázok 3.4: Diagram znázorňujúci algoritmus navrhnutej metódy

Na základe tejto metódy dôjde k získaniu presnejších výsledkov ako použitím metódy modifikovaného random indexingu. Náročnosť na potrebný čas a výpočetný výkon vzrastie len lineárne oproti modifikovanému random indexingu, čím by nemalo dochádzať k predraženiu ceny vyhľadávania. Ako nevýhodu môžeme považovať, že použitím citácii stratí táto metóda výhodu, ktorú obsahuje random indexing, v podobe použitia na ľubovoľné gramatické celky, ale bude primárne zameraná na vyhľadávanie sémanticky podobných dokumentov, ktoré majú presne definované citácie. Ďalším problémom, ktorý vznikne je úprava korpusu, respektíve pridávanie nových kontextov. Ako už vieme, tak random indexing tento proces umožňuje, problém je ovplyvnenie kontextu článkov, ktoré citujú dokumenty nenachádzajúce sa v korpuse. Ako bolo už uvedené, tak pre ne budú vytvorené náhodné index vektory, ktoré budú reprezentovať ich kontext. Lenže v prípade pridávania nových článkov je nutné, aby tieto náhodné vektory boli rovnaké, musí dochádzať k ovplyvneniu rovnakým kontextom, pretože sa jedná o rovnaký dokument. Riešením môže byť prepočítania všetkých kontextov za pomoci nových náhodne vygenerovaných index vektorov, alebo je možné si tieto náhodné index vektory ukladať a v prípade potreby ich použiť.

4 Návrh systému

Táto kapitola popisuje návrh systému pre vyhľadávanie sémanticky podobných vedeckých článkov. Úvodom je rozobraný návrh predspracovania dokumentov, v rámci ktorého dôjde ku konverzii jednotlivých článkov do vhodného formátu a extrakcii metainformácii, ktoré je možné z daného dokumentu získať. Medzi metainformácie patria názov, autor, konferencia, na ktorej bol publikovaný, abstrakt, kľúčové slová, text článku a zoznam literatúry. Následne je predstavený návrh systému pre indexovanie, ktorého úlohou je uchovávať všetky potrebné dáta. Záverom kapitoly sú predstavené návrhy jednotlivých metód pre vyhľadávanie sémanticky podobných dokumentov, ktoré boli popísané v kapitole 3. Každá metóda funguje ako zásuvný modul do systému pre indexovanie.

4.1 Predspracovanie vstupných dokumentov

Prvým krokom celého systému pre vyhľadávanie sémanticky podobných článkov je predspracovanie vstupných dokumentov, ktoré je nutné z dôvodu rôznorodosti jednotlivých publikácii. Nakoľko nie každý článok je v rovnakom digitálnom formáte, poprípade neobsahuje zoznam kľúčových slov, abstrakt alebo zoznam literatúry explicitne uvedený, je nutné tieto informácie získať nejakým automatizovaným procesom.

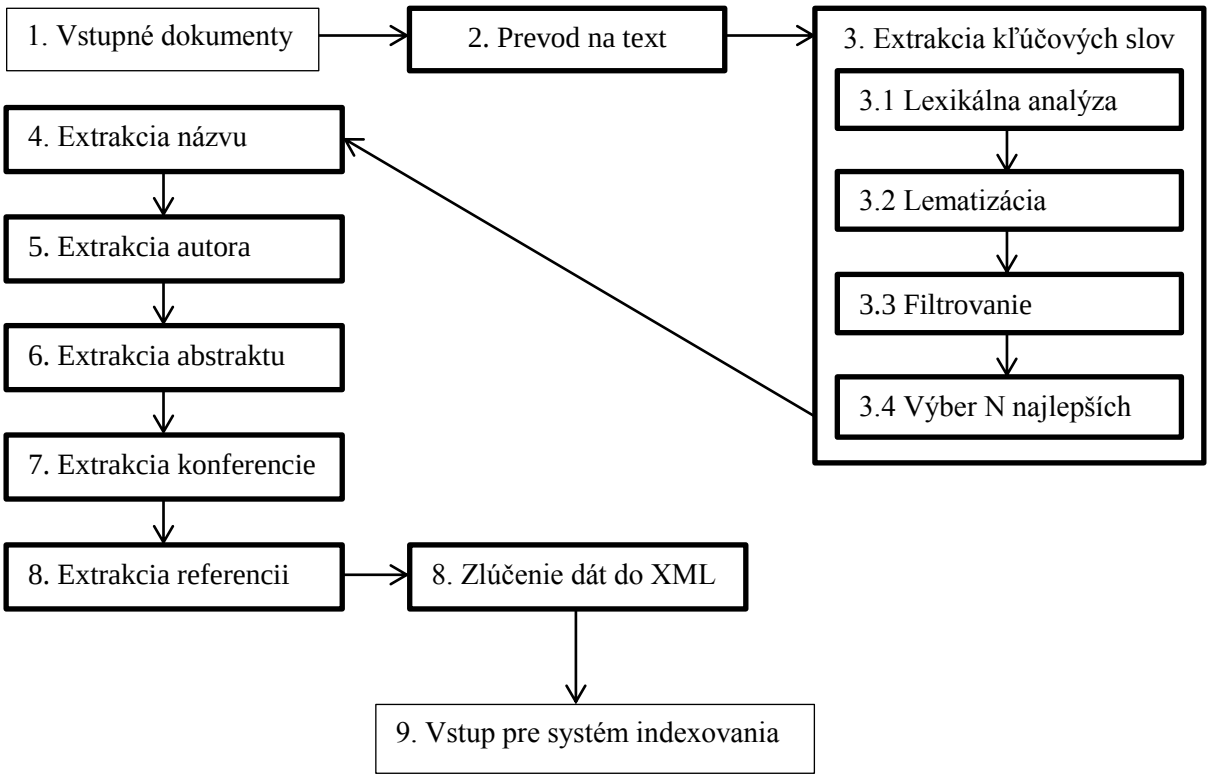
Základný princíp predspracovania spočíva v konverzii vstupných dokumentov do jednotného digitálneho formátu. Typicky sú prevedené na textové dokumenty, ktoré obsahujú len samotný text publikácie. Dochádza k odstráneniu rôznych štýlov formátovania, ako je napríklad písanie do stĺpcov, zvýrazňovaniu slov a tak ďalej. Takisto výsledný text neobsahuje obrázky alebo iné digitálne prílohy. Výsledkom konverzie je samostatný štruktúrovaný text publikácie, na základe ktorého sú odvodené ďalšie nutné metainformácie pre spracovanie systémom.

Ako prvý krok spracovania samotného textu je extrakcia kľúčových slov, ktorá spočíva zo štyroch základných častí. Prvou časťou je lexikálna analýza textu, za pomoci ktorej prebieha identifikácia slov a viacslovných výrazov. Za pomoci procesu tokenizácie dôjde k rozdeleniu textu na tokeny, ktoré predstavujú základnú jednotku používanú pri vyhľadávaní slov a viacslovných výrazov. Druhou časťou je prevod týchto slov a viacslovných výrazov do základného tvaru (nominatív jednotného čísla) a určenie ich slovného druhu. Prevod do základného tvaru je založený na redukcii, ktorá sa nazýva lematizácia. Nasledujúca časť sa zaoberá odstránením slov, ktoré nenesú význam a sú použité pre obohacovanie písaného textu. Jedná sa typicky o spojky, citoslovcia alebo častice. Takisto je možné využiť stop-list pre odstránenie nežiaducich slov. Poprípade je ešte možné využiť slovník, ktorý pomôže vyfiltrovať neexistujúce slová. Poslednou časťou je zoradenie takto vytvorenej množiny termínov podľa početnosti výskytov v texte daného dokumentu. Následne sa vyberie N najpoužívanejších termínov, ktoré označíme ako kľúčové slová.

Nasledujúcim krokom spracovania textu dokumentu je získanie názvu článku, autora, názov konferencie, na ktorej bol publikovaný, poprípade abstraktu. Typicky sú tieto informácie uvádzané na začiatku dokumentu. Úspešnosť tejto extrakcie je nižšia preto je vhodná manuálna kontrola získaných údajov. V našom systéme sa predpokladá, že každý dokument má tieto informácie obsiahnuté v súbore, ktorý bol vygenerovaný na základe dostupných informácii pri sťahovaní daného dokumentu. O generovanie týchto informácii sa stará webový extraktor, ktorý dokáže rozpoznávať a extrahovať požadované entity z webových stránok.

Posledný krok, ktorý sa zaoberá spracovaním textu dokumentu je extrakcia literatúry. Typicky je tento zoznam uvedený na konci dokumentu. Pre identifikáciu sú použité regulárne výrazy, ktoré vyhľadávajú typický formát zápisu referencii (1., 1), [1], I., ...). Úspešnosť tejto metódy je typicky dostačujúca, ale pre zaručenie správnych výsledkov je vhodná manuálna kontrola, najmä u dokumentov s nadmerným počtom referencii.

Na záver je nutné všetky získané informácie spojiť do výsledného formátu, ktorý sa použije ako vstup pre systém indexovania. Pre zápis sa používa formát XML [4], ktorý je vhodný najmä pre jednoduchú vizualizáciu uchovávaných dát. Čo je vhodné najmä ak je nutná manuálna kontrola výstupných dát. Formát obsahuje požiadavku na pridanie nového dokumentu do indexu na základe všetkých získaných informácii. Obsiahnuté informácie korešpondujú so schémou systému pre indexovanie, ktorá je popísaná v nasledujúcej kapitole. Obrázok 4.1 zobrazuje návrh systému.



Obrázok 4.1: Návrh systému pre predspracovanie dokumentov

4.2 Systém pre indexovanie

Návrh systému pre indexovanie je veľmi dôležitý, nakoľko v celom systéme zohráva najdôležitejšiu rolu. Funguje na princípe databázy, v ktorej sú uskladnené všetky informácie o dostupných vedeckých dokumentoch. Samozrejmosťou je pridávanie, editácia, poprípade mazanie za behu, bez toho aby bolo nutné reštartovať alebo nejakým iným spôsobom obnovovať index. Takisto je dôležitá modulárnosť systému tak, aby bolo možné vytvárať jednotlivé metódy pre vyhľadávanie sémanticky podobných dokumentov ako zásuvné moduly, ktoré implementujú poskytované rozhranie.

Vstupom systému pre indexovanie sú XML súbory, ktoré obsahujú informácie o danom vedeckom dokumente. Okrem týchto dát je definované či sa jedná o nový záznam, editáciu alebo mazanie existujúceho záznamu. Štruktúra obsiahnutých dát reflektuje schému systému. Schéma predstavuje formát dát, ktoré sú spracúvané indexom. Analogicky s databázou si môžeme predstaviť, že obsahuje informácie o stĺpcoch tabuľky. Štruktúra schémy systému pre indexovanie je reprezentovaná tabuľkou 4.1.

Pole	**Popis**
id	Identifikátor dokumentu (názov, poprípade absolútna cesta).
title	Názov článku.
conference	Konferencia, na ktorej bol daný článok prezentovaný.
text	Dostupný text dokumentu.
terms	Zoznam N najlepších termínov, ktoré považujeme za kľúčové slová.
references	Zoznam referencii daného dokumentu.

Tabuľka 4.1: Štruktúra schémy systému pre indexovanie

Pre dosiahnutie modulárnosti a možnosti použitia zásuvných modulov je vytvorené rozhranie *Module Prototype*, ktoré po implementácii poskytuje sadu operácii pre prístup k indexu a pre samotné získavanie dát. Zmeny modulov sú možné len pri vypnutom systéme, aby nedochádzalo ku kolíziám. Každý modul predstavuje samostatný celok, ale je mu umožnené využívať ďalšie moduly, poprípade spúšťať aplikácie na zdrojovom počítači. Tu je nutné kontrolovať či moduly, ktoré vyžadujú pre prácu, sú súčasťou systému pre indexovanie a takisto je nutné obmedziť práva prístupu na lokálnom počítači z bezpečnostných dôvodov.

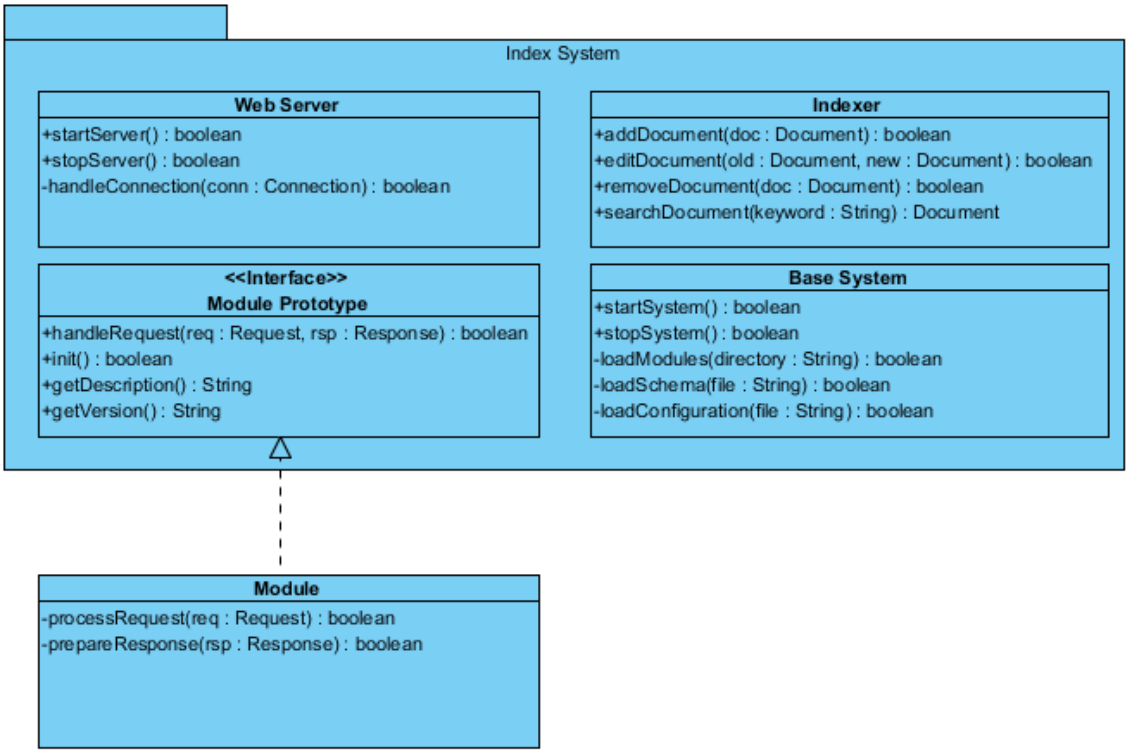
Posledným bodom návrhu systému pre indexovanie je užívateľské rozhranie, ktoré umožňuje spustenie vyhľadávania sémanticky podobných článkov, poprípade umožňuje vyhľadávanie, editovanie alebo mazanie dát v indexe. Systém funguje ako proces na pozadí a prostredníctvom webového rozhrania sprístupňuje požadované možnosti. Tým vzniká problém s viacnásobným prístupom, ktorý je nutné vyriešiť počas implementácie. Užívateľské rozhranie je rozdelené do dvoch častí.

Prvou časťou je administrácia, ktorá umožňuje spravovať nastavenie systému pre indexovanie a takisto sledovať stav indexu. Je možné zobraziť aktuálnu schému indexu a konfiguráciu systému, pre ich editáciu je nutné reštartovanie systému. Takisto je umožnené nahrávanie, editácia a mazanie dokumentov z indexu. Posledným dôležitým prvkom je jednoduchý vyhľadávač, ktorý umožňuje prezeranie samotného indexu.

Druhou časťou užívateľského rozhrania sú samostatné moduly, ktoré implementujú požadované rozhranie. Zobrazované dáta a informácie sú kompletne pod kontrolou modulu. Záleží na konkrétnej implementácii, čo a akým spôsobom je zobrazované.

Výstup systému pre indexovanie je reprezentovaný z užívateľského rozhrania v podobe webových stránok, ktoré informujú užívateľa o stave indexu alebo vyhľadávania. Formát výstupu vyhľadávania sémanticky podobných článkov je v roli jednotlivých modulov, ktoré majú možnosť vypísať výsledok do webovej stránky alebo vytvoriť výstupný súbor na lokálnom disku. Tu opäť môže dôjsť ku kolízii v prípade viacnásobného prístupu a táto skutočnosť bude musieť byť ošetrená vo fáze implementácie.

Obrázok 4.2 predstavuje vizualizáciu popísaného návrhu systému pre indexovanie. Tento diagram tried popisuje systém na abstraktnej úrovni a poukazuje na vysvetlenia vzťahu zásuvného modulu k celému systému. Samotná implementácia bude rádovo zložitejšia.



Obrázok 4.2: Vizualizácia návrhu systému pre indexovanie

4.3 Abstraktný modul

Pre čistý návrh jednotlivých modulov využijeme abstraktný modul, ktorý obsahuje implementáciu operácii, ktoré súvisia so systémom pre indexovanie a ďalšie statické metódy, ktoré sú spoločné pre všetky moduly.

Základom abstraktného modulu je implementácia metód rozhrania systému pre indexovanie. Jedná sa najmä o inicializáciu modulu, ktorá spočíva v inicializácii jednotlivých častí systému ako je samotný indexer a v prípade potreby aj ďalších komponent, ktoré sú modulom vyžadované. Ďalšou dôležitou súčasťou implementácie je predspracovanie požiadavky na vyhľadávanie sémanticky podobných vedeckých článkov, ktoré pozostáva z vyhľadania požadovaných dokumentov v indexe. Následne tieto odkazy do indexu sú spracované tak, že získame potrebné informácie pre vyhľadávanie. Jedná sa najmä o zoznam všetkých termínov a početností výskytov pre jednotlivé dokumenty. Pre tieto účely je vytvorená štruktúra *DocumentData*, ktorá reprezentuje jeden dokument so všetkými dostupnými informáciami a ktorá je súčasťou abstraktného modulu.

Medzi ďalšie metódy, ktoré sú implementované z rozhrania systému pre indexovaniu patrí získavanie popisu modulu, verzie atď. Tieto metódy obsahujú v abstraktnom module základné informácie, ktoré sú doplnené podrobnejšími na základe implementácie jednotlivých modulov.

Najdôležitejšou je abstraktná metóda *handleRequest*, ktorá reprezentuje metódu z rozhrania systému pre indexovanie. Jedná sa o spracovanie samotnej požiadavky na vyhľadanie sémanticky podobných dokumentov. Túto metódu implementujú jednotlivé moduly, ktoré sú potomkami abstraktného modulu. Základom implementácie je zavolanie metódy *startRequest* pre predspracovanie požiadavky, ktorá je implementovaná v abstraktnom module, čím sa pripraví odpoveď, ktorá je generovaná ako výstup. Odpoveď obsahuje informácie o požiadavke, jej parametre a čas potrebný pre spracovanie požiadavky. Jednotlivé moduly môžu do tejto odpovede pripisovať ďalšie informácie ako dodatočné parametre, poprípade oznámiť úspešnosť vykonania požiadavky.

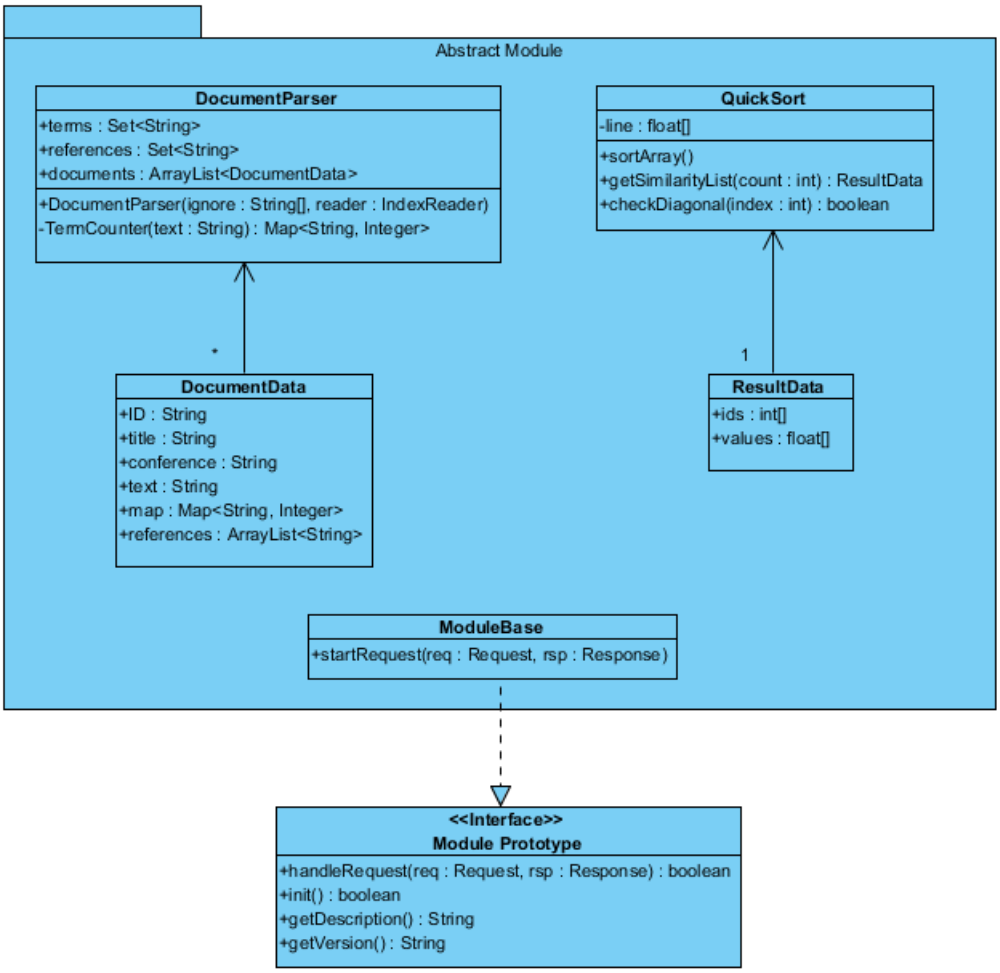
Ako už bolo spomínané abstraktný modul obsahuje štruktúru *DocumentData*, ktorá reprezentuje jeden dokument načítaný z indexu so všetkými informáciami, ktoré bolo možné o ňom získať. Táto štruktúra korešponduje so schémou indexu, čiže v prípade zmeny schémy ju bude nutné upraviť. Vo fáze predspracovania požiadavky sa vytvorí zoznam takýchto štruktúr, ktorý je sprístupnený modulu. O tvorbu tohto zoznamu sa stará ďalšia súčasť abstraktného modulu, a to trieda *DocumentParser*. Načítava jednotlivé dokumenty z indexu, získava všetky potrebné informácie a následne vytvorí lineárny zoznam všetkých dokumentov obsiahnutých v indexe. Okrem tohto zoznamu obsahuje aj informácie o všetkých termínoch a referenciách v indexe. Tým sa všetky potrebné dáta presunú do pamäte RAM, čím vylúčime opakované získavanie informácii z indexu a pracujeme len s referenciami.

Ďalšou súčasťou abstraktného modulu je implementácia radiaceho algoritmu quicksort, ktorá je reprezentovaná triedou *QuickSort*. Všetky moduly, ktoré sú implementované, potrebujú radiť

výsledky, ktoré sú reprezentované jedným riadkom matice podobnosti. Aby sa predišlo opakovanej implementácii v každom module, abstraktný modul obsahuje metódy radenia, ktoré môžu využívať všetky moduly. Jedná sa o klasický algoritmus quicksortu, ktorý pracuje s desatinnými číslami a okrem iného obsahuje aj metódu *checkDiagonal* pre určenie, či dokument na danom riadku matice podobnosti je najpodobnejší sám so sebou. Táto metóda je dôležitá pre testovanie správnosti výsledkov, ktoré daný algoritmus generuje. Skúmaný dokument musí mať sám so sebou podobnosť rovnú 1, čím poukazuje na fakt, že sa jedná o identický dokument.

Poslednou súčasťou abstraktného modulu je štruktúra *ResultData*, ktorá reprezentuje výsledok radenia quicksortom. Je potrebná z toho dôvodu, že pri radení postupnosti riadku matice podobnosti by došlo k strate informácie o tom, ku ktorému dokumentu daná hodnota podobnosti patrí. Táto štruktúra obsahuje pole indexov, ktoré zachová túto informáciu.

Obrázok 4.3 predstavuje diagram tried abstraktného modulu. Zobrazuje vzťahy medzi jednotlivými triedami a takisto zobrazuje spôsob prepojenia so systémom pre indexovanie pomocou implementácie rozhrania.



Obrázok 4.3: Diagram tried návrhu abstraktného modulu

4.4 Modul prienik termínov

Predstavuje modul vyhľadávania sémanticky podobných článkov založený na prieniku množín kľúčových slov. Jedná sa o najjednoduchší modul, ktorého úlohou je načítať dokumenty z indexu, vytvoriť množiny kľúčových slov a na základe ich prieniku vyhodnotiť ich podobnosť.

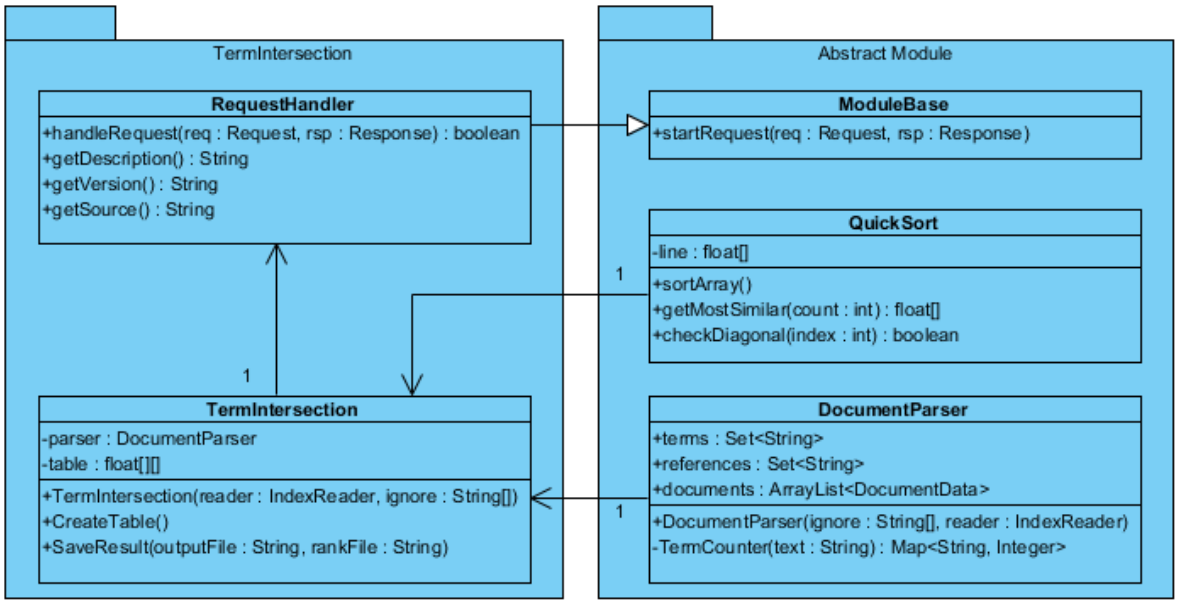
Základná trieda, ktorá sa stará o spracovanie požiadavky dedí abstraktný modul, a tým získava sadu operácii pre prácu s indexom a jednotlivými dokumentmi. Dôležitým aspektom je implementácia abstraktnej metódy *handleRequest* pre spracovanie požiadavky, v rámci ktorej dochádza k vytvoreniu objektu triedy *TermIntersection.* Následne je vyvolaná metóda *CreateTable*, v rámci ktorej prebieha celý výpočet podobnosti a nakoniec za pomoci metódy *SaveResult* dochádza k zápisu výsledku.

Výpočet prebieha nasledovne, stanoví sa veľkosť prieniku množín kľúčových slov dvoch porovnávaných dokumentov. Následne sa táto veľkosť podelí počtom termínov prvého dokumentu, čím sa získava hodnota v rozmedzí 0 až 1, ktorá predstavuje pomer zhodných kľúčových slov daných dokumentov. Táto hodnota sa použije ako veľkosť podobnosti. Pre každý dokument sú tieto hodnoty ukladané do matice podobnosti, na základe ktorej sa určuje výsledná podobnosť jednotlivých vedeckých článkov.

Pre získanie výsledku je nutné zoradiť hodnoty matice podobnosti na jednom riadku od najvyššej hodnoty po najnižšiu, čím sa získava klesajúca postupnosť od najpodobnejšieho dokumentu po najmenej podobný. Pre toto radenie sa využíva implementácia quicksortu z abstraktného modulu. Tento riadok reprezentuje podobnosť jedného dokumentu, ktorý je odvodený na základe indexu riadku s ostatnými článkami v indexe, ktoré sú odvodené na základe indexu stĺpca. Nakoniec dochádza k odstráneniu hodnoty podobnosti samotného článku so sebou, ktorá je vždy rovná 1.

Výsledok je zapísaný do súboru na lokálnom disku v pracovnom adresári systému pre indexovanie. Názov súboru je možné špecifikovať pomocou parametru požiadavky, poprípade sa použije štandardný názov, ktorý predstavuje „INT_output.xml“. Počas implementácie bude nutné dbať na problém viacnásobného zápisu do jedného súboru, poprípade prepisovaniu existujúceho obsahu. Do výstupu pre užívateľské rozhrania sú doplnené parametre spustenia a informácia o úspešnosti vyhľadávania sémanticky podobných dokumentov.

Obrázok 4.4 zobrazuje diagram tried, ktorý popisuje návrh implementácie modulu pre vyhľadávanie sémanticky podobných dokumentov na základe prieniku termínov. Poukazuje na vzťah dedičnosti s abstraktným modulom a takisto vzťahy jednotlivých tried, ktoré obsahuje.



Obrázok 4.4: Návrh modulu prieniku termínov

4.5 Modul latentná sémantická analýza

Jedná sa o modul, ktorý sa zaoberá skúmaním kontextu jednotlivých slov a analogicky aj celých dokumentov. Latentná sémantická analýza predstavuje metódu vyhľadávania sémanticky podobných článkov, ktorá je typická pre získavanie najlepších výsledkov z dostupných metód. Nevýhodou je vysoká náročnosť na výpočetný výkon, ktorá vzniká pri použití transformácie vektorového priestoru pomocou metódy singulárneho rozkladu.

Základná trieda modulu opäť dedí vlastnosti z abstraktného modulu a implementuje abstraktnú metódu *handleRequest*, ktorá sa stará o spracovanie požiadavky. Spracovanie požiadavky je rozšírené o nový parameter, ktorý reprezentuje veľkosť dimenzie vektorov, na ktorú sa majú transformovať počas singulárneho rozkladu. V prípade, že sa daný parameter nevyužije, tak sa použije štandardná hodnota, ktorá predstavuje počet dokumentov v indexe. Následne dochádza k vytvoreniu objektu triedy *LSA*, ktorý riadi celý výpočet.

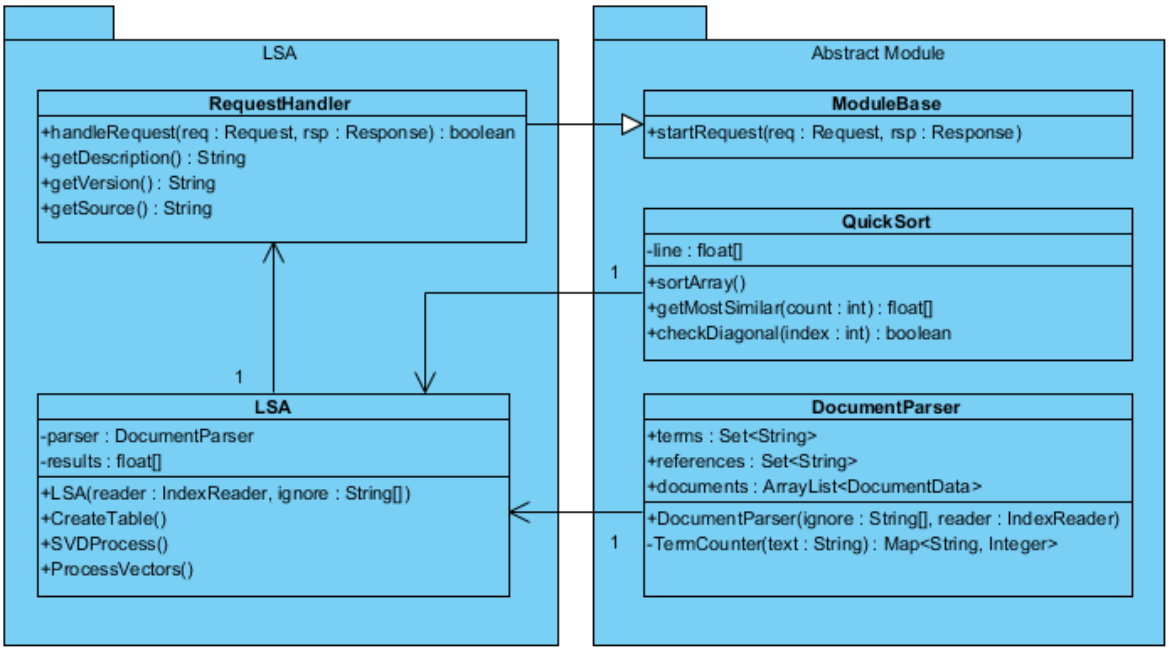
Výpočet latentnej sémantickej analýza spočíva vo vytvorení matice výskytov pomocou metódy *CreateTable*, kde riadky predstavujú dokumenty a stĺpce termíny. Hodnoty tejto matice sú odvodené na základe početnosti výskytov jednotlivých termínov v daných dokumentoch. Tieto hodnoty sú získané pomocou triedy *DocumentParser* z abstraktného modulu, ktorý využíva techniku tokenizácie na základe bežných oddeľovačov ako sú biele znaky, čiarky, bodky a tak podobne, čím získa množinu tokenov, ktoré reprezentujú termíny a ich počet výskytov. Tým vzniknú vektory pre dokumenty, ktoré sú následne transformované volaním metódy *SVDProcess* pomocou singulárneho rozkladu do troch matíc. Následným spustením metódy *ProcessVectors* dochádza k maticovému násobeniu matice V^T,

ktorá predstavuje transponované vektory dokumentov, maticou Σ sprava, ktorá predstavuje diagonálnu maticu, čím získame výsledné kontextové vektory pre dokumenty. Tieto vektory musia byť ešte normalizované a následne ich vzájomným násobením získavame hodnoty do matice podobnosti.

Po vytvorení matice podobnosti je nutné jednotlivé riadky matice zoradiť od najpodobnejšieho dokumentu po najmenej podobný, k čomu sa využije implementácia quicksortu z abstraktného modulu. Nakoniec dochádza k odstráneniu hodnoty podobnosti samotného článku so sebou, ktorá je vždy rovná 1.

Výsledok je zapísaný do súboru v pracovnom adresári systému pre indexovanie rovnako ako u prieniku termínov. Takisto je možné špecifikovať názov výstupného súboru, poprípade sa použije štandardný, ktorý predstavuje „LSA_output.xml“. Rovnako bude nutné ošetriť viacnásobný prístup, poprípade prepisovanie obsahu vo fáze implementácie. Do výstupu pre užívateľské rozhranie sú doplnené parametre spustenia a informácia o úspešnosti vyhľadávania sémanticky podobných dokumentov.

Obrázok 4.5 predstavuje diagram tried, ktorý popisuje návrh implementácie modulu pre vyhľadávanie sémanticky podobných článkov na základe latentnej sémantickej analýzy. Vyjadruje vzťah dedičnosti s abstraktným modulom a zobrazuje jednotlivé triedy a ich vzťahy.



Obrázok 4.5: Návrh modulu pre latentnú sémantickú analýzu

4.6 Modul random indexing

Predstavuje modul, ktorý sa zaoberá skúmaním vektorového priestoru za pomoci náhodných index vektorov. Random indexing na rozdiel od latentnej sémantickej analýzy nevyžaduje transformáciu vektorového priestoru, pretože počas celého výpočtu pracuje s vopred stanovenou veľkosťou dimenzie vektoru.

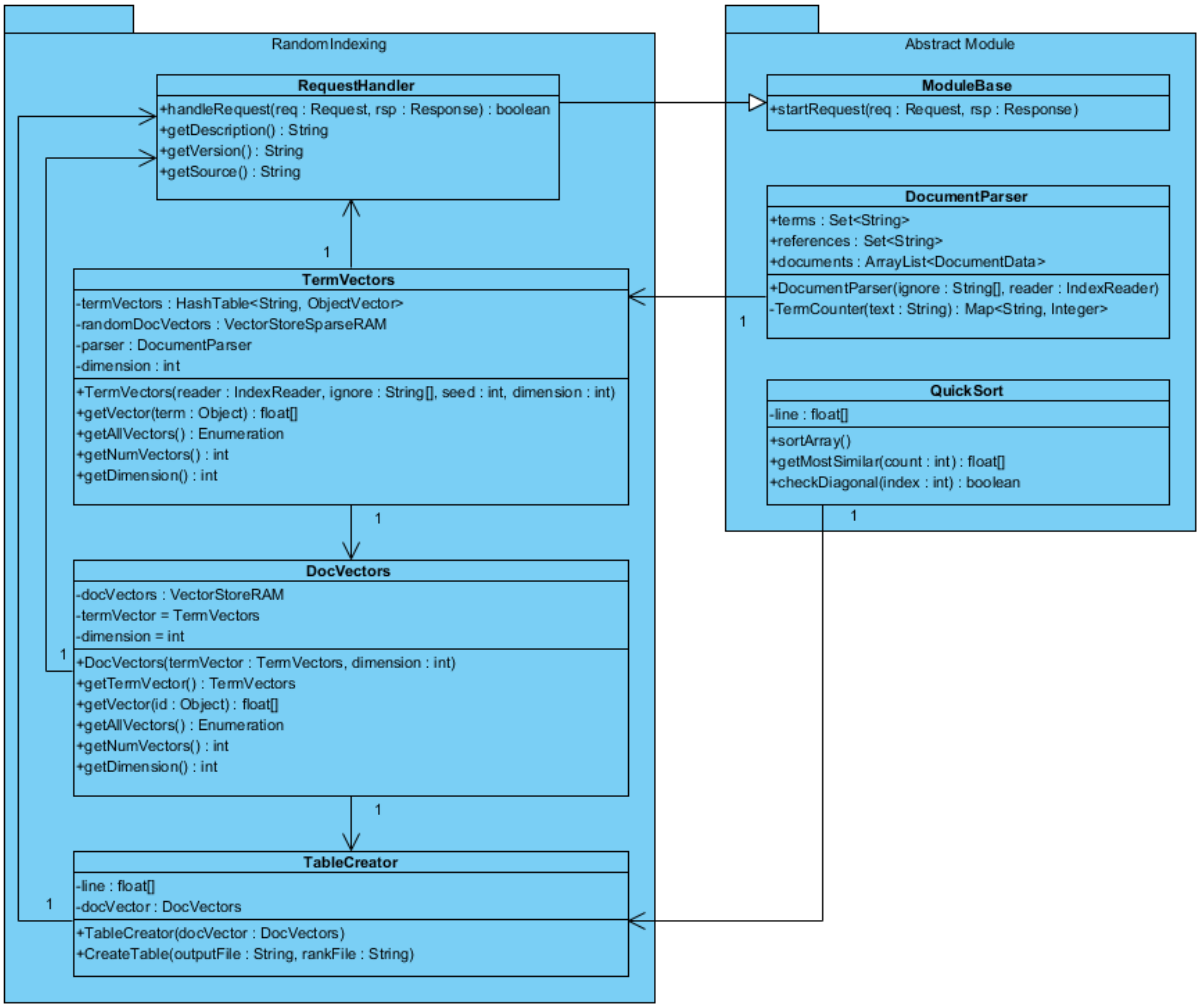
Základná trieda opäť dedí vlastnosti abstraktného modulu a implementuje abstraktnú metódu *handleRequest*, za pomoci ktorej je možné spracovať požiadavku na vyhľadanie sémanticky podobných článkov. K spracovaniu je okrem zistenia veľkosti dimenzie vektorov pridaný nový parameter, ktorý je označovaný ako *seed*. Predstavuje číslo, ktoré sa používa pre inicializáciu pseudonáhodného generátora čísel. Tento generátor sa používa počas výpočtu pre generovanie náhodných index vektorov.

Základom výpočtu je inicializácia objektu triedy *TermVectors*, ktorý má za úlohu vygenerovanie náhodných index vektorov pre jednotlivé dokumenty indexu. Generujú sa na základe pseudonáhodného generátora čísel, ktorý má rovnomerné rozloženie. Inicializuje vektor tak, že na pozíciu každej dimenzie vloží náhodne číslo -1, 0 alebo 1. Následne na základe týchto vektorov sú vytvorené kontextové vektory pre jednotlivé termíny dokumentov. Tie vzniknú súčtom výsledkov násobenia početnosti výskytu a náhodného index vektoru dokumentu, v ktorom sa nachádzajú. Posledným krokom výpočtu vektorového priestoru je vytvorenie objektu triedy *DocVectors*, ktorý má za úlohu ovplyvnenie index vektorov pre dokumenty na základe kontextov termínov, ktoré obsahuje. Opäť sa využije súčet výsledkov násobenia početnosti výskytov daného termínu a jeho kontextového vektoru.

Tým sme získali kontextové vektory pre jednotlivé dokumenty, na základe ktorých dochádza k vytvoreniu výsledku. Nesmieme zabudnúť na normalizáciu týchto vektorov aby sme získali výsledné hodnoty v rozmedzí -1 až 1. Posledným krokom výpočtu je vytvorenie objektu triedy *TableCreator*, ktorý reprezentuje maticu podobnosti. Pomocou násobenia jednotlivých kontextových vektorov dokumentov a následného zoradenia výsledných hodnôt pomocou quicksortu, dochádza k vytvoreniu výsledku. Nesmieme zabudnúť na odstránenie hodnoty podobnosti samotného článku so sebou, ktorá je vždy rovná 1.

Výsledok sa zapisuje do výstupného súboru v pracovnom adresári systému pre indexovanie. Názov súboru je možné špecifikovať pomocou parametra požiadavky, poprípade sa použije štandardný názov, ktorý predstavuje „RI_output.xml“. Opäť vo fáze implementácie bude nutné ošetriť viacnásobný prístup a prepisovanie existujúceho obsahu. Do výstupu pre užívateľské rozhranie sú doplnené parametre spustenia a úspešnosť vyhľadávania sémanticky podobných článkov.

Obrázok 4.6 predstavuje diagram tried, ktorý popisuje návrh implementácie modulu pre vyhľadávanie sémanticky podobných článkov na základe random indexingu. Zobrazuje vzťah dedičnosti s abstraktným modulom a takisto vzťahy jednotlivých tried, ktoré obsahuje.



Obrázok 4.6: Návrh modulu pre random indexing

4.7 Modul modifikovaný random indexing

Predstavuje modul, ktorý sa taktiež zaoberá skúmaním vektorového priestoru. Návrh modulu je takmer totožný s modulom pre random indexing. Spracovanie požiadavky a aj výstup je totožný, rozdiel týchto dvoch modulov je vo výpočte kontextových vektorov pre dokumenty.

Prvým krokom je takisto vygenerovanie náhodných index vektorov pre dokumenty, na základe ktorých je vytvorená množina kontextových vektorov pre termíny. Rozdiel nastáva v druhom kroku, predtým, než dôjde k ovplyvneniu vektorov dokumentov, dochádza k odstráneniu náhodnej zložky tým, že sa vytvoria nulové kontextové vektory pre dokumenty. Následne sa tieto nulové kontextové vektory ovplyvnia rovnakým spôsobom, dochádza k získaniu súčtu výsledkov násobenia početnosti výskytov termínu a jeho kontextového vektoru. Podrobné vysvetlenie sa nachádza v kapitole 4.6.

Štandardný názov pre výstupný súbor, ktorý obsahuje výsledok je odlišný a predstavuje ho „MRI_output.xml“. Umiestnenie a problémy týkajúce sa jeho vytvárania zostávajú totožné a tento fakt bude nutné vyriešiť počas implementácie.

Vizualizácia návrhu modulu pre modifikovaný random indexing je rovnaká ako na obrázku 4.6, s rozdielom názvu balíčku.

4.8 Modul navrhnutej metódy

Posledný navrhovaný modul je založený na skúmaní vektorového priestoru pomocou metódy random indexing, ktorá okrem skúmania kontextov kľúčových slov a samotných dokumentov využíva aj kontext založený na citáciách. Základom návrhu je metóda modifikovaného random indexingu, ktorá je rozšírená o možnosti ovplyvnenia kontextu dokumentov na základe dostupných znalostiach o citáciách.

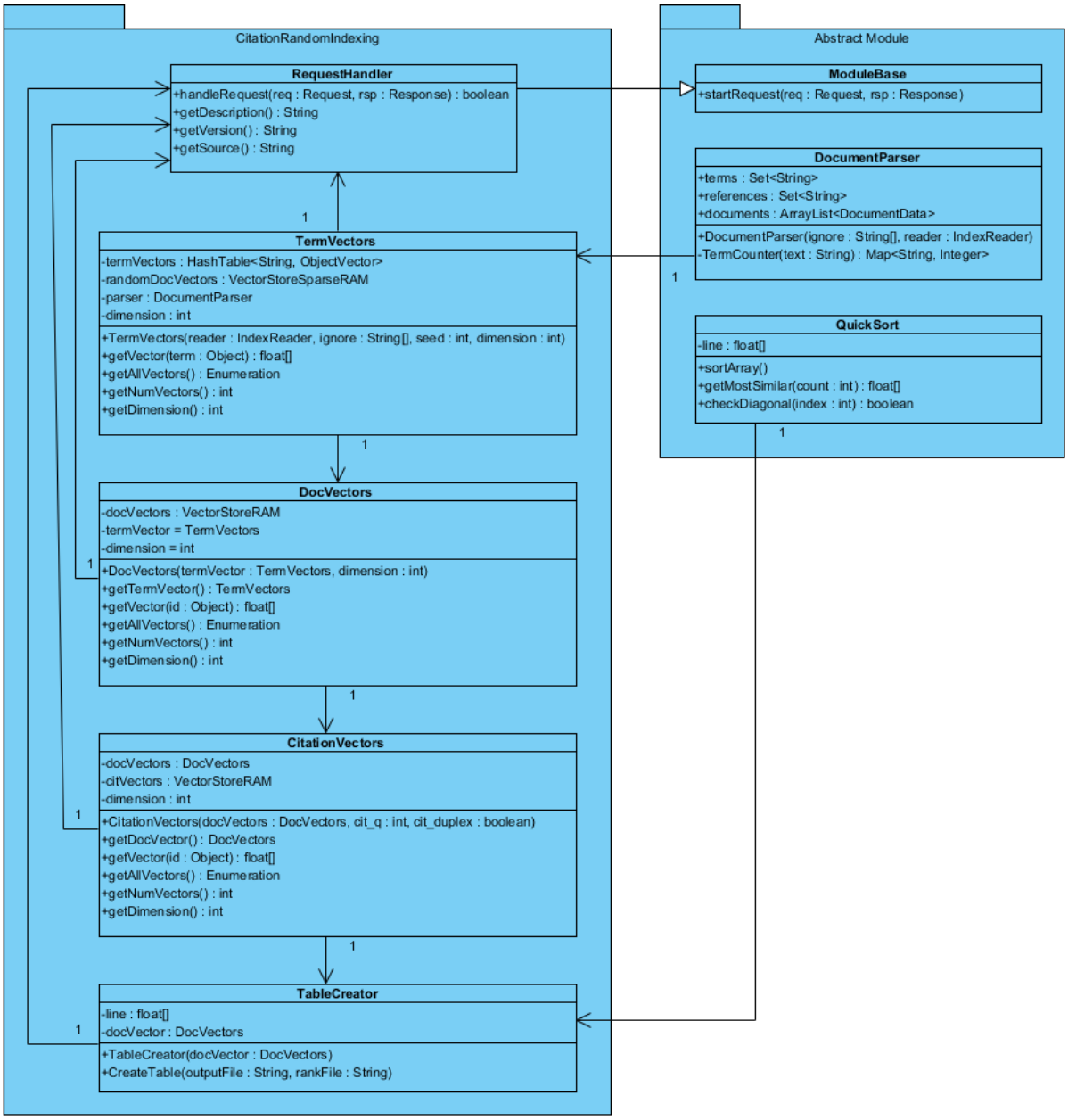
Základná trieda dedí vlastnosti abstraktného modulu a implementuje abstraktnú metódu *handleRequest*, ktorá sa stará o spracovanie požiadavky. Dochádza k rozšíreniu spracovania požiadavky o nové parametre oproti modifikovanému random indexingu, a to o koeficient q súčinu podobnosti pre citácie, ktorý určuje silu väzby na základe citácii, a príznak obojsmernej väzby *duplex* medzi dvoma citovanými dokumentmi.

Výpočet kontextových vektorov pre dokumenty je zhodný s návrhom v module modifikovaného random indexingu s tým rozdielom, že na záver sa tieto kontextové vektory nenormalizujú. Nenormalizujú sa z dôvodu nasledovného využitia týchto vektorov v ďalšom výpočte, ktorý je založený na citáciách. Ako prvé dochádza k inicializácii objektu triedy *CitationVectors* a následnému vygenerovaniu náhodných index vektorov pre referencie, ktoré nereprezentujú žiadny z dokumentov, ktorý sa nachádza v indexe. Pre referencie, ktoré sú dostupné v indexe sú pridelené kontextové vektory daných dokumentov, čím sa získavajú presnejšie výsledky. Následne dochádza k ovplyvneniu kontextu jednotlivých dokumentov na základe referencii, ktoré obsahujú. Ovplyvnenie spočíva v súčte výsledkov súčinu koeficientu podobnosti pre citácie a vektorom danej referencie. V prípade nastavenia príznaku obojsmernej väzby, dochádza k rovnakému ovplyvneniu kontextového vektoru referencie. Posledným krokom je normalizácia kontextových vektorov dokumentov, ktoré sa následne použijú pre výpočet matice podobnosti. Spôsob výpočtu matice podobnosti je zhodný s návrhom v module pre modifikovaný random indexing.

Výsledky sú zapísané do výstupného súboru, ktorý je uložený na lokálnom disku v pracovnom adresári systému pre indexovanie. Názov súboru je opäť možné špecifikovať pomocou parametru požiadavky, poprípade dochádza k použitiu prednastaveného názvu, ktorý predstavuje „CRI_output.xml“. Takisto je nutné vo fáze implementácie ošetriť možnosť viacnásobného prístupu, poprípade prepisovanie existujúceho obsahu. Do výstupu pre užívateľské rozhranie okrem

parametrov požiadavky je doplnená informácia o úspešnosti vyhľadávania sémanticky podobných dokumentov.

Obrázok 4.7 predstavuje diagram tried, ktorý popisuje návrh implementácie modulu pre vyhľadávanie sémanticky podobných článkov na základe random indexingu, ktorý je ovplyvnený citáciami. Poukazuje na vzťah dedičnosti s abstraktným modulom a takisto zobrazuje vzťahy jednotlivých tried, ktoré obsahuje.



Obrázok 4.7: Návrh modulu pre random indexing ovplyvnený citáciami

5 Implementácia systému

Táto kapitola popisuje implementáciu systému pre vyhľadávanie sémanticky podobných vedeckých článkov. Ako prvé je popísaný spôsob implementácie predspracovania dokumentov a takisto sú popísané jednotlivé nástroje, ktoré sú využité. Nasleduje výber systému pre indexovanie a jeho modifikácia pre nami potrebné účely. Záverom tejto kapitoly sú rozobrané spôsoby implementácie jednotlivých navrhnutých modulov, ktoré predstavujú zásuvné moduly zvoleného systému pre indexovanie.

5.1 Predspracovanie dokumentov

Predspracovanie dokumentov predstavuje prvý krok systému pre vyhľadávanie sémanticky podobných vedeckých článkov. Jedná sa o sekvenčný proces prevodu rôznorodých vstupných dokumentov na XML formát, ktorý je akceptovaný systémom pre indexovanie.

Pre vykonanie sekvencie príkazov sa využívajú možnosti Unixu a celé predspracovanie dokumentov je reprezentované shellovským skriptom. Tento skript sa volá *indexer.sh* a na základe vstupných parametrov vykoná požadované predspracovanie. Vstupné parametre sú reprezentované tabuľkou 5.1.

Parameter	**Popis**
-c <adresár>	Adresár obsahujúci vstupné vedecké články.
-d <adresár>	Pracovný adresár používaný pre medzi výsledky.
-n <číslo>	Číslo reprezentujúce počet extrahovaných kľúčových slov. Štandardná hodnota je 50.
-t <adresár>	Výstupný adresár pre vytvorené XML súbory.

Tabuľka 5.1: Vstupné parametre predspracovania dokumentov

Prvým krokom skriptu predspracovania dokumentov je preskúmanie vstupného adresára vedeckých článkov a konverzia rôznorodých dokumentov do jednotného textového formátu. Pre konverziu je využitý program *publication_file_text*[1]. Jedná sa o skript napísaný v programovacom jazyku Python a jeho hlavnou úlohou je určenie formátu zdrojového dokumentu na základe prípony, následná konverzia a uloženie výsledku do výstupného adresára. Konverzia spočíva v použití externých programov pre daný formát. Pre formát *pdf* je to nástroj *pdftotext*, ktorý okrem konverzie na text dokáže odstrániť rôzne štýly formátovania textu, ako napríklad písanie do stĺpcov. Ďalším typickým formátom je *postscript*, označovaný ako *ps*, na ktorého konverziu sa využíva nástroj *ps2pdf*, ktorý ho prevedie do formátu *pdf* a následne pomocou *pdftotext* do textovej podoby. Pre konverziu

[1] Autorom je Tomáš Lokaj.

formátu *rtf* sa používa externý program s názvom *unrtf*, ktorý okrem iného dokáže odstrániť zo zdrojového dokumentu obrázky, ktoré sú pre nás nepodstatné. Posledným podporovaným formátom je *doc*, na konverziu ktorého sa používa *wvText*, ktorý je súčasťou knižnice *wvWare*. Súbory s príponou *txt* sú už v správnom formáte a tak dôjde len k prekopírovaniu do výstupnej zložky. Všetky skonvertované dokumenty sa uložia s rovnakým názvom, ale zmenenou príponou na *txt* do pracovného adresára do zložky *txt*.

Druhým krokom procesu predspracovania dokumentov je extrakcia kľúčových slov, ktorá má za úlohu získať N najpoužívanejších termínov. Ako prvý použijeme *TreeTagger*[2], ktorý predstavuje nástroj pre určovanie slovných druhov a základných tvarov slov. Prevádza tokenizáciu za pomoci, ktorej dokáže v texte určiť jednotlivé slová, ktoré predstavujú tokeny. Tento nástroj pre každé slovo, ktoré sa nachádza v dokumente vygeneruje trojicu, ktorá sa skladá z pôvodného tvaru slova, slovného druhu a základného tvaru. V prípade, že základný tvar sa nepodarilo určiť, tak namiesto tretieho prvku bude vygenerovaný výraz „<unknown>". Výstupy pre jednotlivé dokumenty sú uložené do zložky *tagged* v pracovnom adresári, ktoré sú použité pre extrakciu kľúčových slov.

Samotná extrakcia kľúčových slov je založená na nástroji implementovanom v Jave s názvom *KeywordsGenerator*[3]. Umožňuje nám na základe výstupu z *TreeTaggeru* určiť N najlepších termínov, ktoré označíme ako kľúčové slová. Základom je sekvenčné prechádzanie dostupných slov z dokumentu, vyhľadávanie podstatných mien, poprípade viac slovných fráz, ktoré tvoria prídavné mená a podstatné mená. Následne dochádza k vytvoreniu lineárneho zoznamu slov alebo viacslovných fráz, ktorý je zoradený podľa počtu výskytov zostupne a z neho sa vyberie prvých N termínov, ktoré považujeme za kľúčové slová. Výstupom tohto programu je súbor, ktorý je umiestnený v pracovnom adresári. Tento dokument s názvom *documents.txt* má formát v nasledovnej podobe. Zoznam termínov pre jeden dokument začína tagom <DOC name="nazov_dokumentu.txt"> a je ukončený tagom </DOC>. Jednotlivé slová alebo viacslovné výrazy, ktoré reprezentujú kľúčové slová sú vymenované na samostatných riadkoch medzi týmito tagmi.

Tretím krokom procesu predspracovania dokumentov je extrakcia zoznamu referencii. Pre tieto účely sa využije nástroj s názvom *publication_text_data*[4]. Tento extraktor je implementovaný v programovacom jazyku Python. Na základe dostupného textu pomocou regulárnych výrazov okrem iného dokáže získať zoznam literatúry. Vstupným parametrom je adresár, ktorý obsahuje texty vedeckých článkov v jednotnom textovom formáte a výstupom je zložka, ktorá obsahuje XML súbory s vyextrahovanými informáciami. Výsledky tohto nástroju niekedy nezodpovedajú očakávaniam a je vhodná manuálna kontrola dokumentov, ktoré obsahujú nadmerný počet referencii.

Posledným a najdôležitejším krokom procesu predspracovania dokumentov je zlúčenie všetkých získaných informácii do jedného výsledného XML súboru, ktorý je prijímaný systémom pre

[2] Bol vyvinutý Helmutom Schmidom v Ústave počítačovej lingvistiky na Univerzite Stuttgard. Úspešne bol testovaný na mnohých jazykoch ako napríklad angličtina, nemčina alebo francúzština [20].
[3] Autorom je Tomáš Strachota.
[4] Autorom je Tomáš Lokaj.

indexovanie. Jednotlivé výstupné súbory sú generované do zložky, ktorá bola špecifikovaná ako výstupný adresár procesu. Základom je sekvenčné prechádzanie dokumentu *documents.txt*, ktorý obsahuje jednotlivé názvy dokumentov a ich vyextrahované kľúčové slová. Na základe názvu dokumentu je možné dohľadať ďalšie informácie, ktoré sa v priebehu procesu získali. Ako prvé je získaný názov článku, autora a konferencie, na ktorej bol publikovaný zo súboru, ktorý sa nachádza v zdrojovej zložke vstupných dokumentov a bol vygenerovaný pri sťahovaní pomocou webového extraktoru. Pokračuje načítaním dostupného textu článku, kľúčových slov a ako posledný získava zoznam dostupných referencii. V tomto bode sú získané všetky potrebné informácie o danom článku a môže dôjsť k vytvoreniu výstupného XML súboru, ktorého príklad je reprezentovaný na obrázku 5.1. Formát tohto súboru je založený na očakávanom formáte vstupu zvoleného systému pre indexovanie. Tag <add> reprezentuje pridanie nového dokumentu do systému pre indexovanie, tag <doc> začína informácie o dokumente. Pole *id* označuje názov súboru dokumentu, *title* názov článku, *conference* názov konferencie, *text* predstavuje dostupný text, *terms* jednotlivé kľúčové slová oddelené čiarkou a *references* predstavujú jednotlivé referencie.

```
<?xml version="1.0" encoding="UTF-8"?>
<add>
  <doc>
    <field name="id">/Documents/Peer-to-Peer_Computing/4306288.pdf</field>
    <field name="title">Handling Network Partitions and Mergers in Structured Overlay Networks</field>
    <field name="conference">Peer-to-Peer Computing</field>
    <field name="text">red overlay networks form a major class of peerto-peer systems, ...</field>
    <field name="terms">be,clutter,node,internet technologies,algorithm,augment,ring,time, ...</field>
    <field name="references">[1] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting ...</field>
    <field name="references">[2] Ken Birman. Gossip Algorithms and Emergent Shape. Invited talk at ...</field>
  </doc>
</add>
```

Obrázok 5.1: Príklad výstupného XML súboru

5.2 Systém pre indexovanie

Systém pre indexovanie zohráva najdôležitejšiu rolu, nakoľko reprezentuje centrálny systém, ktorého úlohou je nielen indexovanie a správa indexu, ale aj vytvorenie jednotného rozhrania pre moduly vyhľadávania sémantickej podobnosti a musí disponovať aj užívateľským rozhraním pre správu. Vlastná implementácia takéhoto komplexného systému by bola nad rozsah problematiky tejto práce a z toho dôvodu použijeme nejaké existujúce riešenie.

Ako nástroj pre indexovanie a správu indexu som sa rozhodol použiť knižnicu s názvom *Apache Lucene*[5]. Jedná sa o knižnicu, ktorá je dostupná zadarmo, spolu aj so zdrojovým kódom pre prípadné úpravy. Je podporovaná v operačnom systéme Linux a samozrejmosťou sú operácie, ktoré

[5] http://lucene.apache.org/

sú očakávané od nástroja tohto typu, ako pridávanie, editovanie a mazanie dokumentov v indexe za behu.

Apache Lucene je vyvíjaný firmou Apache Software Foundation a jedná sa o knižnicu, ktorá umožňuje tvorbu a správu indexu. Originálne bol vytvorený Dougom Cuttingom v programovacom jazyku Java. Je napísaný pod licenciou Apache Software, na základe ktorej sa jedná o voľne dostupnú knižnicu so zdrojovými kódmi. Medzi funkcie, ktoré sú zaujímavé z pohľadu nástroja pre indexovanie je možné spomenúť indexovanie celého textu, kompresia, rozdelenie jednotlivých záznamov do polí, vyhľadávanie na základe rôznych parametrov [1].

Následne na základe zvolenej knižnice pre indexovanie som zvolil systém pre indexovanie s názvom *Apache Solr*[6]. Predstavuje nadstavbu, ktorá využíva knižnicu *Apache Lucene* a spĺňa ďalšie kritéria vyžadované systémom pre indexovanie. Medzi tieto kritéria patrí možnosť tvorby zásuvných modulov na základe implementácie poskytovaného rozhrania, užívateľské rozhranie v podobe webových stránok, ktoré poskytuje správu systému, indexu a možnosť interakcie užívateľa s jednotlivými modulmi, ktoré budú slúžiť pre vyhľadávanie sémanticky podobných článkov. Takisto je tento systém podporovaný v operačnom systéme Linux a je možné ho používať ako proces na pozadí. Ako posledné opäť platí, že sa jedná o nástroj, ktorý je dostupný zadarmo, spolu aj so zdrojovým kódom pre prípadné úpravy alebo analýzu chovania systému.

Apache Solr predstavuje platformu, ktorá spĺňa všetky naše požiadavky. Je vyvíjaný firmou Apache Software Foundation a je distribuovaný zadarmo vrátane zdrojových kódov pod licenciou Apache Software. Tento systém je implementovaný v programovacom jazyku Java a obsahuje rôzne komponenty v podobe knižníc [2]. Jednou z nich je knižnica *jetty* predstavujúca webový server [7], ktorý je možné využiť pre správu systému. Obsahuje takisto už spomínanú komponentu pre indexovanie *Apache Lucene*. V neposlednej rade disponuje rozhraním pre zásuvné moduly.

Pred samotným použitím systému *Apache Solr* je nutná jeho konfigurácia. Postup konfigurácie je možné nájsť v užívateľskej príručke, ktorá sa nachádza v prílohách. Prípadne na priloženom optickom médiu sa nachádza predkonfigurovaná verzia systému pre indexovanie, ktorá obsahuje požadovanú schému a všetky zásuvné moduly, ktoré boli vypracované v tejto práci.

Posledným bodom implementácie systému pre indexovanie je shellovský skript s názvom *post.sh*, ktorého úlohou je odoslanie výstupných XML súborov z predspracovania dokumentov na vstup systému pre indexovanie. Tento skript má len jeden parameter reprezentujúci zložku, ktorá obsahuje požadované XML súbory. Predpokladá, že systém pre indexovanie beží na lokálnom počítači a štandardnom porte, v prípade že tomu tak nie je, bude nutná jeho modifikácia. Informácie o tejto modifikácii sa nachádzajú v užívateľskej príručke.

[6] http://lucene.apache.org/solr/

5.3 Abstraktný modul

Na základe zvoleného systému pre indexovanie je zrejmé, že jednotlivé moduly sú implementované v programovacom jazyku Java. Distribúcia spočíva v archíve jar, ktorý vzniká ako výstup kompilačného procesu a následnej kompresii. Abstraktný modul a jeho ďalšie komponenty sú reprezentované balíčkom *related_docs*.

Hlavnou triedou balíčku je *ModuleBase*, ktorá má za úlohu implementovať poskytované rozhranie systémom pre indexovanie. Jedná sa o implementáciu rozhrania *SolrCoreAware* a dedenie triedy *RequestHandlerBase*. Čím vzniká nutnosť implementácie metódy *inform*, ktorá informuje centrálny systém o novom module a takisto sa stará o inicializáciu komponent, ktoré sú vyžadované týmto modulom. Táto trieda ďalej obsahuje definíciu statických dát, ktoré sú využívané ako konštanty ďalšími modulmi. Medzi ne patrí špecifikácia prednastavených hodnôt parametrov požiadavky, ako sú napríklad názvy výstupných súborov alebo zoznam ignorovaných kľúčových slov. Ďalšie metódy, ktoré preťažujú definíciu z rodičovskej triedy sú *getDescription, getSourceID, getSource* a *getVersion*. Jedná sa o definovanie základných informácii o danom module, ktoré je možné vypísať v užívateľskom rozhraní. Poslednou implementovanou metódou je *startRequestBody*, ktorá predstavuje inicializáciu odpovede požiadavky na vykonanie vyhľadania sémanticky podobných článkov. Okrem implementovaných metód trieda obsahuje aj abstraktnú metódu *handleRequestBody*, ktorá predstavuje spracovanie samotnej požiadavky. K jej implementácii dochádza v jednotlivých moduloch pre sémantické vyhľadávanie podobnosti.

Ďalšou triedou balíčku *related_docs* je *DocumentParser*, ktorá predstavuje rozhranie pre prácu s indexom využívané ďalšími modulmi. Jej hlavnou úlohou je vyčítať všetky dokumenty z indexu a na základe získaných informácii inicializovať objekty *DocumentData*. Táto trieda po inicializácii za pomoci konštruktora, ktorý vyžaduje prístup do indexu v podobe odkazu na objekt triedy *IndexReader* a zoznamu ignorovaných kľúčových slov vytvorí dve množiny a jeden lineárny zoznam. Prvou množinou je *terms*, ktorá predstavuje zoznam všetkých termínov v indexe. Druhú množinu predstavuje *references*, ktorá obsahuje všetky referencie v indexe. Lineárny zoznam s názvom *documents* obsahuje objekty triedy *DocumentData* pre každý dokument obsiahnutý v indexe.

Poslednou triedou balíčku *related_docs* je *QuickSort*, ktorá predstavuje implementáciu tradičného radiaceho algoritmu quicksort pre prácu s desatinnými číslami. Po inicializácii konštruktorom, ktorý vyžaduje pole desatinných čísel určených pre radenie dôjde k premiešaniu tohto pola pomocou metódy *Shuffle*, aby nedošlo k radeniu opačnej postupnosti. Následne je vykonané samotné radenie za pomoci metódy *Sort*. Okrem radenia táto trieda obsahuje metódu *checkDiagonal* pre kontrolu diagonály matice podobnosti, ktorej hodnoty na jednom riadku radíme. Vyžaduje parameter s indexom riadku a hodnota tohto indexu sa musí rovnať 1. Poslednou metódou je *GetSimilarityList*, ktorá reprezentuje získanie výsledku radenia v podobe objektu triedy *ResultData*.

5.4 Modul prienik termínov

Modul prienik termínov je reprezentovaný vlastným balíčkom *term_intersection*, ktorý patrí v stromovej štruktúre pod balíček *related_docs*. Základom je dedenie abstraktnej triedy *ModuleBase* a následné využívanie ďalších tried abstraktného modulu.

Hlavnou triedou balíčku je *TermIntersectionRequestHandler*, ktorá predstavuje potomka triedy *ModuleBase* a stará sa o spracovanie požiadavky na vyhľadanie sémanticky podobných článkov pomocou prieniku termínov. Základom je implementácia abstraktnej metódy *handleRequestBody*, ktorá využije metódu *startRequestBody* z abstraktného modulu pre inicializáciu odpovede na požiadavku. Ako prvé dôjde k prečítaniu parametrov požiadavky, poprípade k použitiu preddefinovaných hodnôt ak neboli špecifikované. Následne sa vytvorí objekt triedy *TermIntersection*, ktorý má na starosť samotný výpočet. Táto trieda okrem iného obsahuje aj preťaženie metód abstraktnej triedy súvisiacich s poskytovaním základných informácii o module.

Balíček *term_intersection* obsahuje ešte spomínanú triedu *TermIntersection*, ktorá obsahuje výpočet sémantickej podobnosti. Za pomoci konštruktora dôjde k inicializácii objektu triedy *DocumentParser* z abstraktného modulu, čím získame informácie o dostupných dokumentoch v indexe. Metóda *CreateTable* má na starosť tvorbu matice podobnosti, ktorá predstavuje štvorcovú maticu odvodenú z počtu skúmaných článkov. Následným prechádzaním po riadkoch a stĺpcoch dôjde pomocou metódy *CompareDocuments* k výpočtu hodnoty podobnosti na základe veľkosti prieniku množín kľúčových slov. Ďalšou metódou je *SaveResult*, ktorá predstavuje vytvorenie a zápis výsledku do súboru. Postupným prechádzaním matice podobnosti po riadkoch a radením týchto hodnôt pomocou objektu triedy *QuickSort*, dôjde k získaniu výsledku v podobe objektu triedy *ResultData*. Tento výsledok sa zapíše do výstupného súboru, ktorého výstup je demonštrovaný na obrázku 5.2. Tag <result> začína výsledok, <file> označuje výsledok skúmaného dokumentu uvedeného v parametre *name* a jednotlivé tagy <hit> predstavujú podobné dokumenty s hodnotou podobnosti.

```
<result>
  <file name="/Documents/IEEE_International_Symposium/4104539.pdf"/>
    <hit name="/Documents/Parallel,_Distributed_and_Network-Based_Processing/2467053.pdf">0.5419964</hit>
    <hit name="/Documents/Peer-to-Peer_Computing/6604948.pdf">0.5306574</hit>
    <hit name="/Documents/IEEE_International_Symposium/543529.pdf">0.51780915</hit>
    <hit name="/Documents/Peer-to-Peer_Computing/1775296.pdf">0.51096004</hit>
    <hit name="/Documents/Networked_Systems_Design_and_Implementation/4305317.pdf">0.46208912</hit>
    <hit name="/Documents/Symposium_on_Reliability_in_Distributed_Software/4319779.pdf">0.42848977</hit>
    <hit name="/Documents/Supercomputing/4313178.pdf">0.42776796</hit>
  </file
</result>
```

Obrázok 5.2: Demoštrácia výstupného súboru

5.5 Modul latentná sémantická analýza

Modul latentnej sémantickej analýzy je reprezentovaný balíčkom *latent_semantic_analysis*, ktorý taktiež v stromovej štruktúre patrí pod balíček *related_docs*. Základom tohto je rovnaký princíp ako u modulu prieniku termínov, čiže dôjde k dedeniu abstraktnej triedy *ModuleBase* a následnému použitiu ďalších tried z abstraktného modulu.

Hlavnou triedou balíčku je *LatentSemanticAnalysisRequestHandler*, ktorá dedí vlastnosti z triedy *ModuleBase*. Jej úlohou je obsluha prichádzajúcej požiadavky pre vyhľadanie sémanticky podobných článkov pomocou latentnej sémantickej analýzy. Základom je opäť implementácia abstraktnej metódy *handleRequestBody*, ktorá na základe inicializácie odpovede pomocou metódy *startRequestBody* z abstraktného modulu spracuje požiadavku. Ako prvé dôjde k načítaniu parametrov požiadavky, prípadne použitiu preddefinovaných hodnôt ak neboli definované. Spracovanie pokračuje vytvorením objektu triedy *LSA*, ktorá má za úlohu samotný výpočet latentnej sémantickej analýzy. Samozrejmosťou je preťaženie metód abstraktnej triedy, ktoré súvisia s poskytovaním základných informácii o module.

Balíček *latent_semantic_analysis* obsahuje ešte spomínanú triedu *LSA*, ktorá predstavuje výpočet latentnej sémantickej analýzy. Pomocou konštruktora, ktorý vyžaduje prístup do indexu pomocou odkazu na objekt triedy *IndexReader* a zoznamu ignorovaných kľúčových slov dôjde k inicializácii objektu triedy *DocumentParser* z abstraktného modulu. Tým sme získali prístup k informáciám o všetkých dokumentoch, ktoré sa nachádzajú v indexe. Následne samotný výpočet spočíva vo vytvorení matice výskytov za pomoci metódy *CreateTable*, ktorej rozmery sú nasledovné. Riadky predstavujú počet dokumentov a stĺpce predstavujú počet všetkých termínov obsiahnutých v indexe. Jednotlivé hodnoty tejto matice získame pomocou vstavanej funkcie Apache Lucene *getTermFrequencies*, ktorá predstavuje počet výskytov termínu v danom dokumente. Následne dôjde k transformácii pomocou singulárneho rozkladu volaním metódy *SVDProcess*, ktorú vykoná nástroj SVDLIBC[7]. Jedná sa o program písaný v programovacom jazyku C, ktorého úlohou je na základe matice výskytov vytvoriť matice U, Σ a V^T. Následne pomocou metódy *ProcessVectors* dôjde k načítaniu matíc Σ a V^T, ktoré maticovým násobením zľava vytvoria maticu podobnosti, na základe ktorej vytvoríme výsledok. Tvorba výsledku je totožná s metódou *SaveResult* z modulu prienik termínov. Výstupný súbor má rovnakú štruktúru ako na obrázku 5.2.

[7] http://tedlab.mit.edu/~dr/SVDLIBC/

5.6 Modul random indexing

Modul random indexing je takisto reprezentovaný vlastným balíčkom s názvom *random_indexing*. V stromovej štruktúre patrí pod balíček *related_docs*. Využíva rovnaký princíp ako predchádzajúce moduly, a to dedenie triedy *ModuleBase* a triedy abstraktného modulu.

Hlavnou triedou je *RandomIndexingRequestHandler*, ktorá je potomkom triedy *ModuleBase* a jej úlohou je spracovanie požiadavky na vyhľadanie sémanticky podobných dokumentov za pomoci random indexingu. Základom je implementácia abstraktnej metódy *handleRequestBody*, ktorá využije metódu *startRequestBody* z abstraktného modulu pre inicializáciu odpovede na požiadavku. Opäť dôjde k načítaniu parametrov, prípadnému použitiu preddefinovaných hodnôt ak neboli špecifikované. Následne sa vytvoria postupne objekty tried *TermVectors*, *DocVectors* a *TableCreator*, ktoré slúžia pre samotný výpočet. Opäť táto trieda obsahuje preťaženie metód abstraktnej triedy, ktoré súvisia so základnými informáciami o module.

Trieda *TermVectors* implementuje rozhranie *VectorStore*, ktoré umožňuje využiť vstavané funkcie pre prácu s vektormi. Hlavnou úlohou konštruktoru je inicializácia index vektorov pre dokumenty a výpočet kontextových vektorov pre termíny. Okrem toho dôjde k inicializácii objektu triedy *DocumentParser*, ktorý nám získava všetky dostupné informácie o dokumentoch z indexu. Ďalšie metódy, ktoré táto trieda obsahuje slúžia pre získanie odkazov na jednotlivé premenné, ktoré boli inicializované v konštruktore.

Ďalšou triedou balíčku *random_indexing* je *DocVectors*, ktorej úlohou je výpočet kontextových vektorov pre dokumenty. Takisto implementuje rozhranie *VectorStore*, ktoré umožňuje pracovať so vstavanými metódami vektorov. Konštruktor si získa index vektory pre dokumenty a kontextové vektory termínov z objektu triedy *TermVectors* a na základe nich dôjde k výpočtu kontextových vektorov pre dokumenty. Ďalšie metódy, ktoré táto trieda obsahuje slúžia pre získavanie premenných, ktoré boli inicializované v konštruktore.

Poslednou triedou je *TableCreator*, ktorá pre inicializáciu vyžaduje odkaz na objekt triedy *DocVectors*. Následne volaním metódy *CreateTable*, dôjde k výpočtu matice podobnosti na základe kontextových vektorov dokumentov. Výpočet výsledku z tejto matice je totožný s metódou *SaveResult* z modulu pre prienik termínov. Rovnakým je aj formát výstupného súboru, ktorý je demonštrovaný na obrázku 5.2.

5.7 Modul modifikovaný random indexing

Modul modifikovaného random indexingu je taktiež reprezentovaný vlastným balíčkom s názvom *modified_random_indexing* a v stromovej štruktúre je pod balíčkom *related_docs*. Implementácia tohto modulu je totožná s modulom random indexing.

Rozdielom je inicializácia triedy *DocVectors*. Kde v konštruktore nedôjde k získaniu index vektorov pre dokumenty z objektu triedy *TermVectors*, ale k vytvoreniu nových nulových kontextových vektorov.

5.8 Modul citation random indexing

Posledný implementovaný modulom je modul citation random indexing, ktorý reprezentuje navrhnutú metódu vyhľadávania sémantickej blízkosti. Je reprezentovaný samostatným balíčkom s názvom *citation_random_indexing*. V stromovej štruktúre sa nachádza pod balíčkom *related_docs*. Základom je implementácia modulu modifikovaného random indexingu, ktorú upravuje a rozširuje o nový rozmer, a to citácie.

Upravuje názov triedy na *CitationRandomIndexingRequestHandler* a táto trieda sa stará o spracovanie požiadavky a implementáciu triedy *DocVectors*. Kde nedôjde k normalizácii kontextových vektorov pre dokumenty. V prípade, že by došlo k normalizácii nebolo by možné objektívne ovplyvniť kontextový vektor dokumentu na základe citácie. K normalizácii dôjde až po modifikácii kontextových vektorov, pred ich použitím pre výpočet matice podobnosti.

Implementáciu modulu modifikovaného random indexingu rozširuje o novú triedu *CitationVectors*, ktorá implementuje rozhranie *VectorStore*, aby mohla využívať vstavané funkcie pre prácu s vektormi. Inicializácia prebiehajúca v konštruktore, ktorý vyžaduje odkaz na objekt triedy *DocVectors*, si získa kontextové vektory dokumentov. Následne dochádza k ovplyvneniu týchto kontextových vektorov na základe vektorov vytvorených pre citácie. Posledným krokom je normalizácia, a tým získame množinu kontextových vektorov dokumentov, ktorú použijeme pre výpočet matice podobnosti.

Výpočet matice podobnosti ako aj vytvorenie výsledku je totožné s implementáciou v module modifikovaného random indexing. Formát výstupného súboru je demonštrovaný na obrázku 5.2.

6 Testovanie

V tejto kapitole je predstavený spôsob testovania jednotlivých metód, ktoré boli v rámci tejto práce implementované. Úvodom je objasnený výber testovacej sady, na základe ktorej prebiehalo testovanie. Nasleduje metodika testovania a návrh samotných testov. Predposlednou kapitolou sú samotné testy a demonštrácia ich niektorých výsledkov. Záverom sú vyhodnotené získané výsledky jednotlivých modulov, ktoré reprezentujú rôzne metódy prístupu k vyhľadávaniu sémanticky podobných článkov.

6.1 Testovacia sada

Základom testovania je výber vhodnej testovacej sady. Pre tieto účely som zvolil voľne dostupný vyhľadávací portál *Microsoft Academic Search*[8]. Jedná sa o vyhľadávač, ktorý obsahuje odkazy na rôzne akademické dokumenty a zdroje z oblasti počítačovej vedy. V súčasnosti táto databáza obsahuje bibliografické informácie pre viac ako 15 miliónov publikácii. Okrem iného sú súčasťou aj informácie o autoroch, ktorých je viac ako 11 miliónov. Pravidelne v týždenných intervaloch dochádza k aktualizácii a pridávaniu nových informácii [16].

Microsoft Academic Search je vhodný pre naše účely z nasledujúcich dôvodov. Jedná sa o pravidelne aktualizovanú databázu, ktorá sa zaoberá zhromažďovaním informácii o publikovaných vedeckých článkoch a napriek tomu je verejne dostupná bez poplatkov. Obsahuje všetky potrebné bibliografické informácie a odkaz na publikáciu v elektronickej podobe, na základe ktorých získame vhodné vstupné dáta pre sémantické vyhľadávanie podobných vedeckých článkov. Všetky uverejnené informácie sú v podobe internetových stránok, ktoré je možné spracovať vhodným webovým extraktorom.

Pre získavanie samotných textov publikácii a takisto aj ich metainformácii sa použije webový extraktor s názvom *AcademicSearch*[9]. Jeho hlavnou úlohou je sťahovanie dostupných publikácii v elektronickom formáte a vytvorenie XML súboru, ktorý obsahuje všetky dostupné metainformácie. Výber získavaných dokumentov je možné ovplyvniť na základe špecifikácie konferencie, v inom prípade dochádza k získavaniu naposledy pridaných publikácii. Všetky získané dáta sa využijú v predspracovaní dokumentov.

Výsledkom sťahovania bolo získanie 9382 vedeckých článkov rozdelených do 87 konferencii. Následne z týchto všetkých dokumentov bola vybraná malá množina, zhruba 60 článkov, ktorá sa následne využije pre testovanie. Veľkosť tejto množiny je malá z dôvodu, že výber dokumentov prebieha ručne pre maximalizáciu správnosti výsledkov. Pri veľkej množine by bola analýza výsledku

[8] http://academic.research.microsoft.com/
[9] Autorom je Michal Jurzykowski.

veľmi zložitá a mohla by obsahovať aj dokumenty, ktoré obsahujú chybné údaje, ktoré vznikli v procese automatickej extrakcii informácii. Výber prebieha nasledovne. Ako prvé je zvolených 10 konferencii, ktoré obsahujú väčší počet dokumentov tak, aby ich tematické zameranie bolo rozdielne. Následne z každej je vybraných 6 dokumentov, ktorých text reprezentuje tému danej konferencie a obsahuje nejaké referencie. Takisto v tomto kroku dôjde ku kontrole, či informácie získané extraktormi odpovedajú skutočnosti a nejedná sa o chybné dáta, ktoré mohli vzniknúť v rámci automatizovaného procesu. Výsledkom je testovacia sada, ktorá je vhodná pre testovanie vyhľadávania sémanticky podobných článkov. Jej vhodnosť je zaručená ručným výberom článkov, ktoré vhodne reprezentujú danú tému a neobsahujú žiadne chybné dáta. Tak isto správnym výberom je zaručená znalosť sémantických vzťahov jednotlivých dokumentov. Vytvorenú testovaciu sadu je možné nájsť na priloženom optickom médiu.

6.2 Návrh testov

Cieľom tejto práce je porovnať spomínané metódy vyhľadávania sémantickej podobnosti. Pre zaručenie objektívnosti budú všetky testy vykonávané na rovnakom počítači, s rovnakou konfiguráciou a nad rovnakou testovacou sadou. Základom testovania budú tri hlavné kritéria:

1. Správnosť výsledkov, ktorá je určená na základe takzvaných rankov.
2. Potrebný čas na spracovanie požiadavky.
3. Využitá operačná pamäť počas spracovania požiadavky.

Cieľom určovania správnosti výsledkov je, že pre skúmaný dokument, ktorý patrí do nejakej konferencie, nájdeme v zoradenej postupnosti podobných dokumentov také články, ktoré patria do rovnakej konferencie, čiže sa zaoberajú rovnakou problematikou. V ideálnom prípade prvých 5 najpodobnejších dokumentov by malo byť z rovnakej konferencie. Pre automatické vyhodnocovanie získaných výsledkov využijeme takzvané ranky. Rank predstavuje číslo, ktoré reprezentuje súčet hodnôt, určených na základe poradia v zoradenej postupnosti všetkých dokumentov pre články z rovnakej konferencie. V ideálnom prípade, keď najpodobnejších 5 dokumentov je z rovnakej konferencie jeho hodnota by bola 10, čo predstavuje súčet indexov 0+1+2+3+4. Týmto spôsobom je možné porovnať jednotlivé metódy s ohľadom na sémantické vyhľadávanie založené na téme konferencie. Čím nižší rank daná metóda získa, tým lepší výsledok dosiahla.

Druhým kritériom testovania je potrebný čas na spracovanie požiadavky. O meranie tohto času sa stará systém pre indexovanie, ktorý disponuje touto funkciou. Jednotlivé časy v milisekundách sú uvedené do výstupu užívateľského rozhrania po dokončení vyhľadávania sémanticky podobných článkov. Týmto spôsobom dochádza k porovnaniu potrebného času pre jednotlivé metódy a samozrejme nižší čas znamená lepší výsledok.

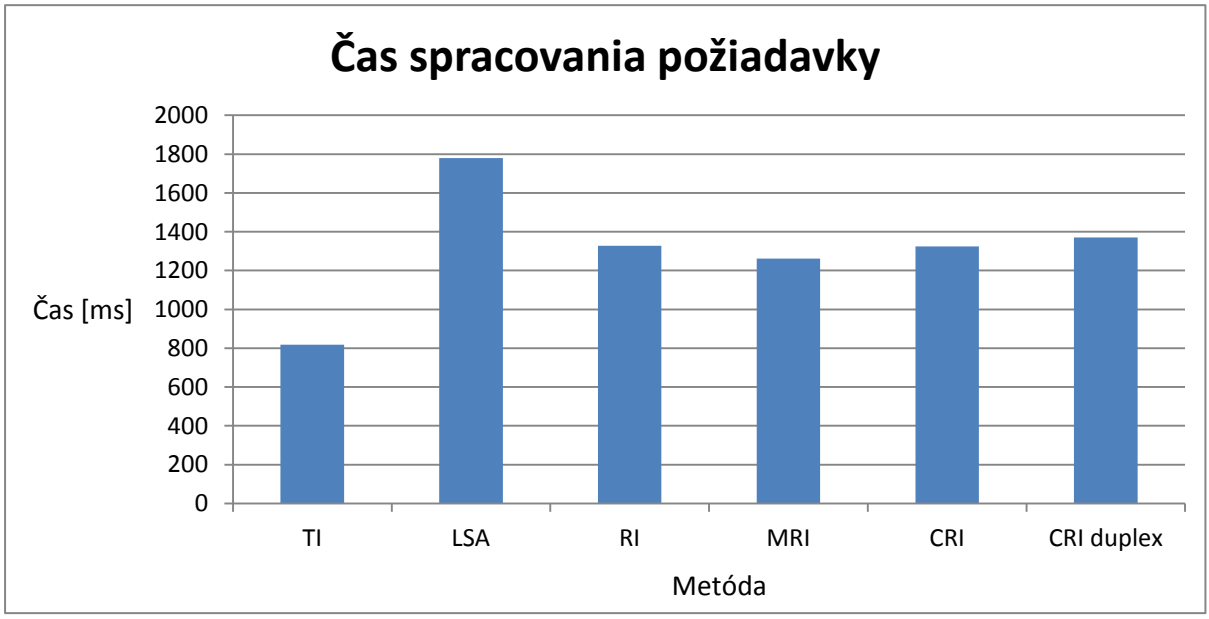
Posledným kritériom testovania je využitá operačná pamäť počas spracovania požiadavky. Pre monitorovanie využitej pamäte sa použije voľne dostupný nástroj *jstat*[10]. Jedná sa o monitorovací nástroj určený pre Javu, ktorý poskytuje informácie o používanej operačnej pamäti. Tým je možné porovnať jednotlivé metódy na základe vyžadovanej pamäti a samozrejme, že nižšia spotreba predstavuje lepší výsledok.

Každý test bude vykonaný desaťkrát a na základe toho vznikne priemerná hodnota, ktorá bude reprezentovať výsledok danej metódy. Následne bude možné z týchto výsledkov graficky, v podobe grafov, zobraziť porovnanie jednotlivých metód.

6.3 Testy

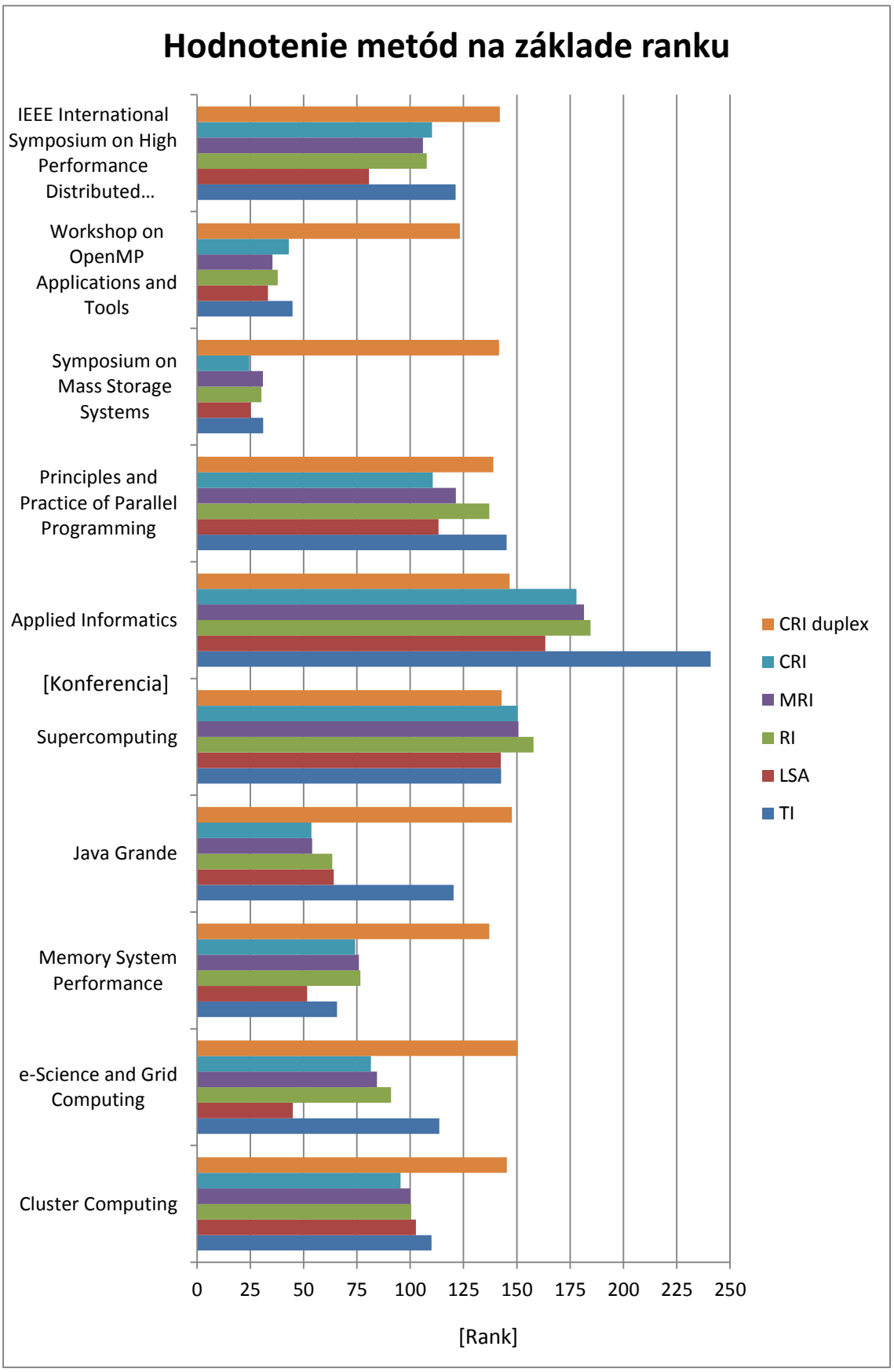
Na nasledujúcich grafoch je možné vidieť získané výsledky počas testovania. Každý graf reprezentuje jedno kritérium z návrhu testov a v kapitole 6.4 sa nachádza podrobná interpretácia týchto výsledkov. Podrobné dáta získané z testovania, na základe ktorých vznikli tieto grafy, je možné nájsť na priloženom optickom médiu.

Počas testovania boli použité nasledujúce parametre jednotlivých metód. Veľkosť kontextových vektorov pre metódy využívajúce random indexing bola rovná 200. U metódy latentnej sémantickej analýzy sa použila veľkosť dimenzie pre singulárny rozklad na základe počtu dokumentov v indexe, ktorý bol rovný 60. Pre metódu random indexing, ktorá využíva väzby citácii bol použitý koeficient súčinu podobnosti rovný hodnote 5. Táto hodnota bola určená experimentálne a následne sa použila aj ako prednastavená hodnota koeficientu tejto metódy.

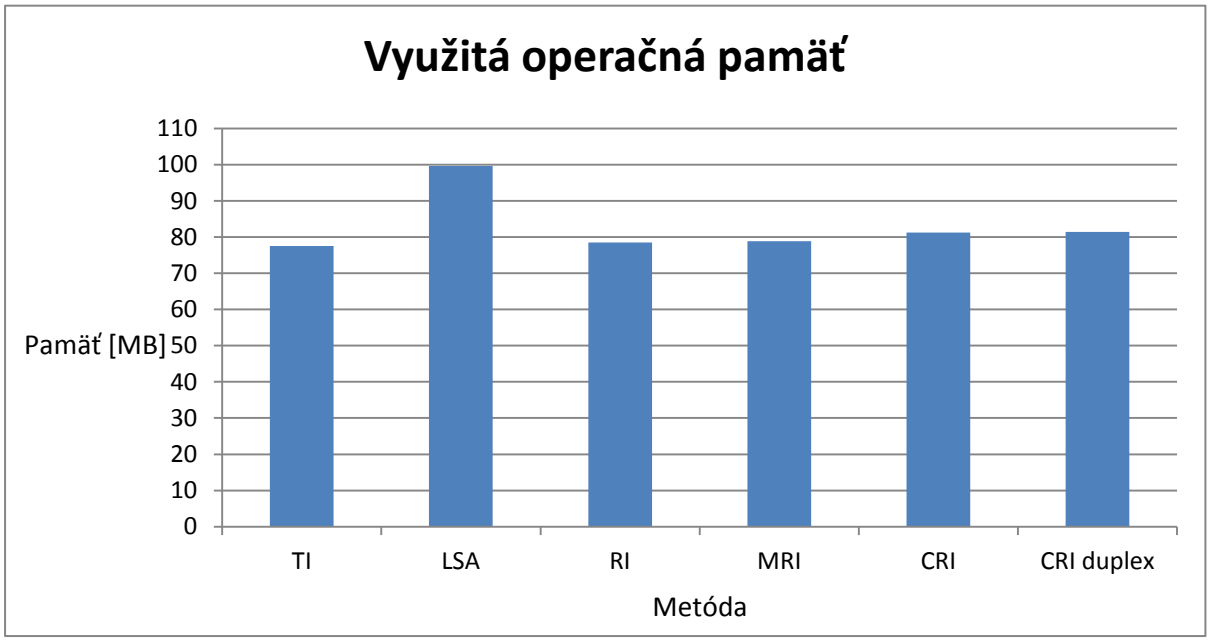


Obrázok 6.1: Graf času spracovania požiadavky

[10] http://download.oracle.com/javase/1.5.0/docs/tooldocs/share/jstat.html



Obrázok 6.2: Graf úspešnosti jednotlivých metód



Obrázok 6.3: Graf využitej operačnej pamäte

Jednotlivé názvy metód sú v grafoch reprezentované skratkami, ktoré sú interpretované nasledovne. *TI* predstavuje metódu prienik termínov, *LSA* označuje metódu latentnej sémantickej analýzy, *RI* metódu random indexing, *MRI* metódu modifikovaného random indexingu, *CRI* predstavuje random indexing založený na citáciách s jednosmernou väzbou a poslednou metódou je *CRI duplex*, ktorá predstavuje random indexing založený na citáciách s obojsmernou väzbou.

6.4 Vyhodnotenie testov

Táto kapitola sa zaoberá vysvetlením získaných výsledkov v jednotlivých testovacích kritériách. Podrobne popisuje pozitívne a negatívne vlastnosti jednotlivých metód, ktoré boli odhalené počas testovania. Záverom každého kritéria je uvedený zostupný zoznam zoradený podľa úspešnosti jednotlivých metód v danej oblasti.

6.4.1 Kritérium potrebného času

Prvým grafom je obrázok 6.1, ktorý reprezentuje čas potrebný na spracovanie požiadavky sémantického vyhľadávania podobných vedeckých článkov. Jednotlivé skúmané metódy sú reprezentované na horizontálnej ose, ich dosiahnutý čas v milisekundách je možné vidieť na vertikálnej ose. Výsledky tohto testovacieho kritéria nepriniesli žiadne neočakávané prekvapenia a poradie jednotlivých metód na základe rýchlosti sa dalo logicky odvodiť. Poradie jednotlivých metód je nasledovné:

1. Prienik termínov – Podľa očakávaní najkratší potrebný čas na spracovanie požiadavky vyžaduje metóda prienik termínov. Je to spôsobené tým, že metóda sa nezaoberá skúmaním kontextu jednotlivých termínov.
2. Modifikovaný random indexing – Ďalšie v poradí sú metódy založené na princípe random indexing. Modifikovaný random indexing získal z týchto metód najlepší čas, nakoľko v druhom kroku spracovania vektorov nemusel použiť náhodné index vektory, ale vytvoril nové nulové.
3. Random indexing – Na treťom mieste sa umiestnil random indexing, ktorý stratil potrebný čas v druhom kroku spracovania vektorov, kde využíval náhodné index vektory pre dokumenty z prvého kroku.
4. Random indexing, jednosmerná väzba citácii – V tesnom závese je metóda random indexing využívajúca jednosmerné väzby citácii, ktorá musí vykonať o jedno spracovanie vektorov viac, ako metóda modifikovaného random indexingu.
5. Random indexing, obojsmerná väzba citácii – Nasleduje random indexing, ktorý využíva obojsmerné väzby citácii na základe čoho je zrejmé, že upravuje dvojnásobný počet vektorov ako metóda s jednosmernou väzbou.
6. Latentná sémantická analýza – Poslednou v poradí sa stala metóda latentnej sémantickej analýzy vďaka singulárnemu rozkladu, ktorý aj na takto malej testovacej sade vyžaduje veľké množstvo času.

6.4.2 Kritérium spotrebovanej operačnej pamäte

Druhý graf, ktorý je zobrazený na obrázku 6.3 a predstavuje využitie operačnej pamäte systému pre indexovanie počas spracovania požiadavky. Z tejto pamäte bolo 60 megabajtov využitých samotným *Apache Solr*, zvyšná pamäť bola využitá pre výpočet sémantickej podobnosti. Vertikálna osa predstavuje využitie pamäte v megabytoch a na horizontálnej ose je možné vidieť jednotlivé metódy. Opäť nedošlo k neočakávaným výsledkom a poradie je nasledovné:

1. Prienik termínov – Podľa očakávaní víťazom tohto testovacieho kritéria sa opäť stala metóda prienik termínov, nakoľko neskúma kontext jednotlivých termínov, a tým nedochádza k veľkému využitiu operačnej pamäte.
2. Random indexing – Na rozdiel od predchádzajúceho testovacieho kritéria, kde bolo využitie náhodných index vektorov pre dokumenty, ktoré boli vytvorené v prvom kroku, vnímané ako spomalenie, tu dochádza k ušetreniu využitej pamäte oproti metóde modifikovaného random indexingu.
3. Modifikovaný random indexing – Na základe toho, že je nutné vytvoriť novú množinu nulových kontextových vektorov pre dokumenty dochádza k zvýšeniu využitej operačnej pamäte oproti metóde random indexing.

4. Random indexing, jednosmerná väzba citácii – Nasleduje metóda, ktorá využíva jednosmerné väzby citácii. Počas výpočtu dôjde k vytvoreniu množiny vektorov pre jednotlivé citácie, čím dochádza k zvýšeniu potrebnej operačnej pamäte.
5. Random indexing, obojsmerná väzba citácii – Podľa očakávania, takmer s identickým využitím pamäte sa umiestnila metóda, ktorá využíva obojsmerné väzby citácii.
6. Latentná sémantická analýza – Opäť na poslednom mieste, vďaka singulárnemu rozkladu sa umiestnila metóda latentnej sémantickej analýzy.

6.4.3 Kritérium správnosti výsledkov

Posledným testovacím kritériom bolo porovnanie jednotlivých metód na základe správnosti výsledkov vyhľadávania sémanticky podobných dokumentov. Výsledný graf pre jednotlivé konferencie, ktoré sú prezentované na vertikálnej ose je možné vidieť na obrázku 6.2. Horizontálna osa predstavuje získaný rank a jednotlivé metódy sú reprezentované v podobe farebných stĺpcov. Toto testovacie kritérium je rozdelené do štyroch skupín, ktoré vznikli na základe výberu článkov a konferencii testovacej sady. Prvou skupinou, ktorá je reprezentovaná štyrmi konferenciami, je množina normálnych dokumentov. Množina normálnych dokumentov znamená, že nie sú skúmané ich vzájomné vzťahy alebo citácie. Jedná sa o náhodný výber dokumentov, ktoré adekvátne reprezentujú danú konferenciu. Druhou skupinou, ktorá je zastúpená dvoma konferenciami, je všeobecná téma konferencie. Jedná sa o konferencie všeobecného charakteru, čo znamená, že aj kľúčové slová zvolených dokumentov majú skôr všeobecný charakter. Treťou skupinou, ktorá je opäť zastúpená dvoma konferenciami, je zameranie na citácie medzi článkami. Jedná sa o výber dokumentov, ktoré majú spoločné citácie, poprípade sa vzájomne citujú. Poslednou skupinou, ktorá je taktiež reprezentovaná dvoma konferenciami, je zameranie na jedinečné citácie. Jedná sa o dokumenty, ktoré nemajú spoločné citácie, ani sa vzájomne necitujú.

Konferencie *Cluster Computing* a *Java Grande* patria do prvej skupiny. Na základe získaných výsledkov pre tieto konferencie najlepší priemerný rank získala metóda random indexing využívajúca jednosmerné väzby citácii. Na základe skúmania dokumentov týchto konferencii bol objavený dostatočný počet spoločných citácii týchto dokumentov, čím došlo k zlepšeniu výsledku modifikovaného random indexingu. Latentná sémantická analýza sa umiestnila na štvrtom mieste, až za metódami využívajúce random indexing, ktoré ju len aproximujú. Na základe skúmania bola zistená príčina, a to malý počet výskytov jednotlivých termínov v samotnom texte dokumentu, čo spôsobilo horšie výsledky singulárneho rozkladu. Posledné miesto obsadila metóda využívajúca obojsmerné väzby citácii a z výsledku je veľmi dobre vidieť, že v tomto prípade nemá zmysel, pretože vykazuje značne horšie výsledky ako referenčná metóda prienik termínov.

Ďalšou konferenciou, ktorá patrí do prvej skupiny je *e-Science and Grid Computing*. Bezkonkurenčne najlepší priemerný rank, takmer o polovicu pred ďalšou metódou, získala latentná

sémantická analýza. Nasleduje metóda využívajúca jednosmerné väzby citácii, nakoľko opäť vďaka niekoľkým vzájomným citáciám došlo k zlepšeniu výsledkov metódy modifikovaného random indexingu. Opäť metóda, ktorá využíva obojsmerné väzby citácii, ukazuje značne horšie výsledky ako referenčná metóda prienik termínov a v tomto prípade jej použitie určite nemá zmysel.

Poslednou konferenciou prvej skupiny je *Memory System Peformance*. Najlepší priemerný rank opäť získala metóda latentnej sémantickej analýzy. Prekvapujúce je druhé miesto, ktoré získala referenčná metóda prienik termínov. Tento výsledok je spôsobený veľmi špecifickými kľúčovými slovami vybraných dokumentov, ako sú názvy programov, ktoré sa zaoberajú alokáciou alebo spracovaním pamäte. Najlepšou z metód random indexing je metóda využívajúca jednosmerné väzby citácii, čo je spôsobené spoločnými citáciami viacerých dokumentov. Posledné miesto opäť obsadila obojsmerná väzba citácii, ktorá vykazuje takmer dvojnásobný priemerný rank ako ostatné metódy.

Druhou skupinou je všeobecná téma, do ktorej patria konferencie *Supercomputing* a *Applied Informatics*. Získané výsledky zobrazujú, že jednotlivé metódy mali so všeobecným výberom kľúčových slov problém. Prekvapujúce je, že metóda využívajúca obojsmerné väzby citácii mala veľmi dobré ranky oproti predchádzajúcim pokusom, ale z celkového hľadiska táto metóda podávala počas celého testovania takmer rovnaké hodnoty rankov, a tým si myslím, že tento výsledok vôbec o ničom nevypovedá. Ostatné metódy aj napriek vysokej hodnote priemerného ranku zachovali poradie, kde latentná sémantická analýza predchádzala ostatným metódam využívajúcim random indexing, z ktorých jednosmerná väzba citácii bola najlepšia. Referenčná metóda prienik termínov v jednom prípade získala priemerný rank takmer totožný s metódou latentnej sémantickej analýzy, čo bolo spôsobené tým, že jednotlivé dokumenty obsahovali veľký počet rovnakých kľúčových slov.

Tretia skupina je zameraná na dokumenty, ktoré obsahujú vzájomné citácie, poprípade ich niektoré referencie sú zhodné. Do tejto skupiny patria konferencie *Principles and Practice of Parallel Programming* a *Symposium on Mass Storage Systems*. Vo výsledkoch pre obe tieto konferencie sa podarilo metóde založenej na jednosmernej väzbe citácii prekonať latentnú sémantickú analýzu. To dokazuje predpokladanú teóriu, že v prípade dobrej testovacej sady, ktorá má správne definované vzájomné vzťahy pomocou citácii, dôjde k prekonaniu výsledkov latentnej sémantickej analýzy pomocou jej aproximácie.

Posledná skupina je zameraná na skúmanie chovania jednotlivých metód v prípade, ak jednotlivé dokumenty nie sú vzájomne citované, ani neobsahujú spoločné referencie. Patria sem konferencie *Workshop on OpenMP Applications and Tools* a *IEEE International Symposium on High Performance Distributed Computing*. V oboch prípadoch najlepší priemerný rank získala latentná sémantická analýza, nasledovaná metódou modifikovaného random indexingu. Čo poukazuje nato, že v prípade zle definovaných vzájomných vzťahov medzi jednotlivými dokumentmi pomocou citácii dôjde k zhoršeniu výsledkov metódy, ktorá využíva jednosmerné väzby citácii aj napriek tomu, že vychádza z výsledku, ktorý získa metóda modifikovaného random indexingu.

Výsledné poradie jednotlivých metód na základe správnosti výsledkov vyhľadávania sémanticky podobných vedeckých článkov je nasledovné:

1. Latentná sémantická analýza – Podľa očakávaní metóda latentnej sémantickej analýzy podávala najlepšie výsledky. Na druhej strane je pre jej fungovanie potrebné mať dostatočný počet výskytov kľúčových slov v texte, inak môže dôjsť k zhoršeniu výsledkov singulárneho rozkladu.
2. Random indexing, jednosmerná väzba citácii – Ako druhá skončila metóda založená na jednosmernej väzbe citácii, ktorá typicky funguje správne, ale v extrémnych prípadoch, kde články neobsahujú vzájomné citácie môže spôsobiť zhoršenie výsledkov.
3. Modifikovaný random indexing – Na treťom mieste sa umiestnila metóda modifikovaného random indexingu, ktorá podávala dobré výsledky počas celého testovania.
4. Random indexing – O niečo horšie výsledky podávala metóda random indexing, ktorá sa umiestnila na štvrtom mieste.
5. Prienik termínov – Ako referenčná metóda poslúžil prienik termínov, ktorý počas testovania podával očakávané výsledky.
6. Random indexing, obojsmerná väzba citácii – Ako najhoršia metóda sa prejavila obojsmerná väzba citácii. Jedine v prípade všeobecných kľúčových slov podávala vhodné výsledky, ale s ohľadom na celý priebeh testovania podávala stále rovnaké hodnoty, ktoré nie sú dostačujúce ani na prekonanie referenčnej metódy. Ovplyvňovanie oboch vektorov počas výpočtu spôsobilo zásadný problém. Tento problém spočíval v tom, že kontext jedného dokumentu bol ovplyvnený kontextom nejakého citovaného dokumentu a naopak. Následne počas výpočtu sa takýmto spôsobom menili kontexty všetkých dokumentov na toľko, až si nezachovali vzťahy, ktoré získali pomocou vytvorenia kontextov na základe kľúčových slov. Čím nedošlo k zlepšeniu výsledkov modifikovaného random indexingu, ale k zmene celých výsledkov podľa úplne nových kontextov.

7 Záver

Hlavným cieľom tejto práce bolo zoznámiť sa s existujúcimi metódami pre výpočet sémantickej blízkosti vedeckých dokumentov a na základe týchto informácii navrhnúť novú metódu, ktorá dokáže prekonať súčasné metódy. Z existujúcich metód som zvolil prienik termínov, latentnú sémantickú analýzu, random indexing a modifikovaný random indexing. Metóda navrhnutá v tejto práci vychádza z modifikovaného random indexingu a prináša nový prvok, a to citácie, ktoré predstavujú statickú väzbu medzi článkami.

Pre porovnanie jednotlivých algoritmov bolo nutné vytvoriť komplexný systém, ktorého úlohou bolo spracovanie požadovaných vedeckých článkov tak, aby bolo možné opätovne vyhľadávať sémanticky podobné dokumenty pomocou zvolenej metódy. Pred samotným porovnaním bolo nutné vytvoriť testovaciu sadu, ktorá obsahovala dokumenty s vopred známymi vzťahmi a navrhnúť testy, na základe ktorých dôjde k získaniu výsledkov. Posledným krokom bola interpretácia týchto výsledkov a vyhodnotenie metód v jednotlivých kritériách.

Na základe výsledkov, ktoré boli získané počas testovania jednotlivých metód za účelom porovnania, som určil štyri najvhodnejšie algoritmy sémantického vyhľadávania vedeckých dokumentov.

Najlepšou metódou sa stala latentná sémantická analýza, ktorá počas testovania podávala najpresnejšie výsledky sémantickej podobnosti. Jedinou nevýhodou tejto metódy je potrebný čas a spotrebovaná operačná pamäť počas výpočtu, čo je spôsobené singulárnym rozkladom. Využitie tejto metódy je vhodné ak potrebujeme veľmi presné výsledky a ich výpočet bude vykonávaný vo väčších intervaloch.

Spoločne na druhom mieste sa umiestnili metódy modifikovaný random indexing a random indexing, využívajúci jednosmernú väzbu citácii. Jedná sa o kompromis medzi metódami, ktoré skúmajú kontext dokumentov. Výber jednej z týchto dvoch metód závisí na vstupnej množine dokumentov. Z testovania bolo vidieť, že ak dokumenty nemajú vzájomné vzťahy v podobe citácii dochádzalo k zhoršeniu výsledku. Využitie jednej z týchto metód je vhodné, ak potrebujeme častejšie vytvárať nové výsledky podobnosti.

Na treťom mieste sa umiestnila metóda prienik termínov, ktorá aj napriek tomu, že neskúma kontext jednotlivých dokumentov, dokáže v rýchlom čase poskytovať dostačujúce výsledky. Využitie si táto metóda nájde v systémoch, ktoré sú pravidelne aktualizované veľkým počtom dokumentov.

Nevýhodou prístupu metódy navrhovanej v tejto práci je spoliehanie sa na vhodnú množinu dokumentov a ich metainformácie. V prípade, že dostupný zoznam literatúry je neúplný alebo došlo k chybe pri automatizovanej extrakcii, typicky dôjde k zhoršeniu výsledkov, ktoré sú vypočítané pomocou modifikovaného random indexingu. Riešením tohto problému môže byť dodatočná manuálna kontrola získaného zoznamu referencii pre nové dokumenty.

Možností rozšírenia sémantického vyhľadávania podobných vedeckých článkov môže byť niekoľko. Zaujímavé by bolo pridávanie nových aspektov do vyhľadávania, napríklad v podobe autorov, na základe ktorých by došlo k úprave kontextu dokumentu. Poprípade sa môže jednať o zlepšenie automatizovaných procesov, napríklad extrakcie kľúčových slov, ktorá zohráva dôležitú rolu vo vytváraní kontextov pre jednotlivé dokumenty. Takisto proces automatickej extrakcie zoznamu referencii by bolo vhodné rozšíriť o získavanie počtu citovaných celkov z danej referencie, čím by mohlo dochádzať k priamo úmernému ovplyvneniu váhy podobnosti. Tým by dochádzalo k rozlíšeniu jednotlivých referencii a ich váhy v rámci podobnosti, pretože častejšie citované články typicky majú podobnejší kontext ako články, ktoré sú citované raz. Výsledky získané v tejto práci boli vytvorené na malej testovacej sade, určite by bolo zaujímavé sledovať chovanie jednotlivých algoritmov na väčšom množstve dokumentov, najmä rastúcu spotrebu operačnej pamäte a potrebného času na spracovanie, nakoľko zložitosť jednotlivých algoritmov je kvadratická.

Literatúra

[1] Apache Lucene. [online] [cit. 2011-04-22]
Dostupné na URL: <http://lucene.apache.org/java/docs/index.html>

[2] Apache Solr. [online] [cit. 2011-04-22]
Dostupné na URL: <http://lucene.apache.org/solr/>

[3] Götze Jürgen, Rieder Peter, Hekstra J. Gerben: SVD-Updating Using Orthonormal μ-Rotations, The Journal of VLSI Signal Processing, Volume 14, Number 1, 1996.

[4] Harold Rusty Elliote, Means W Scott: XML In A Nutshell, O'Reilly & Associates, Inc., 2004, ISBN 9780596007645.

[5] Hecht-Nielsen Robert: Neurocomputing, Addison-Wessley Pub. Co., 1990, ISBN 0201093553.

[6] Homes P. Michael, Gray G. Alexander, Isbell Lee Charles Jr.: Fast SVD for Large-Scale Matrices, In Workshop on Efficient Machine Learning at NIPS, 2007.

[7] jetty://. [online] [cit. 2011-04-22]
Dostupné na URL: <http://jetty.codehaus.org/jetty/>

[8] Jörg Brigitte: Towards the Nature of Citations, Poster Proceedings of Formal Ontology in Information Systems 2008, Fifth International Conference, 2008.

[9] Kanerva Pentti, Kristoferson Jan, Holst Anders: Random Indexing of Text Samples for Latent Semantic Analysis, In L.R. Gleitman and A.K. Josh, Proceeding 22nd Annual Conference of the Cognitive Science Society, 2000.

[10] Kanerva Pentii: Sparse Distributed Memory, The MIT Press, 1988, ISBN 0262111322.

[11] Karlgren Jussi, Sahlgren Magnus: From Words to Understanding, Computing with Large Random Patterns, 2001.

[12] Krátky Michal: Využití SVD pro indexování latentní sémantiky, Technická zpráva, 2002.

[13] Kyjovský Marek: Extrakce klíčových slov z vědeckých článků, diplomová práce, Brno, FIT VUT v Brně, 2010.

[14] Landauer Thomas, Dumais Susan: A Solution to Plato's Problem: The Latent Semantic Analysis Theory of the Acquisition, Induction, and Represantation of Knowledge, Psychological Review, Number 104, 1997.

[15] Landauer K. Thomas, Foltz W. Peter, Laham Darrell: An introduction to Latent Semantic Analysis, In Discourse Processes, No. 25, 1998.

[16] Microsoft Academic Search. [online] [cit. 2011-04-28]
Dostupné na URL: <http://academic.research.microsoft.com/About/Help.htm>

[17] Moonen Marc, Dooren Van Paul, Vandewalle Joos: An SVD Updating Algorithm for Subspace Tracking, SIAM J. Matrix Anal. Appl., Volume 13, Number 14, Pages 1015-1038, 1992.

[18] Novák Ján: Sémantická blízkost termínů, bakalářská práce, Brno, FIT VUT v Brně, 2009.

[19] Salton Gerard, Buckley Christopher: Term-weight approaches in automatic text retrieval, Information Processing & Management, Volume 24, Number 5, Pages 513-523, 1988.

[20] Schmid Helmut: TreeTagger. [online] [cit. 2011-04-22].
Dostupné na URL: < http://www.ims.uni-stuttgart.de/TreeTagger/DecisionTreeTagger.html>

[21] Shalgren Magnus: An introduction to Random Indexing, In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE, 2005.

[22] Shalgren Magnus, Cöster Rickard: Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization, COLING '04 Proceedings of the 20th international conference on Computional Linguistics, 2004.

[23] Tougas E. Jane, Spiteri J. Raymond: Updating the Partial Singular Value Decomposition in Latent Semantic Indexing, Computational Statistics & Data Analysis, Volume 52, Issue 1, Pages 174-183, 2006.

[24] Widdows Dominic, Ferraro Kathleen: Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application, Proceedings of the Sixth International Language Resources and Evaluation, 2008.

Zoznam príloh

Príloha A. Užívateľská príručka predspracovania dokumentov

Inštalácia

Samotná inštalácia a konfigurácia jednotlivých súčastí prebieha pomocou skriptu *install.sh*, ktorý je umiestnený v koreňovom adresári aplikácie. Pred spustením tohto skriptu je nutné mu prideliť patričné práva pre spustenie pomocou príkazu *chmod +x install.sh*.

Pre správne fungovanie aplikácie je vhodné ju inštalovať a spúšťať na školských serveroch, ktoré obsahujú všetky potrebné súčasti. Jedná sa najmä o programy *ant, java, python* a *tar*. Okrem iného školské servery obsahujú ďalšie moduly naprogramované v jazyku Python v rámci projektu *ReReSearch*, na ktoré sa odkazujú jednotlivé skripty.

Používanie

Spustenie aplikácie predspracovania dokumentov spočíva v spustení skriptu *indexer.sh*. Tento skript vyžaduje tri povinné parametre a jeden voliteľný. Poradie týchto parametrov je ľubovoľné a ich znenie je nasledovné:

- -d <adresár> – povinný parameter, ktorý definuje cestu k pracovnému adresáru aplikácie, do ktorého budú vytvárané medzi výsledky jednotlivých krokov predspracovania.
- -c <adresár> – povinný parameter, ktorý definuje adresár korpusu obsahujúci všetky zdrojové dokumenty podliehajúce predspracovaniu.
- -t <adresár> – povinný parameter, ktorý definuje výstupný adresár aplikácie, do ktorej budú umiestnené výsledné XML súbory v požadovanom formáte systému pre indexovanie.
- -n <číslo> – voliteľný parameter, ktorý definuje požadovaný počet termínov, ktoré sú vyextrahované počas automatizovaného procesu extrakcie kľúčových slov. Štandardná hodnota je 50.

Príloha B. Užívateľská príručka systému pre indexovanie

Inštalácia

Inštalácia prebieha pomocou skriptu *install.sh*, ktorý je umiestnený v koreňovom adresári aplikácie. Pred spustením tohto skriptu je nutné mu prideliť patričné práva pomocou príkazu *chmod +x install.sh*. Tento skript vyžaduje program *tar* a administrátorské systémové práva. Počas inštalácie dôjde k inicializácie knižnice potrebnej pre singulárny rozklad používaný latentnou sémantickou analýzou.

Konfigurácia

Pred samotným spustením systému pre indexovanie je nutná jeho konfigurácia. Konfigurácia prebieha pomocou dvoch konfiguračných súborov, ktoré sú umiestnené v zložke *solr/conf* a jedného archívu umiestneného v zložke *webapps*, ktorý obsahuje všetky používané knižnice.

Prvým krokom konfigurácie je definícia schémy indexu, ktorá sa prevádza v konfiguračnom súbore *schema.xml*. Systém obsahuje predkonfigurovanú schému, ktorú stačí pozmeniť pre naše účely. Polia *id*, *title*, *conference* a *references* sú reprezentované textovým reťazcom a budú používať preddefinovaný typ string. Pre pole *text* využijeme preddefinovaný typ text a povolíme tvorbu *termVectors*, ktorá nám umožní vstavané vyhľadávanie početností termínov v danom poli. Posledným používaným polom je *terms*, ktoré predstavuje kľúčové slová oddelené čiarkou, a preto je nutné definovať oddeľovač v podobe nového dátového typu. Výslednú modifikáciu konfiguračného súboru *schema.xml* je možné vidieť na obrázku B.1.

```
<schema name="example">
  <types>
    <fieldtype name="text_comma" class="solr.TextField">
      <analyzer>
        <tokenizer class="solr.PatternTokenizerFactory" pattern=",*"/>
      </analyzer>
    </fieldtype>
  </types>
  <fields>
    <field name="id" type="string" indexed="true" stored="true" required="true"/>
    <field name="title" type="string" indexed="true" stored="true" multiValued="false"/>
    <field name="conference" type="string" indexed="true" stored="true" multiValued="false"/>
    <field name="text" type="text" indexed="true" stored="true" multiValued="false" termVectors="true"/>
    <field name="terms" type="text_comma" indexed="true" stored="true"/>
    <field name="references" type="string" indexed="true" stored="true" multiValued="true"/>
  </fields>
</schema>
```

Obrázok B.1: Modifikácia konfiguračného súboru *schema.xml*

Ďalším krokom konfigurácie systému pre indexovanie je modifikácia súboru *solrconfig.xml*, ktorý umožňuje podrobné nastavenie systému. Pre nás je zaujímavá definícia nových odkazov pre webový server v podobe *requestHandler*, ktoré budú odkazovať na implementácie zásuvných modulov, a tým budú sprístupnené pre interakciu s užívateľom. Modifikáciu konfiguračného súboru *solrconfig.xml*, ktorá odpovedá implementácii modulov navrhnutých v tejto práci je možné vidieť na obrázku B.2.

```
<config>
  <requestHandler name="/related_docs/citation_random_indexing"
    class="related_docs.citation_random_indexing.CitationRandomIndexingRequestHandler"/>
  <requestHandler name="/related_docs/term_intersection"
    class="related_docs.term_intersection.TermIntersectionRequestHandler"/>
  <requestHandler name="/related_docs/random_indexing"
    class="related_docs.random_indexing.RandomIndexingRequestHandler"/>
  <requestHandler name="/related_docs/modified_random_indexing"
    class="related_docs.modified_random_indexing.ModifiedRandomIndexingRequestHandler"/>
  <requestHandler name="/related_docs/latent_semantic_analysis"
    class="related_docs.latent_semantic_analysis.LatentSemanticAnalysisRequestHandler"/>
</config>
```

Obrázok B.2: Modifikácia konfiguračného súboru *solrconfig.xml*

Tretím krokom konfigurácie systému je nahranie jednotlivých zásuvných modulov a ich knižníc. Toto je možné vykonať dvoma spôsobmi, prvým je nakopírovanie do zložky *lib*, kde sa nachádzajú knižnice inicializované pri štarte systému. Druhým spôsobom je modifikácia súboru *solr.war*, ktorý sa nachádza v zložke *webapps*. Jedná sa o archív, ktorý okrem iného obsahuje taktiež zložku *lib* v adresári *WEB-INF*, do ktorej je možné umiestniť požadované moduly a knižnice.

Na priloženom optickom médiu je možné nájsť systém pre indexovanie, ktorý už je vopred nakonfigurovaný podľa týchto pokynov a obsahuje všetky zásuvné moduly, ktoré boli vytvorené v rámci tejto práce.

Inicializácia indexu

Pred vyhľadávaním sémanticky podobných vedeckých dokumentov je nutné naplniť index požadovanými článkami. Toto je možné pomocou skriptu *post.sh*, ktorý sa nachádza v koreňovom adresári systému pre indexovanie. Pred plnením samotného indexu je nutné aby systém pre indexovanie bol spustený.

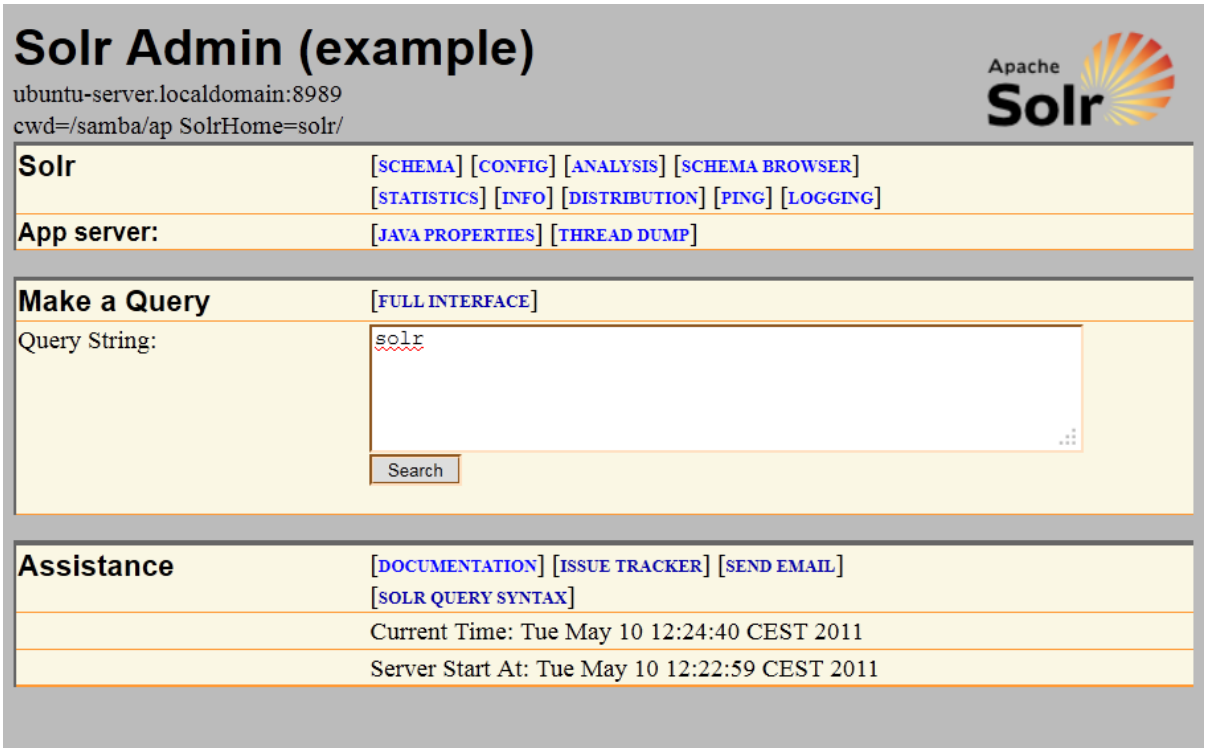
Tento skript vyžaduje jeden povinný parameter, ktorý určuje zložku obsahujúcu XML súbory vytvorené počas predspracovania dokumentov. Následne odošle všetky tieto súbory do systému pre indexovanie a nakoniec vykoná takzvaný *commit*, ktorým potvrdí odoslané dáta a tie sa prejavia do indexu.

Predpokladaná adresa systému pre indexovanie je *localhost* a použitý port *8989*. V prípade, že sa systém pre indexovanie nachádza na inej adrese je nutné modifikovať skript *post.sh*. Jedná sa o jednoduchú zmenu premennej *URL* v zdrojovom kóde s požadovanou cieľovou adresou.

Používanie systému pre indexovanie

Systém pre indexovanie funguje ako proces na pozadí a jeho používanie spočíva v pristupovaní k užívateľskému rozhraniu v podobe webových stránok. Toto užívateľské rozhranie je v predkonfigurovanej verzii na adrese http://0.0.0.0:8989/solr.

Základom systému pre indexovanie je administrácia, ktorá umožňuje monitorovanie a správu systému. Na obrázku B.3 je možné vidieť vizualizáciu tejto administrácie. Je možné zobraziť aktuálne nastavenie konfiguračných súborov, zobraziť rôzne štatistiky, napríklad počet dokumentov v indexe, dobu prevádzky systému, využitie jednotlivých zásuvných modulov a ďalšie. Okrem iného je možné aj vyhľadávať v indexe na základe kľúčových slov.



Obrázok B.3: Administrácia systému pre indexovanie

Druhou časťou užívateľského rozhrania sú samotné zásuvné moduly, ktoré sú prístupné na adresách definovaných v konfiguračnom súbore *solrconfig.xml*. Vzhľad alebo zobrazované informácie závisia len na danej implementácii modulu. Jednotlivé zásuvné moduly môžu pristupovať do pracovného adresára aplikácie a majú právo spúšťať ďalšie programy, ktoré sú nainštalované na danom počítači.

Zásuvné moduly implementované v tejto práci majú jednotný výstup užívateľského rozhrania, ktorý reprezentuje XML dokument obsahujúci parametre požiadavky na vyhľadanie sémantických dokumentov, čas potrebný na vykonanie a správu o ukončení výpočtu. Príklad výstupu je možné vidieť na obrázku B.4. Samotný výstup sémantického vyhľadávania sa nachádza v koreňovom priečinku systému pre indexovanie v súbore s požadovaným názvom. Pre prístup k jednotlivým metódam sa využijú nasledujúcu adresy, ktoré boli definované v konfiguračnom súbore *solrconfig.xml*:

- Prienik termínov – http://0.0.0.0:8989/solr/related_docs/term_intersection/?q=id:*
- Latentná sémantická analýza – http://0.0.0.0:8989/solr/related_docs/latent_semantic_analysis/?q=id:*
- Random indexing – http://0.0.0.0:8989/solr/related_docs/random_indexing/?q=id:*
- Modifikovaný random indexing – http://0.0.0.0:8989/solr/related_docs/modified_random_indexing/?q=id:*
- Random indexing, využívajúci väzbu citácii – http://0.0.0.0:8989/solr/related_docs/citation_random_indexing/?q=id:*

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">1725</int>
  </lst>
  <str name="query">id:*</str>
  <arr name="ignore">
    <str/>
  </arr>
  <int name="dimension">60</int>
  <str name="outputFile">LSA_output.xml</str>
  <str name="rankFile">LSA_rank.txt</str>
  <str name="Result">Finished!</str>
</response>
```

Obrázok B.4: Výstup užívateľského rozhrania pre metódu latentnej sémantickej analýzy

V prípade, že daná metóda podporuje nejaké parametre, je ich možné špecifikovať pridaním *&parameter=hodnota* na koniec požadovanej adresy. Zoznam podporovaných parametrov všetkými metódami je nasledovný:

- ignore=”term1,term2,...” – predstavuje zoznam kľúčových slov oddelených čiarkami, ktoré majú byť vylúčené z výpočtu, štandardná hodnota je prázdny zoznam.
- outputFile=“filename“ – predstavuje názov výstupného súboru sémantickej podobnosti pre dokumenty, štandardný názov je *method_output.xml*.

- rankFile=”filename” – predstavuje názov výstupného súboru, ktorý vyjadruje ranky získané jednotlivými dokumentmi, štandardný názov je *method_rank.txt*.

Parametre pre latentná sémantická analýza:

- dimension=number – predstavuje veľkosť dimenzie singulárneho rozkladu, štandardná hodnota je počet dokumentov v indexe.

Parametre pre random indexing a modifikovaný random indexing:

- dimension=number – predstavuje veľkosť kontextových vektorov, ktoré budú použité počas výpočtu, štandardná hodnota je rovná 200.
- seed=number – predstavuje inicializačnú hodnotu pre pseudonáhodný generátor čísel.

Parametre pre random indexing, využívajúci väzbu citácii:

- cit_q=number – predstavuje koeficient súčinu podobnosti pre ovplyvnenie kontextovým vektorom citácie, štandardná hodnota je 5.
- cit_duplex=boolean – predstavuje príznak obojsmernej väzby citácii, štandardná hodnota je false.
- dimension=number – predstavuje veľkosť kontextových vektorov, ktoré budú použité počas výpočtu, štandardná hodnota je rovná 200.
- seed=number – predstavuje inicializačnú hodnotu pre pseudonáhodný generátor čísel.

Príloha C. Priložené CD

Priložené CD obsahuje nasledujúce dáta:

- Adresár */Bin/PreProcessing* – obsahuje všetky potrebné programy, ktoré sú používané predspracovaním dokumentov.
- Adresár */Bin/Solr* – obsahuje predkonfigurovaný systém pre indexovanie, vrátane modulov pre sémantické vyhľadávanie vypracovaných v tejto práci.
- Adresár */Example* – obsahuje príklad výstupu systému pre indexovanie pre všetky implementované metódy v tejto práci.
- Adresár */JavaDoc* – obsahuje programovú dokumentáciu, vo forme *JavaDoc*, zásuvných modulov systému pre indexovanie.
- Adresár */Results* – obsahuje výsledné dáta testovania, ktoré boli použité ako zdrojové dáta pre grafy publikované v kapitole 6.3.
- Adresár */Source/related_docs* – obsahuje zdrojové kódy jednotlivých zásuvných modulov systému pre indexovanie a súbory nutné pre importovanie projektu do vývojového prostredia *NetBeans*.
- Adresár */Source/SemanticSimilarity* – obsahuje zdrojový kód aplikácie, ktorá má za úlohu spojiť získané informácie z predspracovania dokumentov do výsledného formátu, ktorý je akceptovaný systémom pre indexovanie.
- Adresár */TestingSet* – obsahuje testovaciu sadu, ktorá bola vytvorená počas testovania jednotlivých modulov vypracovaných v tejto práci.
- Adresár */Text* – obsahuje úplné znenie technickej správy vo formáte *PDF*.