Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Rozšiřování dotazu pomocí sémantických prostorů

Plzeň, 2013 Petr Matejovič

Tady bude originál zadání práce (s razítkem)

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Tachově dne 8. srpna 2013

Petr Matejovič

Poděkování

Rád bych tímto poděkoval svým nejbližším, kteří mě v průběhu studia podporovali. Děkuji také Ing. Lubomíru Krčmářovi za jeho odborné vedení, přínosné konzultace a čas, který mi věnoval. Snad největší díky patří mému kolegovi Ing. Petru Vajskebrovi, který mi svým benevolentním přístupem studium umožnil.

Abstract

Information plays more essential role then ever before. The world is overloaded with information of various quality. Therefore, the increased pressure is put on improving and developing new algorithms to find relevant information.

This thesis deals with the application of Semantic Spaces in the task of automatic expansion of the user´s query in order to improve the precision and recall of the answer.

The key task of the thesis is to design algorithms for query expansion based on Semantic Spaces and to design a user-friendly application which enables to test these algorithms in batches of supplied data. Thesis deals with the evaluation of collected results with the help of manually annotated data.

Obsah

1 Úvod

Svět je zahlcen digitálními daty. Data nejsou strukturována, jsou ukládána v různých formátech, různých jazycích, jejich kvalita bývá odlišná a není jim přiřazen žádný význam. Úloha vyhledávání relevantních informací se stává čím dál náročnější.

Běžné vyhledávací systémy fungují tak, že na uživatelův dotaz vrací seznam nalezených dokumentů. Pokud uživatel hledaný dokument nenajde hned na první stránce, snaží se svůj dotaz změnit tak, že k jednotlivým slovům hledá alternativy, které opět nechá vyhledat.

Neměl by ale tyto alternativy vymýšlet sám vyhledávací systém?

1.1 Téma

Tato práce se zabývá návrhem a realizací vyhledávacího systému nad korpusem textů iniciativy CLEF[1] a možnostmi automatického rozšiřování uživatelova dotazu pomocí sémantických prostorů (dále SP[2]).

Cílem práce je navrhnout takové rozšíření uživatelova dotazu, které proces vyhledávání vylepší v tom smyslu, že na původní dotaz vrátí uživateli více dokumentů vyhovujících jeho požadavkům.

Cílem práce naopak není pochopení jednotlivých algoritmů pro vytváření SP ani vytvoření vlastního vyhledávacího stroje.

Vedlejším cílem práce je vytvořit uživatelsky přívětivé a rozšiřitelné prostředí, které poskytne veškerý komfort pro vyhledávání, návrh a testování různých způsobů rozšiřování dotazů.

[1] Cross-Language Evaluation Forum, http://www.clef-initiative.eu

[2] Vektorový sémantický prostor (*Vector Space Models of Semantics*).

2 TEORETICKÝ ÚVOD

Běžné vyhledávací systémy pracují na základě booleovského nebo vektorového modelu a výrokové logiky. Slabost těchto modelů spočívá v tom, že neuvažují význam jednotlivých slov. Jiné vyhledávací systémy implementují slovníky (tezaury[3] či ontologie[4]). Jejich nevýhodou je fakt, že někdo tyto slovníky musí manuálně sestavit a průběžně aktualizovat. Google pro lepší vyhledávání používá metodu *PageRank*[5], její nevýhodou je, že relevantní dokument umístěný na neznámém webu nebude mezi prvními nalezenými a uživatel ho mezi získanými dokumenty bude muset hledat.

Jedním z hlavních nedostatků využití počítačů v úloze vyhledávání je skutečnost, že počítače nerozumí významu lidského jazyka. Některé nové metody se proto zabývají sémantickou analýzou textu, sémantikou ve smyslu významu slova (resp. věty, kontextu).

Se sémantickou analýzou souvisí sémantické prostory (dále SP). SP lze využít k určení vztahů mezi slovy nacházející se v různých kontextech a tím do jisté míry zjistit jejich význam.

Tato kapitola čerpá z [1] a [2]. Slouží k uvedení čtenáře do problematiky SP, poskytuje základní teoretické znalosti, terminologii a zběžný přehled technik, které se používají při vytváření SP. Kapitola je členěna do několika sekcí. Nejprve je vysvětlen termín sémantický prostor, dále se kapitola věnuje zpracování textových informací směrem k jeho vytvoření a popisuje k tomu potřebné algoritmy.

2.1 Vyhledávání dokumentů

V tomto oddílu je čerpáno z [3] a jsou zde stručně vyjmenovány základní metody vyhledávání dokumentů a popsána základní terminologie vyhodnocení odpovědi na uživatelův dotaz.

*Obor **získávání informací** je součást informatiky zabývající se ukládáním, údržbou a vyhledáváním ve velkém množství informací. Informace mohou být textové, obrazové, zvukové nebo multimediální.*

***Dokumentografický informační systém (dále DIS)** je prostředek, který slouží k reprezentaci a uložení informací pro pozdější potřebu. Dokumentů zpracovávaných DIS může být velké množství. Důraz je kladen na snadný a rychlý přístup k uloženým informacím.*

Kolekce dokumentů je pro účely vyhledávání zanalyzována a vymodelována do vhodné formální vnitřní reprezentace. Tomuto procesu se říká *indexace*.

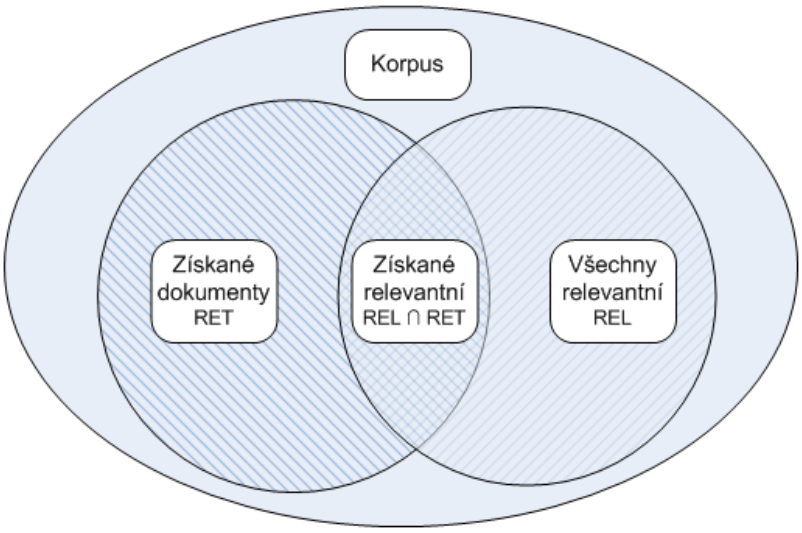
[3] Slovník, který uživateli nabízí seznam synonym, někdy i antonym.

[4] Ontologií je míněn výslovný a formalizovaný popis určité problematiky pomocí slovníku.

[5] Metoda založená na struktuře hypertextových odkazů jako "doporučování" stránek. Stránky hodnotí na stupnici od 1 do 10. Uvažuje váhu slov, atraktivitu i důvěryhodnost stránky. Viz http://cs.wikipedia.org/wiki/PageRank.

2.1.1 Relevance nalezených dokumentů

Jak je vysvětleno v [3], po přijetí uživatelského *dotazu* DIS hledá takové dokumenty, které jsou vůči dotazu *relevantní*. *Odpovědí* na dotaz je seznam dokumentů, které DIS za relevantní považuje. Dokumenty se řadí podle *hodnotící strategie* DIS. *Relevancí odpovědi* se rozumí relevance všech dokumentů získaných na základě vyhledávacího dotazu.



Obrázek 2-1: Rozdělení odpovědi na dotaz z hlediska relevance.

Dokumenty mohou být k dotazu relevantní různou měrou, závislou na subjektivním měřítku uživatele. Pro následující text definujme pojmy (obrázek 2-1):

- **RET** - počet vrácených dokumentů, o kterých si systém myslí, že jsou relevantní,
- **REL** - počet všech relevantních dokumentů k danému dotazu,
- **REL ∩ RET** - počet vrácených relevantních dokumentů, o kterých si uživatel myslí, že jsou relevantní (uspokojují jeho požadavky).

V ideálním případě je REL = RET.

2.1.2 Vyhodnocení odpovědi

Mezi základní měřítka vyhodnocení relevance odpovědi patří přesnost (precision) a úplnost (recall):

- ***Přesnost P*** = $\frac{|\mathrm{REL} \cap \mathrm{RET}|}{|RET|}$ je pravděpodobnost, že získaný dokument je relevantní.
- ***Úplnost R*** = $\frac{|\mathrm{REL} \cap \mathrm{RET}|}{|REL|}$ je pravděpodobnost, že bude získán relevantní dokument.

Oba koeficienty jsou závislé na subjektivním názoru uživatele. V ideálním případě je P = R = 1, tj. v odpovědi jsou zařazeny právě a pouze všechny relevantní dokumenty k dotazu.

2.1.3 Klasické vyhledávací modely

Podle [3] používají DIS různé modifikace booleovského či vektorového modelu.

Booleovský model

Booleovský model je založen na teorii množin, výrokové logice a *principu úplné shody* dokumentu s dotazem ve smyslu výrokové formule (dotazy se skládají ze slov a logických operátorů *and, or, nor, xor* atd.). Za relevantní jsou považovány ty dokumenty, které zcela vyhovují formuli (dotazu), tj. při vyhodnocení formule pro zkoumaný dokument je výsledek 1. Výsledek 0 je nevyhovující [3]. Například dokument *"Love and War is a 1996 romance drama film"* je relevantní pro dotazy *"Love AND War"* a *"Love OR Hate"*, není ale relevantní pro *"Love AND Hate"*.

Vnitřní reprezentací kolekce dokumentů je binární matice, kde řádky korespondují se slovy a sloupce s dokumenty. Prvky matice t_{ij} říkají, zda se slovo v dokumentu vyskytuje (1 pokud ano, 0 pokud ne):

$$d_i = (t_{i1}, t_{i2}, ..., t_{1m}), \forall t_{i,j} \in \{0,1\}$$

$$D = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nm} \end{pmatrix}$$

Tento model dělí dokumenty na vyhovující a nevyhovující zadanému dotazu, všechny vyhovující dokumenty se považují za stejně relevantní.

Vektorový model

Vektorový model rozšiřuje booleovský model o míru relevance a umožňuje tak seřadit odpověď dle míry relevance k dotazu. Prvky matice patří do množiny nezáporných čísel. Relevance dokumentu je z intervalu `<0;1>`. Existují způsoby vážení elementů matice frekvencí na základě jejich statistického rozložení mezi všemi dokumenty a v rámci jednoho dokumentu, např. TF-IDF (viz kapitola 2.6.2).

2.2 Sémantické prostory

Pojem sémantický prostor uvedl na svět Tannenbaum [4] a jeho hlavní myšlenkou je představa dokumentu jako bodu v prostoru; body, které jsou blízko sebe, jsou sémanticky podobné, body, které jsou daleko, jsou sémanticky vzdálené. Uživatelův dotaz je také bodem v prostoru a proto lze s ostatními dokumenty porovnávat co do vzdálenosti.

Sahlgren [5] definuje SP jako výpočetní model významu slova, který využívá distribučních rozložení slov spočtených přes velká textová data k reprezentaci sémantické podobnosti ve smyslu prostorové podobnosti.

SP je také možné chápat jako slovník slov příslušného slovního prostoru (kolekce dokumentů) [1], ve kterém je každému slovu přiřazen jeden vektor (řádka), sloupce mohou být kontexty (dokumenty, věty či jiné konstrukce), ve kterých se slova mohou vyskytovat, a jednotlivé prvky vektoru vypovídají o výskytu slova v daném kontextu. Všechny vektory jsou stejně dlouhé.

Existuje mnoho typů SP, všechny jsou založeny na distribuční hypotéze, která říká, že *slova vyskytující se v podobných kontextech jsou sémanticky podobná* [6]. Slova spolu sémanticky souvisejí, pokud mají jakýkoliv druh sémantického vztahu [7].

Míra podobnosti slov je definována vzdáleností jejich vektorů. Čím menší je vzdálenost vektorů, tím větší je sémantická podobnost daných slov. Tabulka 2-1 ukazuje, že významově blízká slova *cat* a *dog* nebo *doctor* a *death* mají vysoký koeficient podobnosti, a proto jsou sémanticky podobná.

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	
Cat	-0,54	0,85	0,27	-0,84	0,34	0,51	-0,32	0,28	0,99	0,59	⇨ cos(θ)=0,6381
Dog	-0,30	0,81	0,21	-0,32	0,40	0,55	-0,77	0,24	-0,31	0,42	
Doctor	0,32	0,29	0,56	0,39	-0,71	0,75	0,85	0,50	0,44	0,70	⇨ cos(θ)=0,9097
Death	0,37	0,58	0,39	0,63	-0,82	0,30	0,51	0,30	0,48	0,50	
Car	0,23	0,23	0,47	0,65	-0,83	0,60	-0,11	0,51	0,87	-0,25	⇨ cos(θ)=0,1825
Animal	-0,51	0,03	0,92	-0,79	0,92	0,82	-0,95	0,51	0,91	-0,07	

Tabulka 2-1: Příklad sémantického prostoru. Kosinový koeficient (viz kapitola 2.6.4) pro slova Doctor a Death je 0.9097, tzn., že spolu obě slova silně souvisí. Slova Car a Animal spolu nesouvisí.

Podobností vektorů lze zachytit různé typy podobností slov [2], jako jsou synonyma (*genocide, masacre*), meronyma (*screen* je součástí *notebooku*), antonyma (*hot, cold*) či hyperonyma (*hard drive, floppy drive* – sdílejí hardware). Podobnost může být *atributční* (*rat, mouse* – slova sdílejí některé vlastnosti) nebo *relační* (*bulb, lights* – mezi slovy je nějaký vztah) [2].

Mezi nejznámější algoritmy vytváření SP patří LSA[6], Coals[7], HAL[8], Beagle[9], RI[10]. Tyto algoritmy jsou založeny na statistice slov a jejich výskytů v kontextech. S tím souvisí statistická sémantická hypotéza, která říká, že *statistické modely používání slov člověkem nám mohou pomoci zjistit, co lidé danými slovy myslí* [2].

2.3 Sémantické prostory jako matice

SP jsou matice, kde řádky odpovídají *slovům* (resp. *termům*) a sloupce *kontextům*, ve kterých se slova vyskytují. Slovo je jednotkou přirozeného jazyka nesoucí význam. Term může být libovolný řetězec (obvykle slovo, ale i sousloví, věta apod.). V následujícím textu jsou pojmy slovo a term zaměňovány.

2.3.1 Matice typu Term-Document: Podobnost dokumentů

Řádky *Term-Document* matice (T-D matice) korespondují s jednotlivými unikátními termy, sloupcové vektory charakterizují dokumenty. Jednotlivé prvky matice vyjadřují četnost výskytu daného termu v dokumentu. Tento typ matice je vhodný pro porovnávání jednotlivých dokumentů nebo dokumentu s uživatelským dotazem, protože i ten může být za dokument považován.

[6] Latent Semantic Analysis
[7] Correlated Occurrence Analogue to Lexical Semantic
[8] Hyperspace Analogue to Language
[9] Bound Encoding of the Aggregate Language Enviroment
[10] Random Indexing

	d1	d2	d3	d4	d5	d6	d7	d8
some	0	2	0	1	1	0	0	0
people	1	0	2	0	1	0	1	0
make	0	2	0	3	1	1	0	0
world	1	0	0	2	1	3	0	1
special	0	1	0	0	1	0	0	0
just	1	3	5	2	1	1	2	0
being	0	0	1	0	1	0	0	0

Tabulka 2-2: Příklad T-D matice pro větu "Some people make the world special just by being in it".

2.3.2 Matice typu Word-Context: Podobnost slov

Konstrukce *Word-Context* matice (W-C matice) vychází z myšlenky rozdělení jednotlivých dokumentů na libovolnou velikost (slova[11], fráze, věty, kapitoly a jiné řetězce). Slova jsou reprezentovány vektory, jejichž prvky jsou odvozeny z jejich výskytů v různých kontextech.

	being	just	make	people	some	special	world
being	0	4	1	0	0	3	2
just	0	0	2	1	0	4	3
make	0	0	0	4	3	0	0
people	0	0	0	0	4	0	0
some	0	0	0	0	0	0	0
special	0	0	3	2	1	0	4
world	0	0	4	3	2	0	0

Tabulka 2-3: Předchozí příklad, tentokrát jako W-C matice.

Z tohoto typu matice lze určit atribuční podobnost mezi dvěma slovy. Hledá se podobnost mezi řádky matice. Podobné řádkové vektory znamenají podobný význam slov.

Podle [2] je T-D matice speciálním příkladem W-C matice. Přesněji by bylo možné říci, že obě tyto matice jsou speciálními příklady obecné *Term-Context* matice[12].

2.3.3 Matice typu Pair-Patern: Podobnost vztahů

Pair-Patern matice je využívána ke zjištění relační podobnosti mezi páry slov. Řádky korespondují s páry souvisejících slov (*doctor:heal*), sloupce vyjadřují předdefinované vztahy mezi těmito slovy (vzory). Prvky matice odpovídají výskytu zkoumaného páru v příslušném vzoru.

	X prepares Y	Y results from X	X is the opposite of Y	Y can happen to X	Y solves X	X leads to Y
butcher:meat	4	0	0	0	0	0
accident:death	0	2	0	0	0	3
love:war	0	1	12	0	1	1
black:white	0	0	7	0	0	0
bank:bankrupt	0	5	0	2	0	0
stringency:loan	0	2	0	0	3	2

Tabulka 2-4: Příklad Pair-Patern matice.

Vzory s podobnými sloupcovými vektory mají podobný význam (např. vzory *Y results from X* a *X leads to Y*). Pokud mají páry slov podobné řádkové vektory, zřejmě mají podobný sémantický vztah. Rozšířená distribuční hypotéza říká, že *vzory, vyskytující se v podobných párech slov, mají podobný význam* [2].

---

[11] *Word-by-Word* matice

[12] Slovo je speciálním případem termu, dokument speciálním případem kontextu

2.4 Jazykové předzpracování korpusu

Korpusem dat je míněna kolekce textů v přirozeném jazyce, který slouží pro lingvistický výzkum v daném jazyce a nebo například jako základ pro tvorbu slovníků, ontologií, překladačů apod. Pro účely této práce byl použit korpus iniciativy CLEF v anglickém jazyce[13].

Korpus slouží jako základ pro vyhledávání a vytvoření SP (obrázek 4-1). Text je vhodné jazykově předzpracovat [2]. Nejprve je nutné rozdělit text na jednotlivá slova (odstavec 2.4.1), pak text normalizovat[14] (odstavec 2.4.2) a případně rozlišit a označit významy víceznačných slov (odstavec 2.4.3).

Na obrázku 2-2 jsou vidět rozdíly mezi jednotlivými typy jazykového zpracování textu, které jsou popsány v následujících oddílech.

tokenizace:	Last week I saw a woman flayed, and you will hardly believe how much it altered her appearance for the worse.
stopslova:	Last week I saw woman flayed, you hardly believe how much altered her appearance worse.
case folding:	last week i saw woman flayed you hardly believe how much altered her appearance worse
stemming:	last week i saw woman flai you hardli believ how much alter her appear wors
lematizace:	last week see woman flay hardly believe much alter she appearance worse

Obrázek 2-2: Příklady různého jazykového zpracování věty: "Last week I saw a woman flayed, and you will hardly believe how much it altered her appearance for the worse". Tyto metody jsou popsány v následujících oddílech. Zajímavý je příklad stemmingu, který „y“ v koncovce změkčuje.

2.4.1 Tokenizace

Tokenizace je základním způsobem předzpracování textu, rozděluje řetězec (dotaz, dokument) na *tokeny*. Mezi základní techniky patří oddělení tokenů pomocí mezer (resp. bílých znaků). V průběhu tokenizace se dále mohou odstranit nevýznamová slova (*stopslova)*[15] a převést slova na malá písmena (*case folding*)[16].

Pokročilejší algoritmy tokenizace dokáží rozpoznat víceslovné termy (např. *ice hockey*), složeniny oddělené pomlčkou (*State-of-Art*) a další složeniny, jako například *N/A, don't*, *Petr's*.

2.4.2 Normalizace (stemming a lemmatizace)

Jednou z forem normalizace textu je stemming. Stemming ořezává předpony, přípony a koncovky s cílem převést slovo na jeho kořen. Pracuje s tabulkou předpon, přípon a koncovek, a s pravidly pro jejich ořezávání. Například slova *argue, arguing, argued* převede na a*rgu*, nevýhodou je, že na *argu* převede i slovo *Argus*. Další slabostí stemmingu jsou slova, která stupňováním mění kořen (*best* nepřevede na *good*).

---

[13] Anotovaná česká data nejsou dostupná.

[14] Ve smyslu zobecnění, tedy zjednodušení textu (více slovních tvarů se převádí na jeden společný).

[15] Nevýznamová slova nebo slova s vysokým výskytem (and, or, is, it atd.).

[16] Převod na malá písmena patří mezi metody normalizace [2].

Další formou normalizace textu je Lemmatizace[17]. Lemmatizace je založená na slovníku slov a jejich základních tvarů (tzv. *lemmat*). Slova *arguing, argued* převede na a*rgue*. Odstraňuje nevýhody stemmingu (*best, good*), neporadí si ale s víceznačnými slovy (*lead* může být sloveso nebo podstatné jméno; *bow* je luk, příď nebo poklona). Tento problém řeší anotace. Další nevýhodou lemmatizace je skutečnost, že pracuje se slovníkem. Slovník ale nemůže být dokonalý, protože se jazyk neustále vyvíjí a vznikají nová slova.

Normalizace zobecňuje tvary slov, tím zvyšuje úplnost (sloučení různých tvarů slov vede ke snažšímu vyhledání slova v textu, je tedy nalezeno více dokumentů) na úkor přesnosti (některé slovní tvary mají sémantický význam a jejich odstranění vede k nalezení nerelevantních dokumentů) [2].

2.4.3 Anotace

Anotace rozlišuje víceznačná slova (*lead/vb* – sloveso, *lead/nn* – podstatné jméno). Slova značkuje podle jejich výskytu v kontextu, podle smyslu, který vyjadřují, nebo podle gramatické role (tj. provádí gramatickou analýzu textu). Podrobněji ve [2].

Anotace zvyšuje spíše přesnost odpovědi (označíme-li slovo *program* jako podstatné jméno, můžeme vyhledávat dokumenty související s jednotlivými počítačovými programy, možná ale vyloučíme některé dokumenty související se samotným aktem programování), ale snižuje úplnost (dokumenty související s programováním mají co říct i o programech) [2].

2.5 Matematické zpracování korpusu

V následujících kapitolách jsou popsány jednotlivé kroky matematického zpracování matice, které vedou k vytvoření SP. Vygenerování matice frekvencí, převážení jednotlivých elementů matice a její redukce. V závěru kapitoly jsou popsány principy algoritmů LSA, HAL a Coals.

2.5.1 Matice frekvencí výskytů

Matice jsou matematickým nástrojem, který umožňuje reprezentovat SP (kapitola 2.3); například algoritmus LSA pracuje s T-D maticí, Coals a HAL s W-C maticí. Prvky matice obecně korespondují s termy vyskytujícími se v různých kontextech.

2.5.2 Vážení elementů

Princip vážení elementů vychází ze skutečnosti, že vzácná (málo frekventovaná) slova mají vyšší informační hodnotu. Účelem vážení je přidat vzácným slovům váhu a ubrat ji běžným slovům. Mezi nejpoužívanější způsoby vážení patří TF-IDF[18], PMI[19] a jeho variace PPMI[20].

Princip TF-IDF

Algoritmus TF-IDF je způsob vážení matice frekvencí výskytů.

---

[17] http://en.wikipedia.org/wiki/Lemmatisation

[18] Term Frequency - Inverse Document Frequency

[19] Pointwise Mutual Information

[20] Positive PMI

Složka **tf** (*term frequency*) vyjadřuje četnost slova v příslušném dokumentu. Normalizuje se vydělením délkou dokumentu (součtem počtu výskytů všech slov v dokumentu d_j), aby se předešlo nadhodnocení dlouhých dokumentů:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{m} \mathrm{n}_{\mathrm{k,j}}},$$

kde $n_{i,j}$ je četnost slova t_i v dokumentu d_j a m je počet slov dokumentu d_j.

Složka **idf** (*inverse document frequency*) vyjadřuje inverzní četnost slova ve všech dokumentech (čím častěji se slovo v ostatních dokumentech vyskytuje, tím méně je vzácné, resp. důležité):

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|},$$

kde v čitateli je počet dokumentů korpusu, ve jmenovateli dokumenty, ve kterých se slovo t_i vyskytuje.

Hodnota výrazu **tfidf** roste, když je slovo časté v příslušném dokumentu a málo časté ve všech ostatních dokumentech korpusu, tedy *tfidf* = *td* x *idf*.

2.5.3 Redukce dimenze matic

Matice frekvencí jsou rozsáhlé a řídké, obsahují mnohem více nulových hodnot než nenulových. Mezi nejpoužívanější algoritmy, které matici redukují ve smyslu snížení její dimenze, patří SVD[21], který lze aplikovat na W-C i T-D matice.

Singulární rozklad matice SVD

Singulární rozklad matice **A** je definován vztahem

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}},$$

Matice je rozložena na součin tří matic, kde **U** a **V** jsou matice reálných čísel (sloupce jsou ortogonální[22]), tj. $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{1}$[23], a **S** je diagonální matice sestupně seřazených nezáporných singulárních čísel (obrázek 2-3).

Má-li **A** hodnost *r*, platí, že **S** je také řádu *r* a pro singulární čísla σ_i matice **S** platí vztah:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq \sigma_{r+1} = \ldots = \sigma_n = 0$$

Korpus je takto rozložen na *r* lineárně nezávislých bázových vektorů[24], nazývaných koncepty. Každý term a kontext je indexován konceptem. Koncepty lze chápat jako jednotlivé kontexty korpusu. Důležitost konceptů vyjadřují singulární čísla matice **S**. Tato vlastnost SVD umožňuje termy sémanticky shlukovat [8].

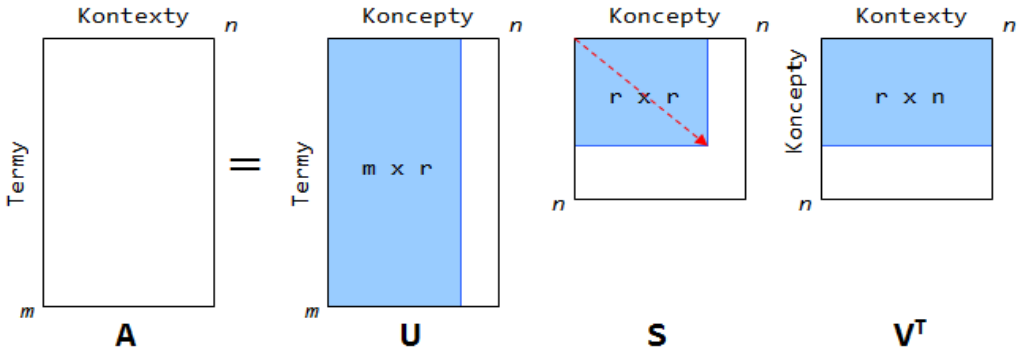
---

[21] Singular value decomposition

[22] Jejich skalární součin je nula.

[23] Jednotková matice - čtvercová matice, která má na diagonále jedničky a nuly na ostatních místech.

[24] Bázové vektory jsou de facto osy, které definují příslušný vektorový prostor.



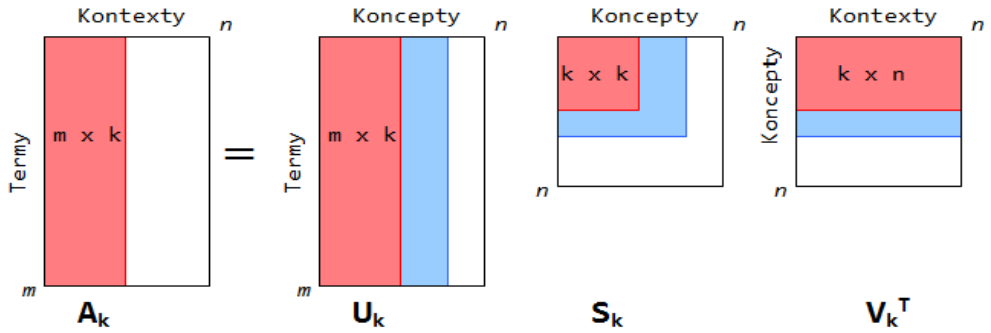
Obrázek 2-3: SVD rozklad matice A. Červená šipka vyjadřuje seřazení čísel na diagonále podle velikosti. Převzato z [8].

Vzhledem k tomu, že singulární čísla jsou seřazena sestupně podle důležitosti konceptů, za sémanticky důležité lze považovat pouze prvních k konceptů [8]. Původní rozklad lze aproximovat

$$\mathbf{A_k} = \mathbf{U_k S_k V_k}^T,$$

kde matice $\mathbf{A_k}$ je podmaticí $\mathbf{A}$ obsahující prvních k sloupců, $\mathbf{S_k}$ (kde $k < r$) je diagonální matice k nejvyšších hodnot matice $\mathbf{S}$ (čtvercová podmatice), matice $\mathbf{U_k}$ a $\mathbf{V_k}$ jsou vytvořené výběrem příslušných k sloupců z matic $\mathbf{U}$ a $\mathbf{V}$.

Této operaci se říká k-aproximace nebo redukce prostoru[25]. Redukovaná matice $\mathbf{A_k}$ nejlépe aproximuje matici $\mathbf{A}$ (obrázek 2-4). Tato matice je výsledným sémantickým prostorem pro slovní tvary korpusu.



Obrázek 2-4: Redukce prostoru.

Ilustrační příklad, který znázorňuje vztah mezi termy a koncepty lze nalézt v [8].

Podle [2] lze SVD chápat jako nástroj pro odhalování skrytých významů[26], pro nacházení spoluvýskytů vyšších řádů[27] nebo pro zhuštění matice[28].

2.5.4 Porovnávání vektorů slov a dokumentů

Porovnávání dotazu s dokumentem (resp. slova se slovem) je založeno na chápání těchto subjektů jako bodů v n-rozměrném prostoru dokumentů (resp. slov), ve kterém jsou

[25] Tato operace pomáhá odkrýt skryté vazby mezi termy a kontexty [8].

[26] Redukce dimenze původní matice vede k vyšší souvislosti mezi slovy a kontexty (viz obrázek 2-6).

[27] Slova, která se spolu přímo nevyskytují, ale vyskytují se se stejnými slovy, spolu souvisí.

[28] Matice A je řídká, A_k je hustá, tj. kompenzuje chybějící data.

porovnávané subjekty vyjádřeny frekvenčními vektory velikosti *n*. Vzdálenost vektorů vyjadřuje podobnost subjektů.

Kosinová (úhlová) vzdálenost

Kosinovou vzdálenost lze interpretovat jako úhel mezi dvěmi frekvenčními vektory **p** a **q**. Tento úhel vyjadřuje jejich podobnost. Kosinovou vzdálenost lze vyjádřit vztahem

$$\cos(\vec{p},\vec{q}) = \frac{\sum_{i=1}^{n} p_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} (p_i)^2 \cdot \sum_{i=1}^{n} (q_i)^2}}$$

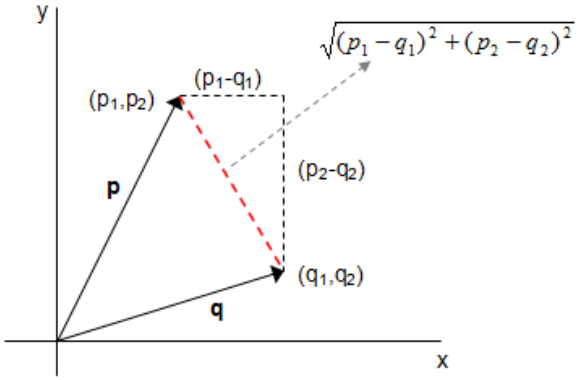
Výsledek leží na intervalu `<-1;1>`. Čím menší kosinová vzdálenost, tím menší podobnost. Hodnota -1 (180°) vyjadřuje opak významu, 0 (90°) nezávislost, 1 (0°) přesnou shodu.

Eukleidovská vzdálenost

Eukleidovská vzdálenost vyjadřuje vzdálenost mezi dvěma body prostoru. Vzdálenost je dána Pythagorovou větou (obrázek 2-5)[29]. Eukleidovská vzdálenost je popsána vztahem

$$e(\vec{p},\vec{q}) = \frac{1}{1+\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}},$$

kde **p** a **q** jsou vektory o stejném počtu prvků.



Obrázek 2-5: Výpočet eukleidovské vzdálenosti dvou bodů v prostoru.

Vzdálenost 0 znamená přesnou shodu, s rostoucí vzdáleností podobnost slov (resp. dokumentů) klesá.

2.5.5 Příklady algoritmů vytváření SP

Tato kapitola vychází z [1] a [9] a popisuje základní principy algoritmů LSA, HAL a Coals. Hlubší pochopení jednotlivých algoritmů není součástí této práce.

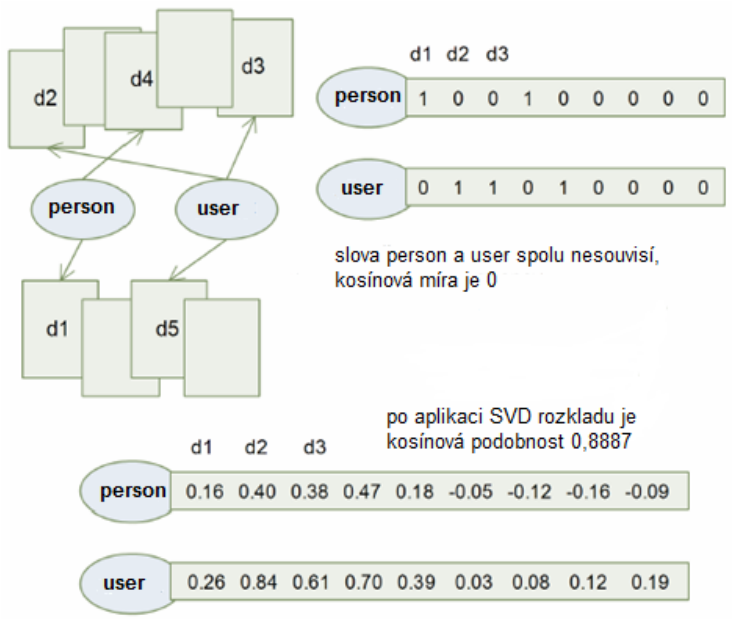
Algoritmus LSA

Algoritmus LSA vychází z vektorového modelu, který redukuje pomocí SVD [10]. Nejprve je vytvořena T-D matice spoluvýskytů (řádky jsou termy, sloupce dokumenty, slova nejsou uspořádána), ta je poté znormalizována algoritmem TF-IDF a zredukována algoritmem SVD (dimenze *k* je standardně nastavena na hodnotu 300, která se ukazuje jako optimální) [9].

[29] http://en.wikipedia.org/wiki/Euclidean_distance

SVD rozklad umožňuje nalézt další vztahy mezi slovy (viz odstavec 2.5.3). Například slova *football* a *sport* se spolu budou vyskytovat častěji než *football* a *history*. Slova *football* a *tennis* se v jednom dokumentu vyskytovat společně nemusí, obě se ale budou vyskytovat společně se slovem *sport*. Tomuto se říká spoluvýskyt druhé úrovně. Takových úrovní může být více. Čím vyšší úroveň spoluvýskytu, tím menší má tato souvislost význam.

Obrázek 2-6 znázorňuje vytvoření vektorů pro slova *person* a *user*. Společně se nevyskytují ani v jednom dokumentu, ale po aplikaci SVD rozkladu a snížení dimenze jsou si vzniklé vektory podobné. Vyskytují se v podobných kontextech.



Obrázek 2-6: Princip algoritmu LSA. Obrázek znázorňuje vytvoření vektorů pro slova person a user. Společně se nevyskytují ani v jednom dokumentu, ale po aplikaci SVD rozkladu a snížení dimenze jsou si vzniklé vektory podobné. Vyskytují se v podobných kontextech. Převzato z [1].

Algoritmus HAL

Myšlenkou algoritmu HAL je definice kontextu jako oblasti okolo slova (této oblasti se říká okénko). Vstupním parametrem HAL je velikost okénka. Narozdíl od LSA, algoritmus HAL pracuje s W-W maticí[30].

Síla spoluvýskytu (součinnost) dvou slov je nepřímo úměrná počtu slov, které je v rámci okna oddělují [11]. Například pro větu *"Don't frown. You never know who is falling in love with your smile"* lze z okénka velikosti 5 vyčíst, že slova *falling* a *love* mají silnou součinnost, naopak *know* a *smile* mají slabou nebo žádnou součinnost.

Výstupem algoritmu HAL je matice spoluvýskytů slov, kde řádky i sloupce jsou slova kontextu a každá položka matice reprezentuje souvislost mezi dvěma slovy (vážený spoluvýskyt slov v okénku). Každé slovo matice (řádek) je popsáno hodnotami, které slovu předcházejí i následují, proto se každé slovo vyskytuje ve sloupci dvakrát (tabulka 2-5).

[30] Word-by-word matice je speciálním případem W-C matice.

	a	as	chuck	could	how	if	much	wood	woodch	would	,	.	?	a	as	chuck	could	how	if	much	wood	woodch	would	,	.	?
a	13	24	12	3	9	20	22	31	16	23	18	0	7	13	7	31	26	0	14	4	21	50	9	16	7	7
as	7	8	15	11	0	5	9	25	10	0	3	0	17	24	8	2	3	0	9	10	10	20	13	11	0	0
chuck	31	2	5	20	5	14	6	9	36	15	12	0	0	12	15	5	6	0	9	8	30	10	2	11	9	12
could	26	3	6	0	0	16	2	4	30	9	14	0	0	3	11	20	0	0	0	6	23	2	1	0	8	8
how	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	5	0	0	3	10	9	7	8	4	0	0
if	14	9	9	0	3	0	8	11	16	15	20	0	2	20	5	14	16	0	0	3	14	18	0	0	5	5
much	4	10	8	6	10	3	0	8	5	0	2	0	9	22	9	6	2	0	8	0	20	18	15	10	0	0
wood	21	10	30	23	9	14	20	7	26	5	11	0	8	31	25	9	4	0	11	8	7	26	20	14	10	10
woodch.	50	20	10	2	7	18	18	26	13	20	16	0	5	16	10	36	30	0	16	5	26	13	10	18	9	9
would	9	13	2	1	8	0	15	20	10	0	0	0	4	23	0	15	9	0	15	0	5	20	0	17	3	0
,	16	11	11	0	4	0	10	14	18	17	0	0	3	18	3	12	14	0	20	2	11	16	0	0	4	4
.	7	0	9	8	0	5	0	10	9	3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
?	7	0	12	8	0	5	0	10	9	0	4	0	0	7	17	0	0	0	2	9	8	5	4	3	0	0

Tabulka 2-5: Příklad matice vytvořené algoritmem HAL pro "How much wood would a woodchuck chuck, if a woodchuck could chuck wood? As much wood as a woodchuck would, if a woodchuck could chuck wood.". Obrázek je převzat z [9].

Velikost okna v tabulce 2-5 je 10. Prvky matice obsahují vážený součet všech výskytů slova ve sloupci v blízkosti slova v řádku (blízkost je určena velikostí okna). Například v řádce *would* má první *woodchuck* hodnotu 10[31], druhý 20[32].

Algoritmus Coals

Coals vychází z HAL, ale snaží se snížit nežádoucí vliv sousedících slov s vysokou četností[33]. Nejdříve je vytvořena W-W matice (pro každé slovo jeden sloupec). Stopslova jsou ignorována a velikost okénka je 4 (pokud slovo *b* sousedí s *a*, potom má příslušný prvek matice hodnotu 4, pokud je ve vzdálenosti dvou slov, má hodnotu 3, atd.), protože se ukazuje jako dostačující a redukuje velikost výsledného SP (tabulka 2-6). Je zachován jen stanovený počet *Open-Class*[34] slov, ostatní sloupce jsou zahozeny.

	a	as	chuck	could	how	if	much	wood	woodch.	would	.	.	?
a	0	5	9	6	1	10	4	8	18	9	10	0	0
as	5	4	2	1	0	0	7	10	3	2	1	0	5
chuck	9	2	0	8	0	5	1	9	11	2	4	3	3
could	6	1	8	0	0	4	0	6	8	0	2	2	2
how	1	0	0	0	0	0	4	3	0	2	0	0	0
if	10	0	5	4	0	0	0	0	10	3	8	0	0
much	4	7	1	0	4	0	0	10	2	3	0	0	3
wood	8	10	9	6	3	0	10	2	8	5	0	4	6
woodch.	18	3	11	8	0	10	2	8	0	8	10	1	1
would	9	2	2	0	2	3	3	5	8	0	5	0	0
,	10	1	4	2	0	8	0	0	10	5	0	0	0
.	0	0	3	2	0	0	0	4	1	0	0	0	0
?	0	5	3	2	0	0	3	6	1	0	0	0	0

Tabulka 2-6: Vytvoření matice, velikost okna je 4. Převzato z [9].

Výskyty slov jsou převedeny na Pearsonovu korelaci[35] mezi jednotlivými slovy. Prvky matice jsou z intervalu `<-1;1>` (tabulka 2-7), kde korelační koeficient vyjadřuje, do jaké míry se slova vyskytují společně.

[31] Vyskytuje se jednou přímo před *would.*

[32] Vyskytuje se dvě slova za prvním *would* (9 bodů), sedm slov za (4 body) a čtyři slova za druhým *would* (7 bodů).

[33] Algoritmus je blíže popsán v [9].

[34] Významová slova (podstatná jména, slovesa, přídavná jména apod.).

[35] Podrobněji na http://cs.wikipedia.org/wiki/Korelace.

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	-0.167	-0.014	0.014	0.009	-0.017	0.085	-0.018	-0.033	0.096	0.069	0.085	-0.055	-0.079
as	-0.014	0.031	-0.048	-0.049	-0.037	-0.077	0.133	0.103	-0.054	-0.021	-0.050	-0.037	0.133
chuck	0.014	-0.048	-0.113	0.094	-0.045	0.021	-0.061	0.031	0.048	-0.046	-0.002	0.088	0.031
could	0.009	-0.049	0.094	-0.075	-0.037	0.033	-0.070	0.022	0.049	-0.075	-0.021	0.069	0.023
how	-0.017	-0.037	-0.045	-0.037	-0.018	-0.037	0.192	0.070	-0.055	0.069	-0.037	-0.018	-0.026
if	0.085	-0.077	0.021	0.033	-0.037	-0.077	-0.071	-0.106	0.085	0.006	0.138	-0.037	-0.053
much	-0.018	0.133	-0.061	-0.070	0.192	-0.071	-0.065	0.128	-0.061	0.019	-0.071	-0.034	0.072
wood	-0.033	0.103	0.031	0.022	0.070	-0.106	0.128	-0.113	-0.033	0.001	-0.106	0.111	0.100
woodch.	0.096	-0.054	0.048	0.049	-0.055	0.085	-0.061	-0.033	-0.167	0.049	0.085	-0.017	-0.051
would	0.069	-0.021	-0.046	-0.075	0.069	0.006	0.019	0.001	0.049	-0.075	0.060	-0.037	-0.053
,	0.085	-0.050	-0.002	-0.021	-0.037	0.138	-0.071	-0.106	0.085	0.060	-0.077	-0.037	-0.053
.	-0.055	-0.037	0.088	0.069	-0.018	-0.037	-0.034	0.111	-0.017	-0.037	-0.037	-0.018	-0.026
?	-0.079	0.133	0.031	0.023	-0.026	-0.053	0.072	0.100	-0.051	-0.053	-0.053	-0.026	-0.037

Tabulka 2-7: Četnosti jsou převedeny na korelační koeficienty. Převzato z [9].

V dalším kroku se záporná čísla vynulují (to, že se slova spolu nevyskytují, nepřidává další informaci) a kladná odmocní (tabulka 2-8) [9].

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	0	0	0.120	0.093	0	0.291	0	0	0.310	0.262	0.291	0	0
as	0	0.175	0	0	0	0	0.364	0.320	0	0	0	0	0.365
chuck	0.120	0	0	0.306	0	0.146	0	0.177	0.220	0	0	0.297	0.175
could	0.093	0	0.306	0	0	0.182	0	0.149	0.221	0	0	0.263	0.151
how	0	0	0	0	0	0	0.438	0.265	0	0.263	0	0	0
if	0.291	0	0.146	0.182	0	0	0	0	0.291	0.076	0.372	0	0
much	0	0.364	0	0	0.438	0	0	0.358	0	0.136	0	0	0.268
wood	0	0.320	0.177	0.149	0.265	0	0.358	0	0	0.034	0	0.333	0.317
woodch.	0.310	0	0.220	0.221	0	0.291	0	0	0	0.221	0.291	0	0
would	0.262	0	0	0	0.263	0.076	0.136	0.034	0.221	0	0.246	0	0
,	0.291	0	0	0	0	0.372	0	0	0.291	0.246	0	0	0
.	0	0	0.297	0.263	0	0	0	0.333	0	0	0	0	0
?	0	0.365	0.175	0.151	0	0	0.268	0.317	0	0	0	0	0

Tabulka 2-8: Záporné hodnoty jsou vynulovány, kladné odmocněny. Převzato z [9].

Sémantická podobnost dvou slov je dána úhlem mezi jejich vektory, jako v předchozích příkladech lze použít například kosinovou vzdálenost.

2.6 Úvod do rozšiřování dotazu

Rozšiřování dotazu je proces přeformulování dotazu, za účelem zvýšení kvality (přesnosti a úplnosti) odpovědi vyhledávacího systému. Přeformulování dotazu probíhá formou jeho rozšíření o slova se stejným nebo podobným významem (nebo slova, která se slovy původního dotazu souvisí).

Například rozšíření dotazu *„'Eddie the Eagle' +won competition“* by mohlo vypadat následovně (mezi jednotlivými slovy je logické nebo):

„Eddie Eagle won competition Edward Edwards bird champion rivalry match winner victory“

Následující oddíl čerpá z [1] a popisuje některé techniky rozšiřování dotazu.

Techniky pro rozšiřování dotazu

Jedním ze způsobů rozšiřování dotazu je využití zpětné vazby, která spočívá ve výběru slov pouze z dokumentů, které jsou součástí odpovědi na původní dotaz [1]. Dotaz je rozšířen

o slova, která se v těchto dokumentech vyskytují častěji, než v ostatních dokumentech, a zároveň nejsou stopslovy.

Dalším způsobem rozšiřování dotazu je využití manuálně vytvořených sítí slov[36] nebo slovníků, ze kterých lze podobná slova vybírat.

Dále je možné využít SP, které vztahy mezi slovy vytváří automaticky a tím umožňují zautomatizování celého procesu rozšiřování dotazu. Slova se k dotazu přidávají na základě jejich sémantické podobnosti se slovy původního dotazu. Takto přidaná slova ale nemusí ve stejném kontextu dávat stejný smysl (nejedná se vždy o synonyma). Tento problém lze řešit filtrací slov [1]. Filtrace slov může být založena na četnosti jejich výskytů v korpusu (tzn. na jejich vzácnosti). Jiný způsob filtrace slov spočívá v rozšíření původního dotazu o slova, která souvisejí s více slovy původního dotazu (např. pro dotaz *„Hot Dog“* by slova *„warm“*, *„cat“* nebo *„pet“* nebyla žádoucí).

[36] Např. WordNet, lexikální databáze slov pro anglický jazyk. Viz http://en.wikipedia.org/wiki/WordNet

3 Analýza

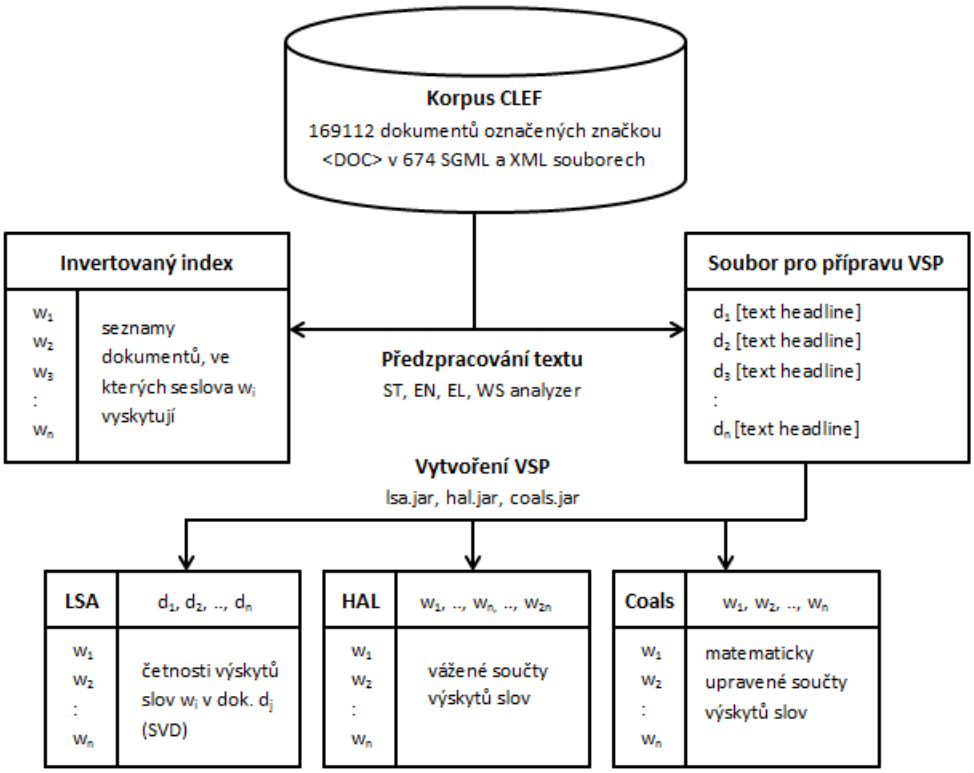
Kapitola se věnuje řešeným problémům, použitým prostředkům a algoritmům k sestavení vyhledávacího systému. Zabývá se korpusem dat, strukturou souborů a jejich převedením do validního formátu a nástrojem TREC, který slouží k vyhodnocení kvality odpovědi na vyhledávací dotaz. V další části je popsáno vytváření SP pomocí knihovny S-Space. Dále se zabývá problematikou rozšíření dotazu a základními funkcemi aplikace.

3.1 Analýza korpusu CLEF

Korpusem dat je míněna kolekce souborů obsahující články, které vyšly v denících Los Angeles Times 1994 (dále LA94) a Glasgow Heralds 1995 (dále GH95) v anglickém jazyce.

Články LA94 jsou uloženy v XML[37] a GH95 v SGML[38] souborech. Oba typy mají podobnou strukturu, v detailech se ale liší. Ani jeden z nich nedodržuje předepsanou strukturu validních souborů ve značkovacím jazyce.

Korpus dat obsahuje manuálně ohodnocená[39] textová data a nástroj pro vyhodnocení relevance nalezených článků na předdefinované dotazy. Obrázek 3-1 znázorňuje využití korpusu k vytvoření indexu a SP.



Obrázek 3-1: Využití korpusu k vytvoření indexu a různých typů SP. Analyzátory textu ES, EN, EL a WS jsou popsány v oddílu 3.2.

[37] Extensible Markup Language (rozšiřitelný značkovací jazyk).

[38] Standard Generalized Markup Language – ISO standard umožňující definovat značkovací jazyky.

[39] Ohodnocená data ve smyslu míry relevance článků korpusu k předdefinovaným dotazům.

3.1.1 XML soubory

Každý soubor obsahuje články celého vydání[40]. Články jsou ohraničeny značkou `<DOC>` a mají své vlastní unikátní číslo `<DOCNO>`. V následující tabulce je uveden popis značek, které se v dokumentu mohou vyskytovat:

`<DOC>`	Ohraničuje článek (a všechny následující značky).
`<DOCNO>`	Unikátní číslo článku.
`<DOCID>`	Unikátní číslo článku v rámci příslušného vydání.
`<SOURCE>`	Zdroj (noviny, ve kterých článek vyšel).
`<DATE>`	Datum, např. *March 13, 1994, Sunday.*
`<SECTION>`	Sekce, do které spadá, např. *Sport.*
`<LENGTH>`	Počet slov článku.
`<HEADLINE>`	Titulek článku.
`<BYLINE>`	Podtitulek, tj. autor a jeho funkce.
`<TEXT>`	Vlastní text článku.
`<TYPE>`	Další upřesňující informace, např. *fejeton.*
`<P>`	Odstavec.

Znaky `&, <, >` nejsou převedeny na escape sekvence[41] a chybí kořenový element.
Článek může vypadat například takto:

```
<DOC>
  <DOCNO> LA010194-0001 </DOCNO>
  <DOCID> 000013 </DOCID>
  <SOURCE>
    <P>Los Angeles Times</P>
  </SOURCE>
  <DATE>
    <P>January 1, 1994, Saturday, Orange County Edition </P>
  </DATE>
  <SECTION>
    <P>Part A; Page 1; Column 1; Metro Desk </P>
  </SECTION>
  <LENGTH>
    <P>330 words </P>
  </LENGTH>
  <HEADLINE>
    <P>ORANGE COUNTY NEWSWATCH </P>
  </HEADLINE>
  <BYLINE>
    <P>By Jerry Hick & Greg Hernandez; Nettie Mackley </P>
  </BYLINE>
  <TEXT>
    <P>
    BUCKLE UP: The new bicycle helmet law for youngsters takes effect today.
    But that doesn't mean your child will be cited for forgetting. "We're
    going to educate the public first," says Fountain Valley police Lt.
    Robert Mosley. "We're going to have a six-month grace period where we'll
    just be issuing warnings." . . . Some local police departments will be
    spreading the word through flyers and programs at schools. But Mosley
```

[40] Jedná se o deníky, soubor tedy obsahuje všechny články jednoho denního vydání.

[41] Escape sekvence dává kompilátoru na vědomí, že se jedná o součást řetězce a ne o funkční znak.

```
    warns: "Kids need to follow the law."
    </P>
  </TEXT>
  <TYPE>
    <P>Column; Brief </P>
  </TYPE>
</DOC>
```

Ve značce <BYLINE> je vidět znak &. Navíc se v dokumentu často vyskytuje značka <P>, která se v textu opakuje a nelze předem říci kolikrát.

3.1.2 SGML soubory

Každý soubor obsahuje články příslušného vydání, které jsou ohraničeny značkou <DOC> a mají své vlastní unikátní číslo <DOCNO>:

<DOC>	Ohraničuje článek.
<DOCNO>	Unikátní číslo článku.
<DOCID>	Unikátní číslo článku, stejné jako <DOCNO>.
<DATE>	Datum, např. *060907*.
<HEADLINE>	Titulek článku.
<BYLINE>	Podtitulek, tj. autor a jeho funkce.
<EDITION>	Číslo edice.
<PAGE>	Stránka, na které článek začíná.
<ARTICLETYPE>	Typ článku, např. *medailonek*.
<GRAPHIC>	Typ obrázku, pokud se ve článku vyskytuje.
<TEXT>	Vlastní text článku.

Znaky &, <, > jsou převedeny na escape sekvence, ale také chybí kořenový element.

3.1.3 Unifikace souborů

Pro účely této práce stačí pracovat se dvěma typy značek. S těmi, co obsahují textové informace (<TEXT>, <HEADLINE>), a s těmi, které jednoznačně identifikují článek, aby ho bylo možno zařadit do vyhodnocení (<DOC>, <DOCNO>). Tyto značky jsou v obou typech dokumentů zastoupeny a navíc vždy řádně uzavřeny.

Soubory jsou převedeny do validního stavu tak, aby si s nimi poradil kompilátor Java, resp. knihovna Apache Commons Digester[42]:

- přidá se kořenová značka <DOCUMENT> na začátek a konec souboru,
- odstraní se všechny značky <P> z XML dokumentů, protože nemají informační hodnotu,
- znaky &, <, > se převedou na escape sekvence, např. &
- přípona všech souborů se změní na XML.

Předzpracování souborů je implementováno do aplikace.

[42] Tato knihovna umožňuje práci s xml soubory, http://commons.apache.org/proper/commons-digester/.

3.1.4 Vyhodnocení relevance (aplikace TREC)

TREC je součástí korpusu a slouží pro vyhodnocení odpovědi na zadaný dotaz. Korpus dále obsahuje:

- sadu číselně označených *trénovacích*[43] vyhledávacích témat (č. 201 - 250),
- sadu číselně označených *testovacích*[44] vyhledávacích témat (č. 251 - 300),
- soubor (zvláštní pro každou sadu), který obsahuje seznam všech relevantních dokumentů k danému tématu.

Tato témata i seznam relevantních dokumentů byla vytvořena manuálně. Témata jsou zapsána v XML a definována následovně:

```
<top>
<num> C259 </num>
<EN-title> Golden Bear </EN-title>
<EN-desc> Which movies have been awarded the Golden Bear… </EN-desc>
<EN-narr> Any document naming a film that has won the Golden Bear is
relevant. </EN-narr>
</top>
```

Dotaz je ohraničen značkou `<EN-title>`, zbylé značky obsahují bližší informace o tématu a vysvětlení, jaký dokument je k danému tématu relevantní.

Vstupní soubory

Seznam relevantních dokumentů k jednotlivým tématům (1) je uložen ve formátu:

```
query_id iteration document_number relevant,
```

kde `query_id` značí číslo tématu, `iteration` se nepoužívá, `document_number` unikátní číslo dokumentu, ve kterém se téma vyskytuje, a `relevant` informaci, zda je tento dokument relevantní.

query_id	iteration	document_number	relevant
201	0	GH951221-000011	0
201	0	GH951221-000092	1
201	0	GH951227-000056	0
202	0	GH951228-000122	1
202	0	LA010294-0203	0
202	0	LA010294-0340	1

Dalším vstupem pro aplikaci, je seznam dokumentů, které vyhledávač na dotaz nalezl (2). Soubor musí být z aplikace vyhledávače vyexportován v následujícím formátu:

```
query_id iteration document_number rank similarity run_id,
```

kde údaj `similarity` vyjadřuje skóre, kterým byl dokument v rámci odpovědi na dotaz vyhledávačem ohodnocen (skóre se používá například pro vyhodnocení střední průměrné přesnosti – viz odstavec Výstupní statistiky). Pole `iteration`, `rank`, `run_ID` jsou ignorována.

[43] Nad těmito tématy budou prováděny všechny experimenty.

[44] Na těchto tématech budou ověřeny výsledky experimentů provedených na trénovacích tématech.

```
query_id   iteration   document_number   rank   similarity   run_id
201        0           LA012994-0002     0      1,504258     0
201        0           LA100194-0244     0      1,247906     0
201        0           GH950105-000035   0      1,209812     0
201        0           LA071494-0343     0      1,074971     0
```

Jednotlivé údaje jsou odděleny mezerou, řetězce mezeru obsahovat nesmí, ignorované položky jsou nastaveny na hodnotu 0 (prázdné nelze vynechat).

Tyto dva soubory slouží jako zdroje, ze kterých bude vygenerován výstup.

Výstupní statistiky

Aplikace TREC poskytuje metriky pro vyhodnocení přesnosti a úplnosti odpovědi. Výstup do příkazové řádky lze spustit příkazem:

```
trec.exe [-q] q_rels topic_eval,
```

kde první parametr je soubor s vyhodnocenými dotazy (1) a druhý je soubor se seznamem dokumentů odpovědi (2). Mimo jiné lze přidat parametr `-q` pro rozdělení výstupního vyhodnocení podle jednotlivých témat (jinak vyhodnotí vše dohromady).

Výstup do příkazové řádky je ve formátu:

```
measure_name query value,
```

kde `measure_name` značí míru, `query` téma (číslo tématu nebo všechna témata) a `value` naměřenou hodnotu. Jednotlivé hlavní míry jsou vysvětleny v následující tabulce:

Počty dokumentů `num_q, num_ret, num_rel, num_rel_ret`	Q – počet vyhodnocených témat, Retrieved – počet nalezených dokumentů, Relevant – celkový počet relevantních dokumentů k tématu, Relevant retrieved – počet nalezených relevantních dokumentů.
Úrovně úplnosti (*Interpolated Recall Precision Averages*) `ircl_prn.0.xx`	Přesnost na různých úrovních úplnosti. Např. přesnost 0.10 (po 10% získaných relevantních dokumentech) znamená maximální přesnost na všech úrovních úplnosti větších než 0.10. Hodnoty jsou zprůměrovány přes všechny dotazy.
Střední průměrná přesnost (*Mean Average Precision*) `map`	Přesnost je měřena pro každý získaný dokument. Pokud je dokument relevantní, zvýší se průměrná přesnost o 1 (AveP). Q je počet získaných dokumentů, střední průměrná přesnost je průměrem všech získaných hodnot. $MAP = [\sum_{q=1}^{Q} AveP(q)]/Q$
Přesnost X (*Precision*) `P10, P15, P500,…`	Přesnost po X nejlépe hodnocených nalezených dokumentech. Pokud nebylo nalezeno X dokumentů, všechny chybějící jsou brány jako nerelevantní.
Přesnost R (*R-Precision*) `R-prec`	Přesnost po R nalezených dokumentech, kde R je celkový počet relevantních dokumentů k danému dotazu. Tzn., že se vynechá prvních R dokumentů a přesnost se měří až ve zbytku seznamu.

Geometrický průměr
(*Geometric Mean*)
`gmap`

Narozdíl od *map* (aritmetický průměr), *gmap* je geometrickým průměrem přesnosti odpovědi na jednotlivé dotazy. Např. pokud se průměrná přesnost dotazu A zvýší z 0.02 na 0.04, ale u dotazu B se sníží z 0.4 na 0.38, *map* se nezmění, *gmap* ano. *Gmap* lze vyjádřit aritmetickým průměrem logaritmů

$GMAP = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln AveP_i\right)$, kde $AveP_i$ je průměrná přesnost dotazu *i*.

Binární preference
(*Binary Preference*)
`bpref`

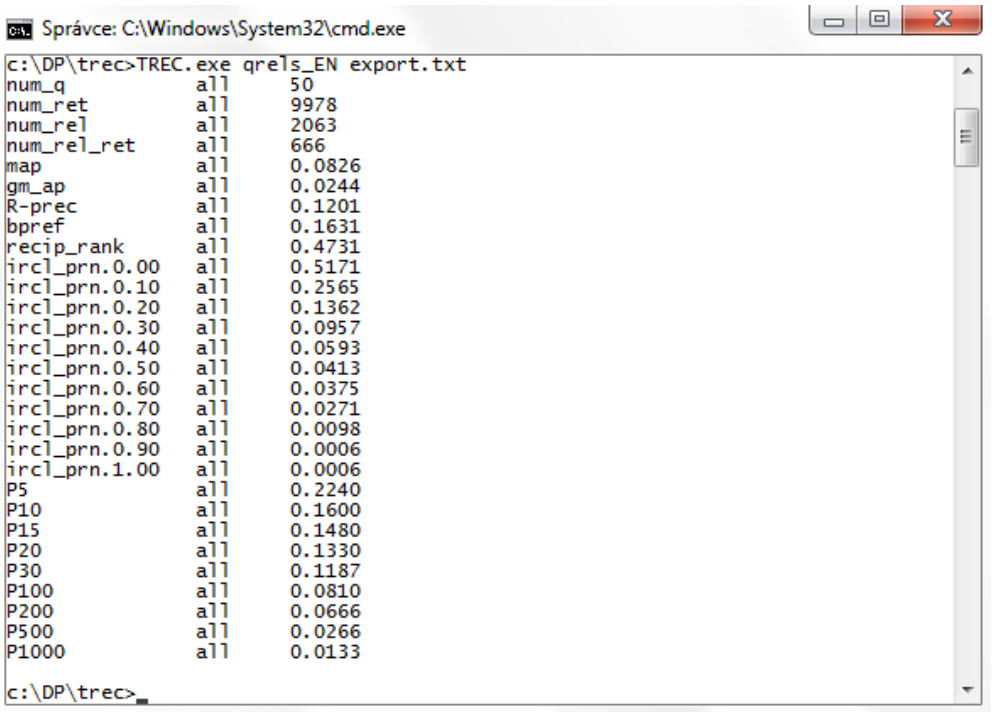
Vyjadřuje, do jaké míry jsou dokumenty považované za relevantní získány před dokumenty považovanými za nerelevantní.

$bpref = \frac{1}{R}\sum_{r}$ (1- |*n* ohodnocených výše než *r*| / min(*R*,*N*)), kde *R* počet získaných relevantních dokumentů, *N* počet získaných nerelevantních dokumentů, *r* je získaný relevantní dokument, *n* je dokument patřící do skupiny prvních *R* nerelevantních dokumentů.

Reciproční skóre
(*Reciprocal rank*)
`recip`

Reciproční skóre prvního nalezeného relevantního dokumentu vyjadřuje vztah $recip = \frac{1}{K}$, kde *K* je pozice dokumentu.

Možný výstup z aplikace je vidět na obrázku 3-2:



Obrázek 3-2: Výstup z aplikace TREC.

3.2 Předzpracování korpusu

Korpus je jazykově zpracován. Získaná slova jsou přidána do indexu a do souboru, který bude použit k vytvoření SP. K předzpracování textu jsou použity následující třídy Apache Lucene (viz oddíl 4.4):

`WhitespaceAnalyzer (WS)`	Rozlišuje slova pouze mezerami - *tokenizace*.
`StandardAnalyzer (ST)`	Rozlišuje stop slova, emaily, akronymy atd. Převádí na malá písmena.
`EnglishAnalyzer (EN)`	Rozšiřuje `StandardAnalyzer` o *stemming*, používá vestavěný slovník.
`EnglishLemmaAnalyzer (EL)`	*Lemmatizuje* slova na základě vlastního slovníku. Součástí knihovny Stanford Postagger.

`StandardAnalyzer` a `EnglishAnalyzer` pracují s tabulkou stopslov[45], tj. slov, která mají spíše funkční než sémantický význam.

3.3 Zpracování dotazu

K vytvoření dotazu, se kterým je aplikace schopná pracovat, se používá technika *parsování* dotazu, která každé slovo rozšíří o pole, ve kterém bude hledáno[46] (viz oddíl 4.4). Hledané informace jsou obsaženy v polích `text`, `headline` (např. dotaz `headline:"golden bear" text:"golden bear"` bude výraz hledat v obou zmíněných polích). K parsování busou použity následující třídy:

`QueryParser`	Parsuje v implicitně nastaveném poli, např. `headline.`
`MultiFieldQueryParser`	Parsuje v rámci výčtu nastavených polí, např. `{"text", "headline"}.`

3.4 Vytváření sémantických prostorů (knihovna S-Space)

Balík knihoven S-Space[47] [12] obsahuje algoritmy pro vytváření SP (`coals.jar`, `lsa.jar`, `hal.jar`, a další). Vstupem je soubor jazykově zpracovaných textů (např. `input.txt`), kde jedna řádka je rovna textu jednoho článku, případně dalších souvisejících textů (např. titulek). Výstupem je SP, se kterým lze dále pracovat (např. `coals.sspace`).

Následující příkaz vytvoří SP algoritmem Coals:

```
java coals.jar -d input.txt coals.sspace
```

Další uvedený příkaz pracuje s 64bitovým JVM[48] (`-d64`), pro zpracování garantuje 10 GB RAM (`-Xmx10g`), běží na čtyřech vláknech (`-t4`) a vypisuje hlášení do konzole (`-v`). Z původního textu vyloučí málo častá slova (`-F exclude=excluded.txt`, soubor `excluded.txt` je vygenerován z aplikace na základě IDF slov v indexu)[49]:

[45] Konkrétně slova *a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with*.

[46] Této technice lze říkat také filtrování obsahu. K vyhledávání se vybírají pouze určité části dokumentu.

[47] https://github.com/fozziethebeat/S-Space/wiki

[48] Java Virtual Machine

[49] Stopslova jsou vyloučena automaticky.

```
java -Xmx10g -d64 -jar coals.jar -d input.txt coals.sspace -t4 -v
-F exclude=excluded.txt
```

3.5 Rozšiřování dotazu

Rozšiřování dotazu je založeno na zjištění podobnosti slov z SP ke slovům dotazu. Pro hodnocení podobnosti slov je použita kosinová a eucleidovská vzdálenost, pro vzácnost slov koeficient IDF, který lze získat z vytvořeného indexu.

Navržené typy rozšíření dotazu:

- Rozšíření *o N nejpodobnějších* slov ke každému slovu dotazu (dále *v1*):
  Pro každé slovo dotazu je vybráno *N* slov, která jsou od něho nejméně vzdálená (mají nejvyšší koeficient podobnosti). Přidaná slova jsou seřazena podle koeficientu podobnosti (viz odstavec 2.5.4). Parametr *N* je nastavitelný (viz oddíl A.4 – parametr `limit`).

- Rozšíření jen *o vzácná slova* (dále *v2*):
  Z *N* nejpodobnějších slov (viz v1) jsou vybrána jen taková slova, která splňují nastavený limit vzácnosti (viz oddíl A.4 – parametr `limitRareWord`). Dotaz je rozšířen pouze o slova, jejichž IDF (získáno z indexu, viz odstavec 2.5.2) je nižší než stanovený limit.

- Rozšíření jen *vzácných slov* dotazu (dále *v3*):
  Dotaz je rozšířen o *N* nejpodobnějších slov k těm slovům původního dotazu, která splňují nastavený limit vzácnosti (viz oddíl A.4 – parametr `limitRareWordQuery`). Jejich IDF (získáno z indexu, viz odstavec 2.5.2) je nižší než stanovený limit.

- *Kombinace předchozích* variant (dále *k1*):
  Alternativy (podobná slova) se hledají pouze k vzácným slovům původního dotazu. Z *N* nejpodobnějších alternativ se vybírají pouze vzácná slova. Vzácnost je vyjádřena pomocí IDF.

- Rozšíření dotazu pouze *o slova související s více slovy* dotazu (dále *v4*):
  Dotaz je rozšířen jen o slova, která patří do množiny *N* nejpodobnějších slov pro *k* libovolných slov původního dotazu. Parametr *k* je nastavitelný (viz oddíl A.4 – parametr `limitExpansion`).

- *Kombinace všech* předchozích variant (dále *k2*):
  Alternativy se hledají pouze k vzácným slovům původního dotazu. Z *N* nejpodobnějších alternativ se vybírají pouze ta vzácná slova, která souvisejí s více slovy původního dotazu. Vzácnost je vyjádřena pomocí IDF.

Další možnosti rozšiřování dotazu společné pro předchozí varianty:

- Stanovení *minimálního koeficientu* zvoleného typu podobnosti porovnávaných vektorů:
  Koeficient podobnosti původního slova a jeho alternativy musí být vyšší nebo roven než je stanovený minimální koeficient podobnosti (viz oddíl A.4 – parametr `similarityCoef`).

- Upřednostnění slov původního dotazu (*boostování*):
  Slova původního dotazu jsou upřednostněna (viz odstavec 4.4.3). Boost je nastavitelný parametr (viz oddíl A.4 – parametr `boost`, interval `<1,10>`).

- Upřednostnění přidaných slov (tuto metodu nazývám *pokročilé* boostování):

Přidaná slova jsou boostována na základě jejich IDF (čím nižší IDF, tím vyšší boost z intervalu `<0,1>` s krokem 0,1). Tuto možnost lze aktivovat parametrem `boostAdv` (viz oddíl A.4).

3.6 Možnosti vlastní aplikace

Vlastní aplikace uživateli umožní:

- Vyhledání jednoho tématu z trénovací nebo testovací sady témat (viz odstavec 3.1.4).
- Vyhodnocení celé sady témat (viz odstavec 3.1.4).
- Spuštění předdefinované dávky. Tato dávka se konfiguruje v CSV souboru (`batch.csv`, viz oddíl A.6) a umožňuje spustit (a vyhodnotit) libovolný počet libovolně nakonfigurovaných sad témat.
  Výstupem všech předchozích tří variant je souhrnná tabulka se vstupní konfigurací a vyhodnocením aplikací TREC.
- Vyhledání nenalezených dokumentů k předdefinovanému tématu.
- Fulltextové vyhledávání libovolného dotazu.
  Výstupem bude seznam nalezených článků s možností náhledu na získaný článek.

Všechny výše zmíněné vyhledávací varianty je možné provést nad trénovací i testovací sadou témat. Aplikace dále umožňuje:

- Nastavení vyhledávací konfigurace (výběr analyzátoru, parseru, a další).
- Výběr typu automatického rozšíření dotazu.
- Nastavit příslušnou konfiguraci rozšiřování dotazu (SP, IDF, koeficient podobnosti, a další).

V rámci vytvoření komfortního prostředí aplikace uživateli umožní:

- Předpřipravit XML soubory s články.
- Vytvořit index na základě výběru analyzátoru[50].
- Exportovat soubor obsahující četnosti slov v indexu, který je použit pro vytváření SP.

Výsledky budou exportovány do:

- CSV souboru (`results.csv`) v případě, že bude výsledkem souhrnná tabulka.
- LOG souboru ve všech případech (`<yyyyMMddHHmmss>.log`).

Vstupní konfigurace aplikace bude uchována v konfiguračním souboru (`config.ini`). Aplikace bude logovat všechny zásadní akce a chybová hlášení (`activity.log`).

[50] Vstupním parametrem pro vytvoření indexu je volba analyzátoru, proto výběr analyzátoru vede i k výběru hotového indexu.

4 IMPLEMENTACE

Tato kapitola popisuje navržené programové řešení [13]. V první části se zabývá architekturou aplikace. V dalších částech se věnuje vytvořenému programu, vyhledávání a rozšiřování dotazu.

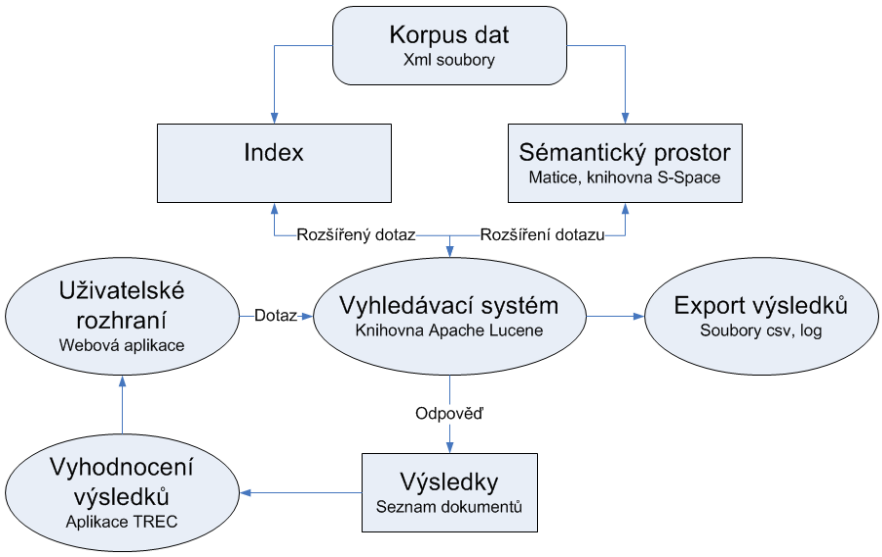
Aplikace je postavena na platformě Java EE[51] a frameworku JSF[52], vyhledávací systém je možné spouštět z prohlížeče. Jako aplikační server, zajišťující veškerou požadovanou funkcionalitu, je použit Apache Tomcat verze 7[53].

Aplikace je vyvíjena v prostředí Windows 7 Professional 64 bit a Eclipse pro Java EE[54]. Exportem projektu do souboru WAR je zajištěna přenositelnost mezi servery a operačními systémy.

Bloky kódu uvedené v této kapitole mají pouze ilustrativní charakter.

4.1 Základní schéma aplikace

Komponenty aplikace a jejich interakci zobrazuje obrázek 4-1:



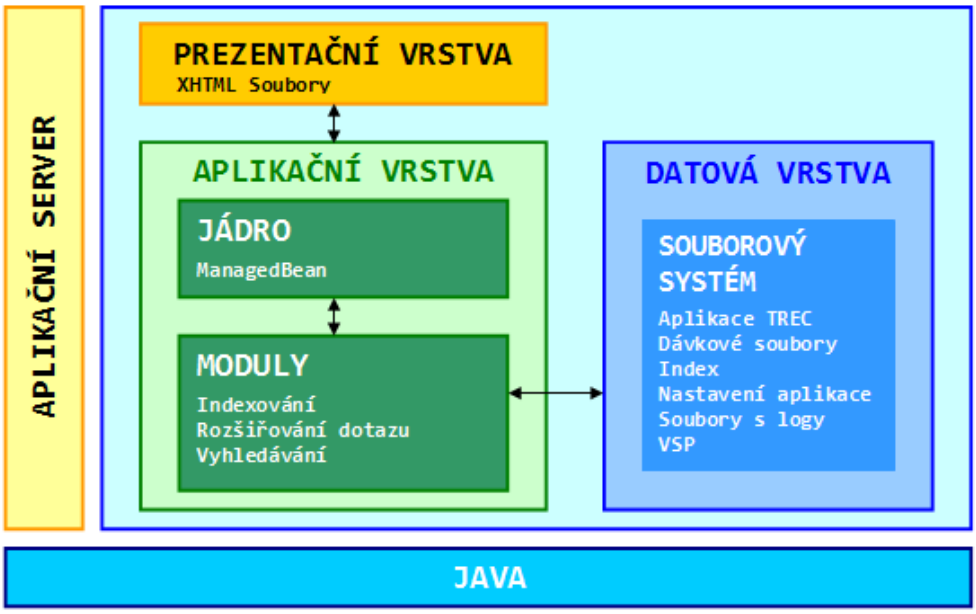
Obrázek 4-1: Základní schéma aplikace. Uživatel zadá dotaz, vyhledávací systém jej rozšíří pomocí SP a vyhledá články v indexu. Nalezené články, případně jejich vyhodnocení, vrátí jako odpověď uživateli.

4.2 Architektura aplikace

Aplikace je rozdělena do tří vrstev, prezentační, aplikační a datové (obrázek 4-2). Prezentační vrstva je tvořená webovou stránkou, která obsahuje veškeré ovládací prvky aplikace a dvě různé tabulky pro zobrazení odpovědi (odstavec 4.2.1). Aplikační vrstva je tvořena jádrem

[51] Java Enterprise Edition pro serverové aplikace.
[52] JavaServer Faces, framework založený na JSP a servletech.
[53] http://tomcat.apache.org/index.html
[54] http://www.eclipse.org/

(*ManagedBean*[55]) a třídami pro indexování, vyhledávání a rozšiřování dotazu (odstavec 4.2.2). Datová vrstva zajišťuje práci s daty (odstavec 4.2.3). Princip celého vyhledávacího systému je znázorněn na obrázku 4-1. Obrázek 4-2 znázorňuje spolupráci jednotlivých vrstev.



Obrázek 4-2: Architektura aplikace.

4.2.1 Prezentační vrstva

Prezentační vrstva aplikace je založena na JSF a RichFaces[56], je tvořena XHTML[57] soubory `index.xhtml` (importuje zbylé dva XHTML soubory), `header.xhtml` (hlavička stránky a parametry vyhledávání) a `search.xhtml`, který slouží k zobrazení odpovědi systému. K vytvoření prezentační vrstvy jsou dále využity technologie Ajax[58], CSS[59] a JavaScript[60].

4.2.2 Aplikační vrstva

Jádrem aplikace je *ManagedBean*[61] `dp.beans.MySearch` a třídy pro vyhledávání, indexování a rozšiřování dotazu. Součástí aplikační vrstvy jsou knihovny jazyka Java, Apache Lucene, SSpace pro vytváření SP, Apache Digester[62] pro parsování XML souborů, Apache Tomcat a další.

Vstupem aplikační vrstvy je uživatelův dotaz[63] a případně parametry jeho rozšíření, výstupem je seznam získaných dokumentů nebo jejich vyhodnocení aplikací TREC.

---

[55] Java třída spravována JSF frameworkem. Obsluhuje všechny komponenty aplikace.

[56] Framework postavený na JSF a Javě, viz http://www.jboss.org/richfaces.

[57] *Extensible HyperText Markup Language*, obsahuje komponenty JSF a RichFaces frameworku.

[58] *Asynchronous JavaScript and XML,* technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání.

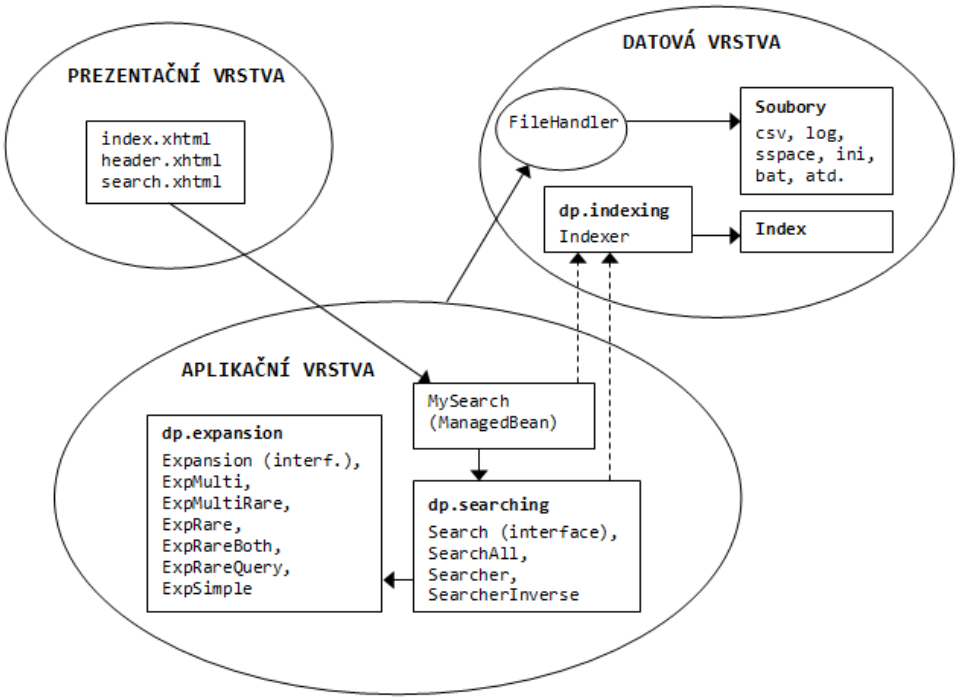
[59] *Cascading Style Sheets*, jazyk pro popis způsobu zobrazení stránek napsaných v HTML, XML atd.

[60] Programovací jazyk založený na Javě. Pro www stránky, vkládaný přímo do HTML kódu stránky.

[61] Java třída spravována JSF frameworkem. Obsluhuje všechny komponenty aplikace.

[62] http://commons.apache.org/proper/commons-digester/

[63] Může jít o sadu dotazů nebo dávku. I ty se ale skládají z jednotlivých dotazů.



Obrázek 4-3: Architektura systému.

4.2.3 Datová vrstva

Datová vrstva aplikace je postavena na souborovém systému a je řízena třídou `dp.common.FileHandler`[64], která disponuje metodami pro spouštění dávkových souborů (BAT[65]) a pro práci s XML soubory (zajištění jejich validace, mazání, přejmenování apod.), SSPACE, CONF, CSV, soubory indexu atd.

V následujících odstavcích jsou stručně popsány důležité třídy aplikace.

Třída MySearch

Třída `dp.beans.MySearch` je jádrem aplikační vrstvy. Disponuje objekty, které jsou provázány s komponentami prezentační vrstvy (seznamy položek dropboxů, nastavení rozšíření dotazu apod.), a metodami, které obsluhují uživatelské požadavky. Mezi klíčové metody patří metoda `Search()`, která je rozcestníkem pro jednotlivé typy vyhledávání, `CreateIndex()` pro spuštění procesu vytvoření indexu, `batchProcessing()` pro hromadné spuštění více sad témat a `importCsvFile()` pro zobrazení CSV souboru s výsledky.

Třída Indexer

Třída `dp.indexing.Indexer` má na starost proces vytvoření indexu[66]. K rozdělení XML souboru na jednotlivé články používá třídu `Digester` a k vlastní indexaci (přidávání článků do indexu) instanci třídy `IndexWriter`.

---

[64] V některých případech může být ale řízena přímo třídami aplikační vrstvy.

[65] Ovládání datové vrstvy prostřednictvím dávkových souborů BAT mi vyhovuje z toho důvodu, že je snažší definovat novou operaci pomocí příkazů příkazové řádky, než doplňovat další kód do aplikace.

[66] Současně vytváří i textový soubor, ze kterého jsou vytvářeny SP.

Třída Searcher

Třída `dp.searching.Searcher` implementuje rozhraní `dp.searching.Search` a stará se o vyhledávání dokumentů na základě dotazu uživatele. Hlavní metodou této třídy je metoda `search()`, která zpracuje vyhledávací dotaz, případně ho rozšíří o další slova v závislosti na výběru typu rozšíření, a vrátí seznam (`List<DocShow>`) získaných dokumentů.

Další třídou implementující rozhraní `Search` je třída `SearchSet`, která obsluhuje vyhledávání celé sady testovacích témat (pro každé téma volá metodu `Searcher.search()`) a třída `SearcherInverse`, jejíž metoda `search()` vrací seznam nenalezených dokumentů k danému dotazu.

Balík `dp.searching` dále obsahuje třídu `Common` disponující statickými metodami, které jsou využívány zbylými třídami balíku.

Třídy balíku expansion

Balík `dp.expansion` obsahuje rozhraní `Expansion`, které definuje metodu `expansion()`. Toto rozhraní implementují třídy `ExpMulti`, `ExpMultiRare`, `ExpRare`, `ExpRareBoth`, `ExpRareQuery` a `ExpSimple`, které slouží k rozšiřování dotazu. Každá z těchto tříd koresponduje s jedním z typů rozšíření dotazu a používá statické metody třídy `Common`, která je součástí balíku. Metody třídy `Common` lze kombinovat za účelem vytvoření nového typu rozšíření (vstupem je vždy dotaz, výstupem rozšířený dotaz).

Například metoda `expansion()` třídy `ExpMultiRare` rozšiřuje vzácná slova dotazu o vzácná slova taková, která souvisejí s více slovy dotazu. Zvýhodňuje navíc původní dotaz i přidaná slova podle jejich IDF:

```
@Override
public String expansion(String query, Settings settings)
{
  String originalQuery = "";
  String expandedQuery = "";
  // boost puvodniho dotazu
  originalQuery = Common.getOriginalQueryBoosted(query, settings);
  // vybrani vzacnych slov z dotazu
  expandedQuery = Common.getRareWordsQueryForExpansion(query, settings);
  // rozsireni vzacnych slov dotazu
  expandedQuery = Common.expandQueryOfRareWords(expandedQuery, settings);
  // vybrani tech slov, ktere se vyskytuji vicekrat
  expandedQuery =
    Common.getCooccurences(query, settings,Common.getHashMap(expandedQuery));
  // spocteni pridanych slov
  settings.setExpandedWordsCount(settings.getExpandedWordsCount() +
    Common.getWordCount(expandedQuery, settings));
  // boost pridanych slov
  expandedQuery = Common.getExpandedQueryBoosted(expandedQuery, settings);
  return originalQuery + " " + expandedQuery;
}
```

Třída Settings

Třída `dp.data.Settings` obsahuje veškerá nastavení aplikace. Ze souboru `dp.properties` načte cestu ke kořenovému adresáři obsahující data aplikace, ze souboru `config.ini` iniciační nastavení všech hodnot (viz příloha A). Nastavení jsou dále měněna uživatelem v rámci jednoho životního cyklu[67] ManagedBeanu `MySearch`.

Třída `Settings` obsahuje[68]:

- cesty k podadresářům kořenového adresáře s daty,
- názvy souborů, které aplikace používá,
- proměnné vázané na komponenty prezentační vrstvy aplikace (limity vzácnosti slov, podobnostního koeficientu, výběr SP, analyzátoru, parseru, a další).

Na základě vybraného typu:

- analyzátoru vytvoří příslušnou instanci třídy `Analyzer`,
- rozšíření vytvoří instanci třídy `dp.expansion.Expansion`,
- parseru vytvoří instanci třídy `QueryParser`,
- podobnosti vytvoří instanci třídy `SimType`,
- ohodnocené sady témat (trénovací, testovací) vybere soubor, obsahující daná témata.

Uživatel si může svou preferovanou konfiguraci nastavit v `config.ini`.

Další třídy

Třída `Tools` balíku `dp.common` disponuje například metodami pro logování, vypisování chybových hlášení na obrazovku, pro práci s řetězci a naplňování objektů jednotlivými seznamy (seznam SP, typů rozšíření, témat apod.) načtenými ze souborů CONF. Třídy `DtoCompInt` a `DtoCompStr` poskytují metody pro řazení tabulek, třídy `Doc` a `DocShow` definují atributy dokumentu a třída `Main` balíku `dp.runable` slouží ke konzolovému spuštění aplikace.

4.3 Příprava vstupních dat

K zajištění validity dodaných XML souborů je nutné přidat značku `</DOCUMENT>` na začátek a konec každého souboru, nahradit znak `"&"` jeho escape sekvencí a odstranit značky `<P>` a `</P>`. Následující blok kódu tyto změny provede a uloží do souboru s příponou `repaired`[69]:

```
public void replaceInFile(File file)
{
  BufferedReader tempFileReader =
    new BufferedReader(new InputStreamReader(new FileInputStream(file)));
  File tempFileBuiltForUse = new File(file + ".repaired");
  Writer changer = new BufferedWriter(new FileWriter(tempFileBuiltForUse));
  String lineContents;

  // Pridani tagu <DOCUMENT> na zacatek souboru
  changer.write("<DOCUMENT>");
```

[67] Životní cyklus je ukončen buď restartováním aplikace nebo reloadem hlavní stránky.

[68] Ke všem objektům obsahuje příslušné getry a případné setry.

[69] Aplikace uživateli umožňuje i smazání původních XML souborů a přejmenování souborů s příponou REPAIRED na XML (viz uživatelská dokumentace).

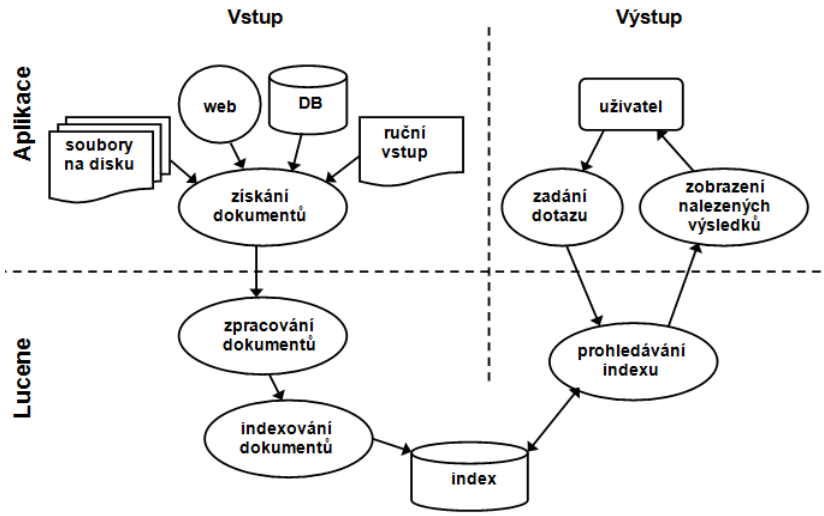
```
  String lineByLine = "";
  while ((lineContents = tempFileReader.readLine()) != null)
  {
    lineByLine = lineContents;
    lineByLine = lineByLine.replaceAll("&", "&amp;") + "\n";
    lineByLine = lineByLine.replaceAll("<P>", "");
    lineByLine = lineByLine.replaceAll("</P>", "");
    changer.write(lineByLine);
  }
  // Ukonceni souboru tagem </DOCUMENT>
  changer.write("</DOCUMENT>");
  changer.close();
  tempFileReader.close();
}
```

Přípravě vstupních dat je třeba věnovat zvláštní pozornost. Jiný korpus pravděpodobně bude obsahovat jinak strukturované soubory a je proto třeba vytvořit nový parser, který tyto soubory převede do jednotného formátu.

4.4 Vyhledávání a indexování dat pomocí Apache Lucene

Apache Lucene je open source API napsané v jazyce Java, která slouží pro indexování a vyhledávání. Lucene není samostatná aplikace, poskytuje pouze *search engine*[70].

Jádro aplikace disponuje logikou pro vytváření, aktualizaci a rušení indexu, pro jazykové zpracování textu (dotazu, dokumentu) a pro vyhledávání dokumentů [14]. Data pro indexaci a vyhledávání si musí připravit sám uživatel. Silnou stránkou Lucene je práce s uživatelskými dotazy [15].



Obrázek 4-4: Integrace Apache Lucene s aplikací. Hlavní síla Lucene spočívá v indexování dokumentů a ve vyhledávání nad hotovým indexem [14].

Obrázek 4-4 znázorňuje typické poskládání jednotlivých komponent aplikace postavené na vyhledávacím stroji Lucene. Z jedné strany získání různých typů dokumentů (zde korpus XML souborů), jejich zpracování (vytvoření a analýza dokumentu) a vytvoření indexu, a ze strany druhé uživatelský dotaz nad vytvořeným indexem a vrácení odpovědi zpět uživateli.

[70] Není to samostatně spustitelná aplikace, poskytuje služby pro indexování a vyhledávání.

4.4.1 Důležité knihovny a třídy

Účely této aplikace pokrývají následujícící knihovny:

- `Lucene-Core.jar` obsahuje jádro Lucene (vyhledávání, idexace, analyzátory, parsery),
- `Lucene-Analyzers.jar` obsahuje další analyzátory textu (`EnglishAnalyzer`),
- `Lucene-Highlighter.jar` obsahuje třídy pro zvýraznění textu.

Třídy pro indexování:

- `IndexWriter` vytváří index a umožňuje k němu přistupovat (`add()`, `delete()`, `update()`),
- `Directory` značí umístění indexu (`FSDirectory`, `RAMDirectory`),
- `Analyzer` pro jazykové zpracování textu (tokenizace, normalizace, anotace),
- `Document` je třída pro reprezentaci vlastního dokumentu,
- `Field` pro pole dokumentu, ze kterých se dokument skládá.

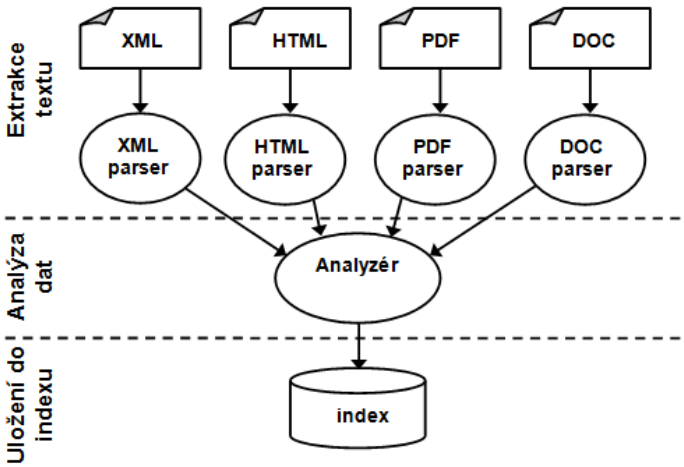
Třídy pro vyhledávání:

- `QueryParser` přeloží dotaz do podoby přijatelné pro vyhledávací systém (objekt `Query`),
- `Query` pro vlastní dotaz (`TermQuery`, `BooleanQuery`, `PhraseQuery`),
- `IndexSearcher` pro vyhledávání v indexu,
- `TopDocs` obsahuje *N* nejlepších nalezených souborů dle jejich skóre,
- `Term` je základní jednotkou vyhledávání.

4.4.2 Vytvoření indexu

Vytvoření indexu probíhá ve třech krocích [15]:

- *Extrakce* textových informací ze zdrojových dat. Text je rozdělen do polí[71] (např. text, headline, byline) a ty jsou vloženy do jednotlivých *dokumentů*[72] (článků).
- *Analýza* (jazykové zpracování) dokumentu.
- *Uložení* polí vytvořeného dokumentu do indexu[73].



Obrázek 4-5: Proces indexování dokumentů.

[71] Každé pole (Field) má jméno (identifikátor) a textovou či binární hodnotu.

[72] Dokument je základní jednotka vyhledávání. Je to kontejner, který obsahuje jedno nebo více polí.

[73] Jde o invertovaný index, tj. seznam slov a dokumentů, ve kterých se vyskytují.

Následující bloky kódu naznačují proces vytvoření dokumentu. Spouštěcí metoda `createIndex()` otevře adresář s indexem, vytvoří objekt třídy `IndexWriter` za použití vybraného analyzátoru k analýze textových informací, vytvoří nový objekt třídy `dp.indexing.Indexer` pro každý další XML soubor. Tím iniciuje konstruktor, který předá všechny dokumenty v souboru ke zpracování.

```
public void createIndex()
{
  Directory dir = FSDirectory.open(new File(settings.getIndexDir()));
  Analyzer analyzer = new EnglishAnalyzer(Version.LUCENE_36);
  IndexWriterConfig config;
  config = new IndexWriterConfig(Version.LUCENE_36, analyzer);

  writer = new IndexWriter(dir, config);
  indexer = new Indexer();
  // pseudokod:
  pro vsechny soubory s priponou xml v adresari dataDir proved:
  indexer = (Indexer) digester.parse(file);
}
```

Konstruktor `dp.indexing.Indexer` při každém nalezení značky `<doc>` volá metodu `addDocument()`. Toto umožňuje třída `Digester`, která slouží pro zpracování XML souborů (umožnuje propojit značky v XML dokumentu s atributy třídy `dp.data.Doc`).

```
public Indexer()
{
  Digester digester = new Digester();
  digester.setValidating(false);
  // inicializace Indexeru a provazani znacky <DOC> s dokumentem
  digester.addObjectCreate("DOCUMENT", Indexer.class );
  digester.addObjectCreate("DOCUMENT/DOC", Doc.class );
  // sparovani jednotlivych znacek XML s atributy Doc.class
  digester.addCallMethod("DOCUMENT/DOC/DOCNO", "setDocno", 0);
  digester.addCallMethod("DOCUMENT/DOC/HEADLINE","setHeadline", 0);
  digester.addCallMethod("DOCUMENT/DOC/TEXT", "setText", 0);
  // pri nalezeni dalsiho clanku (tagu <doc>) vola metodu addDocument(),
  // ktera clanek prida do indexu
  digester.addSetNext("DOCUMENT/DOC", "addDocument" );
}
```

Metoda `addDocument()` přidá extrahovaný článek k zapsání do indexu (objektu třídy `IndexWriter`):

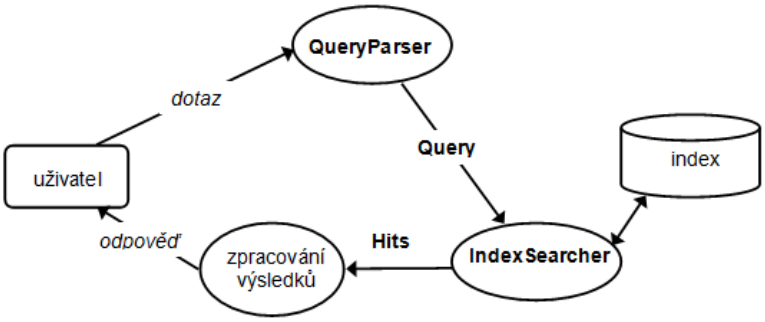
```
public void addDocument(Doc document)
{
  doc  = new Document();
  Field docno = new Field("docno", document.getDocno(), <parametry>);
  Field text = new Field("text", document.getText(), <parametry>);
  Field headline = new Field("headline", document.getHeadline(),<parametry>);
  doc.add(docno);
  doc.add(headline);
  doc.add(text);
  writer.addDocument(doc); // ulozeni do indexu
}
```

`<parametry>` mohou být například:

`Field.Store.YES`	Obsah pole bude uložen do indexu.
`Field.Index.ANALYZED`	Pole bude před uložením do indexu analyzováno (text, headline).
`Field.Index.NOT_ANALYZED`	Pole nebude před uložením do indexu analyzováno (docno).

4.4.3 Vyhledávání v indexu

Uživatelský dotaz je zpracován metodou `parse()` třídy `QueryParser`, výstupem je instance třídy `Query`. Tento objekt je předán instanci třídy `IndexSearcher`, která je inicializována s prohledávaným indexem a objektem `Analyzer` pro zpracování textu. Metoda `dp.searching.Searcher.search()` provede hledání a jako výsledek vrátí instanci třídy `TopDocs`, která obsahuje *N* nalezených nejlépe ohodnocených dokumentů (nalezeným souborům se říká *Hits*) [14].



Obrázek 4-6: Interakce tříd pro vyhledávání.

Mezi základní výrazy pro formulaci dotazu patří:

Výraz	Příklad	Nalezne dokument, který obsahuje
`OR`	*love* `OR` *war*	slovo *love* nebo *war*. Výraz OR lze vypustit.
`AND`	*love* `+`*war*	slova *love* i *war*. Výraz + lze nahradit výrazem AND.
`-`	*love* `-`*war*	slovo *love* ale neobsahuje slovo *war*.
<pole>:slovo	`title:`*war*	slovo *war* v poli *title*.
`""`	`"`*love and war*`"`	celý řetězec "*love and war*".
*	*war**	slova zažínající na *war*, např. *war, warranty, warp*.
`""~`*<číslo>*	`"`*love war*`"~3`	slova *love* i *war* do vzdálenosti tří pozic od sebe.

Metoda `search()` otevře index, vytvoří instance tříd `IndexReader` a `IndexSearcher` pro práci s indexem, přeloží zadaný dotaz `q` do formy srozumitelné vyhledávacímu stroji a provede vyhledání dokumentů (vrátí *N* nejlépe ohodnocených):

```
public void search(String q)
{
  Directory dir = FSDirectory.open(new File(indexDir));
  IndexReader r = IndexReader.open(dir);
  IndexSearcher is = new IndexSearcher(r);
  Analyzer analyzer = new EnglishAnalyzer(Version.LUCENE_36);
  QueryParser parser = new QueryParser(Version.LUCENE_36, "text", analyzer);
  Query query = parser.parse(q);
  TopDocs hits = is.search(query, N);
}
```

Ohodnocení nalezených dokumentů – skórování

Instance třídy `TopDocs` obsahuje *N* nejlépe ohodnocených nalezených dokumentů. K ohodnocení dokumentu vzhledem k zadanému dotazu Lucene používá vzorec[74]:

$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{t \in q} \left[tf(t \in d) * idf(t)^2 * boost(t, field \in d) * lengthNorm(t, field \in d) \right]$$

kde jednotlivé výrazy znamenají [15]:

- *t* je term, *field* je pole, které *t* obsahuje, *d* je dokument a *q* dotaz.
- *tf* a *idf* viz kapitola TF-IDF.
- *boost* upřednostňuje výskyt *t* v zadaném poli (odstavec Ovlivňování výsledného skóre).
- *lengthNorm* představuje normalizovanou hodnotu pro pole. Tato hodnota je stanovena při indexování. Term vyskytující se v kratším poli získá vyšší skóre.
- *coord* značí faktor založený na počtu termů dotazu, které se v dokumentu vyskytují. Dokument obsahující více slov dotazu bude mít vyšší skóre.
- *queryNorm* je normalizovanou hodnotou dotazu (suma druhých mocnin vah všech termů dotazu). Nemá vliv na relevanci dokumentu (je stejná pro všechny dokumenty k danému dotazu).

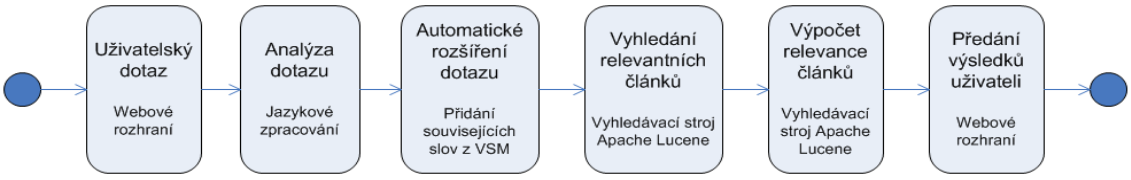
Ovlivňování výsledného skóre – boostování

Boosting upřednostňuje daný dokument (pole, term) na úkor ostatních a tím ovlivňuje výsledné skóre dokumentu. Boost faktor může být nastaven:

- V průběhu procesu indexace:
  Pro **dokument**, voláním metody `document.setBoost()` před jeho vložením do indexu.
  Pro **pole**, voláním metody `field.setBoost()` před vložením pole do dokumentu.
- V průběhu vyhledávání:
  Pro **dotaz**, voláním metody `Query.setBoost()`.
  Pro **výrazy** dotazu (např. *love^3 +"war or peace"^0,5*).

4.5 Automatické rozšiřování dotazu pomocí S-Space

Vstupem vyhledávacího stroje je uživatelův dotaz. Počítač ho chápe jako řetězec znaků, který musí zpracovat. A dále, v rámci úlohy rozšiřování dotazu, najít k jednotlivým slovům vhodné alternativy (obrázek 4-8).



Obrázek 4-7: Zpracování uživatelského dotazu.

SP Algoritmy knihovny S-Space implementují interface `SemanticSpace`, společné API pro práci s výsledným SP, které definuje metody[75]:

[74] http://lucene.apache.org/core/3_6_0/api/all/org/apache/lucene/search/Similarity.html

- `getSpaceName()` pro získání jména a konfigurace SP,
- `getVector()` vrací sémantický vektor daného slova ve SP,
- `getVectorLength()` vrací délku vektoru,
- `getWords()` vrací slova reprezentována SP,
- `processSpace()` provede normalizaci matice a její SVD rozklad.

Další důležitou třídou je `WordComparator`, která poskytuje metodu `getMostSimilar()` pro nalezení *N* sémanticky nejpodobnějších slov k zadanému slovu na základě vybraného typu podobnosti (např. `SimType.COSINE`, `SimType.EUCLIDEAN`).

Následující blok kódu inicializuje SP a instanci třídy `WordComparator`. Pak prochází vstupní dotaz a ke každému slovu vyhledá *N* nejpodobnějších, které pak přidá do původního řetězce:

```
protected static String getExpandedQuery(String query)
{
  StaticSemanticSpace sspace = new StaticSemanticSpace(matrixFile);
  TokenStream streamQuery;
  streamQuery = analyzer.tokenStream(null, new StringReader(query));
  WordComparator wc = new WordComparator();
  String word, temp; word = temp = "";
  while (streamQuery.incrementToken())
  {
    word = streamQuery.getAttribute(CharTermAttribute.class).toString();
    SortedMultiMap<Double,String> mostSimilar = null;
    // nalezeni N nejpodobnejsich slov na zaklade kosinoveho porovnavani
    mostSimilar = wc.getMostSimilar(word, sspace, N, SimType.COSINE);
    // rozsireni puvodniho dotazu
    for (Entry<Double, String> entry : mostSimilar.entrySet())
      temp = temp + " " + entry.getValue();
  }
  return temp;
}
```

4.6 Vyhodnocení odpovědi

Aplikace TREC je vytvořena v jazyku C a nemá Java API. Výstup z aplikace je proto nutné přesměrovat z příkazového řádku (resp. konzole Eclipse) do CSV souboru.

Metoda `trecResultsCreate()`[76] spustí dávku (`trec_run.bat`, která obsahuje příkaz `TREC.exe qrels_EN export.txt`) pro vyhodnocení odpovědi aplikací TREC. Výstup z konzole kopíruje do CSV souboru (uloží vždy poslední údaj získaného řádku, např. číslo `0.0034` z řádky `"map all 0.0034"` a oddělí ho středníkem):

```
private void trecResultsCreate()
{
  PrintWriter out = new PrintWriter(new FileWriter(file, true));
  Runtime rt = Runtime.getRuntime(); // spusteni cmd  pro vyhodnoceni TREC
  Process pr = rt.exec("trec_run.bat");
  BufferedReader input;
  input = new BufferedReader(new InputStreamReader(pr.getInputStream()));
```

[75] https://github.com/fozziethebeat/S-Space/wiki/GettingStarted

[76] Metoda je ilustrativní, ukazuje základní operace nutné k pochopení přesměrování výstupu.

```
  String line = null;
  // presmerovani vystupu z obrazovky/konzole do souboru
  while((line = input.readLine()) != null)
  {
    // rozdeli radku vynechanim mezer a tabulatoru - napr. "map  all 0.0034"
    String[] data = line.split("\\s+");
    String lastEntry = data[data.length - 1]; // vezme posledni udaj z radku
    out.printf(lastEntry + ";"); // uloži do csv souboru a oddeli ";"
  }
  out.close();
}
```

4.7 Rozšiřitelnost aplikace

Aplikace lze rozšířit přidáním nového:

- SP,
- analyzátoru,
- typu parsování dotazu,
- typu výpočtu sémantické podobnosti,
- způsobu vyhledávání dokumentů,
- typu rozšíření dotazu.

Tato rozšíření lze provést změnou příslušného CONF souboru, přidáním objektu do příslušného getru třídy `dp.data.Settings` a, v případě přidání nového typu rozšíření dotazu, vytvořením nové třídy implementující rozhraní `dp.expansion.Expansion`.

Aplikace je pravděpodobně rozšiřitelná i v rámci korpusu CLEF (práce s jinými XML soubory obsahujícími dokumenty, jinými testovacími sadami vyhledávacích témat).

5 TESTOVÁNÍ

Pro testovací účely standardně vyhledávám pouze v textu a titulku článku. Pokud není uvedeno jinak, je do odpovědi zahrnuto 200 nejlépe ohodnocených dokumentů (*hits*) ke každému tématu (pro celou sadu 50 témat tedy 10000). Pro rozšiřování dotazu jsou použity SP vytvořené pomocí algoritmů LSA a Coals. Pro měření vzdálenosti mezi vektory je použita pouze kosinová vzdálenost[77] (implementace eukleidovské vzdálenosti v knihovně S-Space obsahuje chybu, jmenovatel není odmocněn). Pro zpracování textu jsou testovány analyzátory EN a ST (WS dosahuje řádově slabších výsledků, EL je otestován okrajově). Jednotlivé konfigurace se testují na trénovací (50 dotazů, celkem 157 slov) a testovací sadě (50 dotazů, celkem 138 slov).

Testování je zaměřeno zejména na přesnost *P* (*rel_ret*/*ret*), úplnost *R* (*rel_ret*/*rel* –viz odstavec 2.1.2) a *map* (viz odstavec 3.1.4). *P* i *R* vyplývají z počtu získaných relevantních dokumentů *rel_ret*. *Map* v sobě kromě průměrné přesnosti zahrnuje také informaci o tom, na jakých pozicích v odpovědi se relevantní dokumenty vyskytují. Tyto míry pokrývají potřeby uživatele (najít relevantní dokumenty hned na první stránce, před nerelevantními).

5.1 Postup získávání výsledků

Konfigurace[78] rozšíření se testuje nejprve na trénovacích datech. Pokud je nalezena zajímavá konfigurace[79], je otestována i na testovacích datech. Teprve na testovacích datech je vyzkoušeno boostování. Za úspěch je považováno zlepšení odpovědi na testovacích datech.

5.2 Rozšiřování dotazu

Vzhledem k množství různých kombinací (viz oddíl 3.5) jsou vybrány ty konfigurace, které se pro dané typy rozšíření jeví nejzajímavější. Tato kapitola bude nejdříve věnována vlivu rozšiřování dotazu na *P* a *R*, dále na *map* a vlivu boostování na předchozí výsledky.

Grafy se skládají ze souvisejících měření, osy obsahují různé míry vyhodnocení aplikací TREC (dále jen míry TREC) a konfigurace rozšíření dotazu, tj. informaci o použitém analyzátoru (ST, EN), algoritmu SP (LSA, Coals) a sadě dotazů (TR znamená trénovací sadu, TST testovací. TST není v grafech uváděno), případně o počtu přidaných slov nebo boost faktoru (B). Pokud není uveden SP, nebylo provedeno rozšíření dotazu (dále *baseline*[80], zkráceně BL – v gafech není BL uváděno). Např. ST_LSA znamená, že byl použit analyzátor ST, SP LSA a testovací sada. ST_TR znamená, že jde o baseline nad trénovací sadou.

Každé měření obsahuje tabulku s číslem testované konfigurace, vlastní konfigurací (viz oddíly A.4 a A.6), naměřenými výsledky (*ret*, *rel*, *rel_ret*, *P*, *R*) a počtem přidaných slov (PS).

---

[77] Aplikace ale umožňuje pracovat i s dalšími algoritmy (např. HAL) i mírami podobnosti (eukleidovská vzdálenost, míra pearsonovy korelace apod.).

[78] Konfigurace ve smyslu nastavení jednotlivých limitů, koeficientů, matice apod.

[79] Zajímavá ve smyslu zvýšení P, R nebo map.

[80] Baseline je základní stav (vyhledávání bez rozšiřování dotazu), oproti kterému testujeme zlepšení rozšiřováním dotazu.

5.2.1 Testování přesnosti a úplnosti

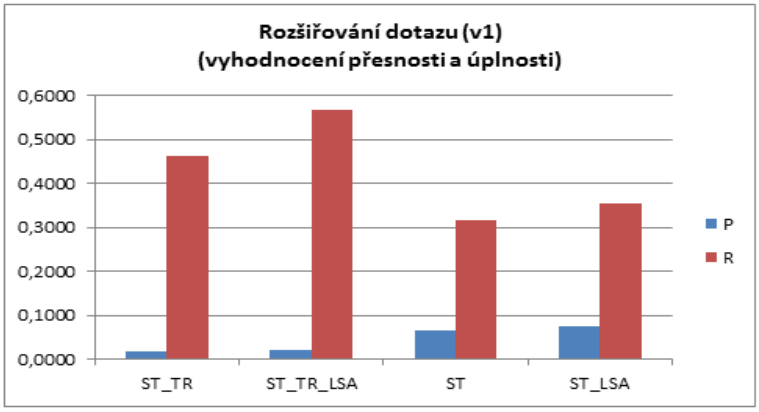
Rozšiřování dotazu - varianta v1

Na obrázku 5-1 je vidět vliv rozšíření typu v1 na *P* a *R*. Na obrázku 5-2 je vidět vliv tohoto rozšíření na ostatní míry TREC. Každé slovo dotazu bylo rozšířeno o 5 slov, celkově bylo přidáno 690 slov pro trénovací sadu a 665 slov pro testovací sadu.

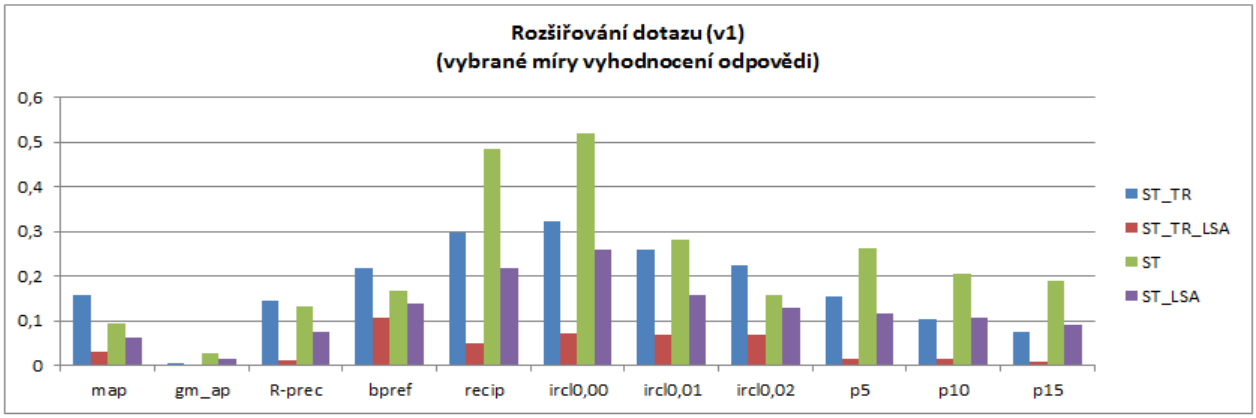
č.	A	SP	Sada	LE	ret	rel	rel_ret	map	P	R	PS
1	ST		TR		9734	375	174	0,1563	0,0179	0,4640	
2	ST	LSA	TR	5	9945	375	213	0,0293	0,0214	0,5680	690
3	ST		TST		9978	2063	652	0,0939	0,0653	0,3160	
4	ST	LSA	TST	5	10000	2063	735	0,0627	0,0735	0,3563	665

Tabulka 5-1: Konfigurace a naměřené výsledky (v1).

Na TR se *P* se zvýšila zhruba o 20%, o 12% na TST. *R* v obou případech o 12%. Další míry TREC se znatelně snížily.



Obrázek 5-1: Vliv rozšíření v1 na přesnost a úplnost (v1).



Obrázek 5-2: Vvliv rozšíření v1 na vybrané míry TREC (v1).

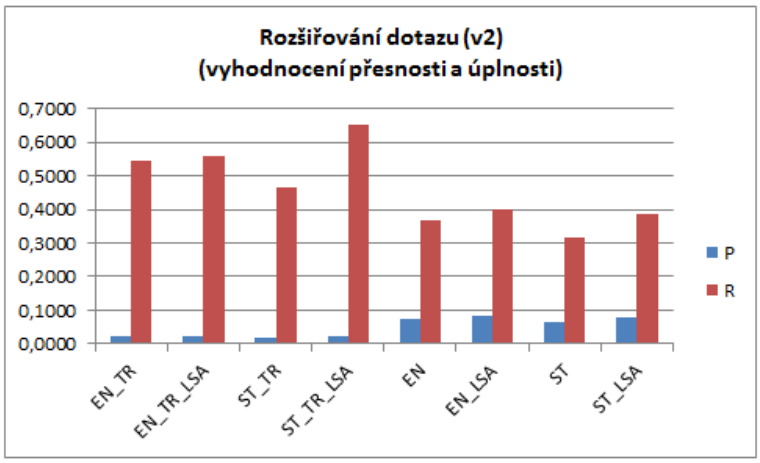
Rozšiřování dotazu - varianta v2

Rozšíření v2 je vidět na obrázcích 5-3 a 5-4.

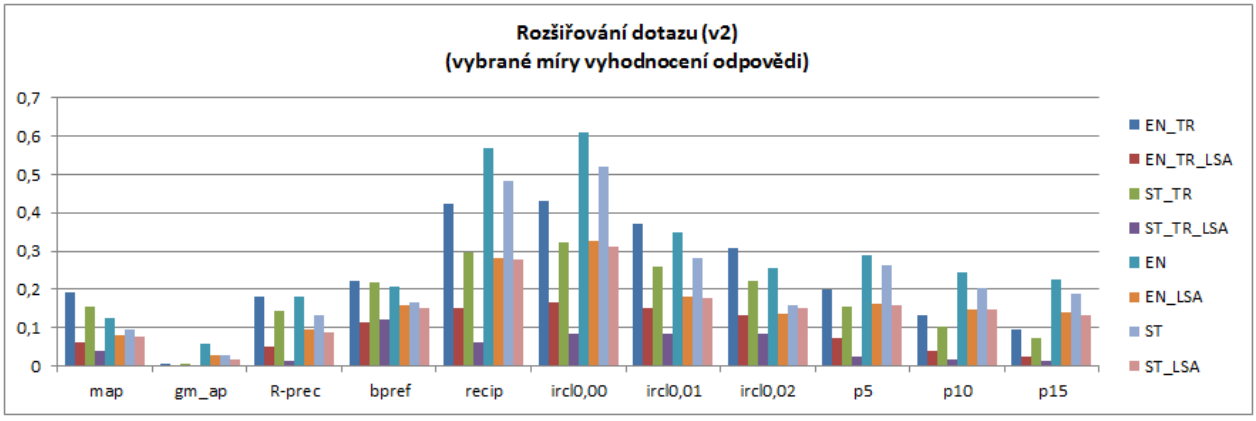
č.	A	SP	Sada	LE	LR	ret	rel	rel_ret	map	P	R	PS
1	EN		TR			9909	375	204	0,1914	0,0206	0,5440	
2	EN	LSA	TR	5	400	10000	375	210	0,0619	0,0210	0,5600	346
3	ST		TR			9734	375	174	0,1563	0,0179	0,4640	
4	ST	LSA	TR	5	400	9945	375	244	0,0393	0,0245	0,6507	367
5	EN		TST			10000	2063	757	0,1259	0,0757	0,3669	
6	EN	LSA	TST	5	400	9922	2063	826	0,0787	0,0832	0,4004	258
7	ST		TST			9978	2063	652	0,0939	0,0653	0,3160	
8	ST	LSA	TST	5	400	10000	2063	793	0,0772	0,0793	0,3844	309

Tabulka 5-2: Konfigurace a naměřené výsledky (v2).

R a *P* se zvýšily (v případě č.4 se *P* zvýšilo oproti baseline o 37%, *R* o 40%, v případě č. 8 *P* i *R* o 21%), ostatní míry TREC se snížily. Na trénovací sadě jsou patrnější rozdíly v mírách TREC.



Obrázek 5-3: Vliv rozšíření v2 na přesnost a úplnost (v2).



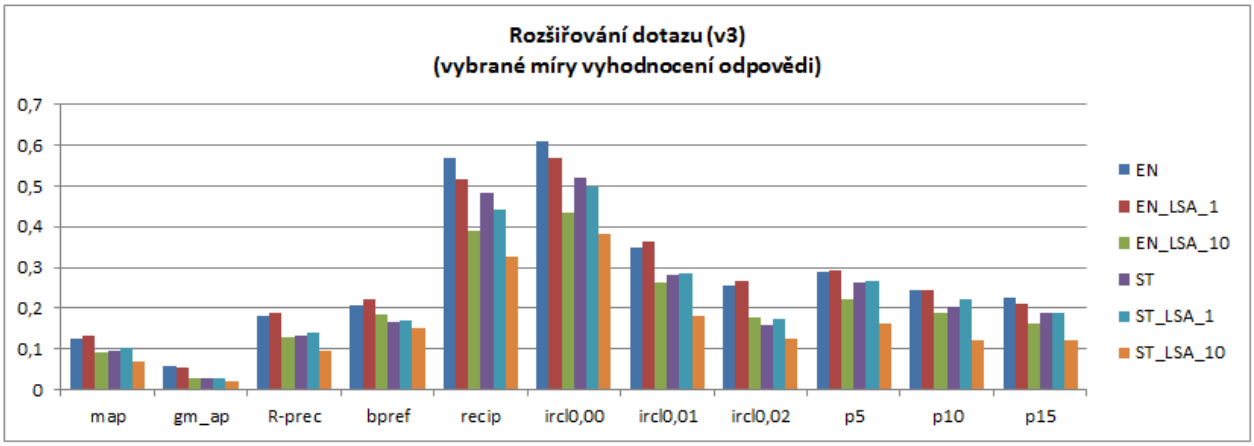
Obrázek 5-4: Vliv rozšíření v2 na vybrané míry TREC (v2).

Rozšiřování dotazu - varianta v3

Na obrázku 5-5 je vidět rozšíření typu v3. Obrázek ukazuje dvě varianty (rozšíření vzácného slova dotazu nejvýše o 1 a o 10 slov), které ilustrují vliv počtu slov, o která byl dotaz rozšířen, na míry související s přesností.

č.	A	SP	Sada	LE	LR	ret	rel	rel_ret	map	P	R	PS
1	EN		TST			10000	2063	757	0,1259	0,0757	0,3669	
2	EN	LSA	TST	1	1000	9854	2063	765	0,134	0,0776	0,3708	29
3	EN	LSA	TST	10	1000	9990	2063	740	0,0911	0,0741	0,3587	290
4	ST		TST			9978	2063	652	0,0939	0,0653	0,3160	
5	ST	LSA	TST	1	1000	10000	2063	738	0,1028	0,0738	0,3577	46
6	ST	LSA	TST	10	1000	10000	2063	727	0,0689	0,0727	0,3524	460

Tabulka 5-3: Konfigurace a naměřené výsledky (v3).



Obrázek 5-5: Vliv rozšíření v3 na vybrané míry TREC (v3).

Je vidět, že přidáním nejvýše jednoho slova ke každému slovu dotazu se zvýšila míra *map*. Tato míra zůstává relativně vysoká i po přidání nejvýše 10 slov ke každému vzácnému slovu dotazu.

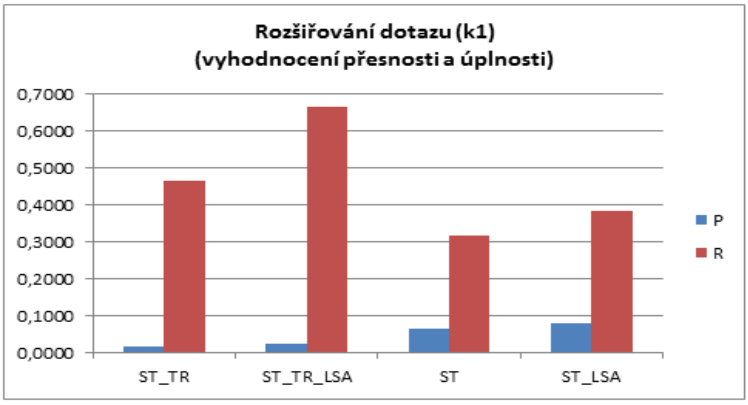
Rozšiřování dotazu - varianta k1

Na obrázcích 5-6 a 5-7 je vidět rozšíření typu k1.

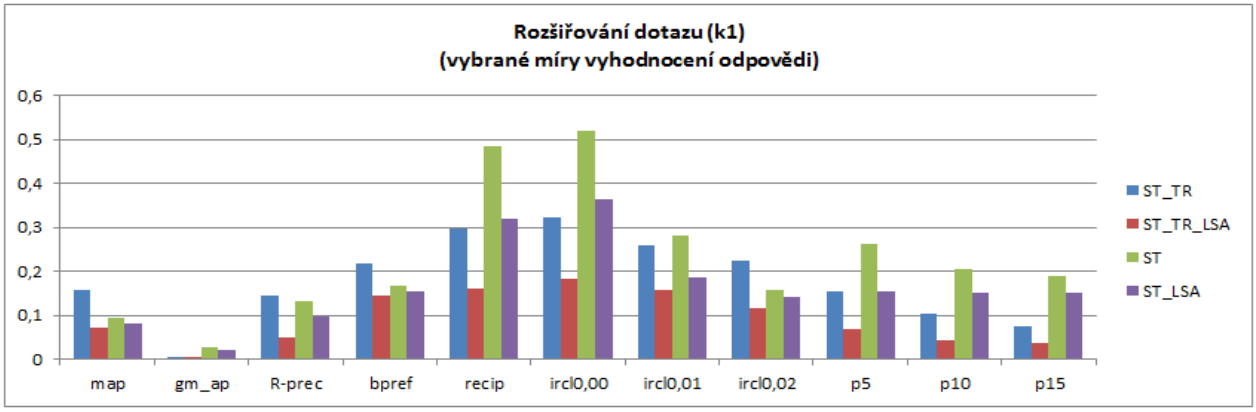
č.	A	SP	Sada	LE	LRQ	LR	ret	rel	rel_ret	map	P	R	PS
1	ST		TR		2000	1000	9734	375	174	0,1563	0,0179	0,4640	
2	ST	LSA	TR	5	2000	1000	9945	375	250	0,0732	0,0251	0,6667	278
3	ST		TST		2000	1000	9978	2063	652	0,0939	0,0653	0,3160	
4	ST	LSA	TST	5	2000	1000	10000	2063	789	0,0803	0,0789	0,3825	285

Tabulka 5-4: Konfigurace a naměřené výsledky (k1).

R a *P* se zvýšily (v případě č.2 se *P* zvýšilo o 40%, *R* o 43%, v případě č. 4 *P* i *R* o 21%).



Obrázek 5-6: Vliv rozšíření k1 na přesnost a úplnost (k1).



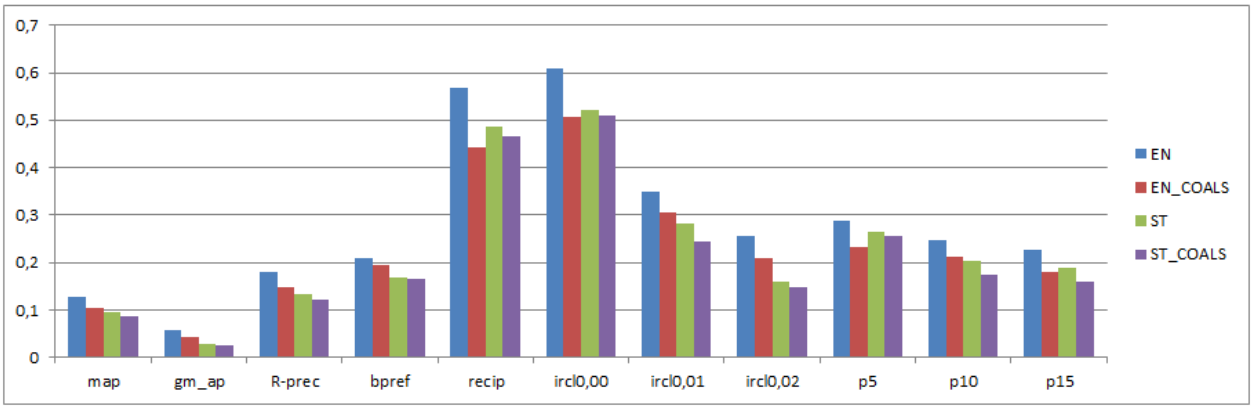
Obrázek 5-7: Vliv rozšíření k1 na vybrané míry TREC (k1).

Rozšiřování dotazu - varianta k2

Rozšíření typu k2 je vidět na obrázku 5-8.

č.	A	SP	Sada	LE	LO	LRQ	LR	SC	ret	rel	rel_ret	map	P	R	PS
1	EN		TST		2				10000	2063	757	0,1259	0,0757	0,3669	
2	EN	Coals	TST	50	2	7000	1500	0,1	9850	2063	761	0,1026	0,0773	0,3689	130
3	ST		TST		2				9978	2063	652	0,0939	0,0653	0,3160	
4	ST	Coals	TST	50	2	7000	1500	0,1	9978	2063	688	0,087	0,0690	0,3335	156

Tabulka 5-5: Konfigurace a naměřené výsledky (k2).



Obrázek 5-8: Vliv rozšíření k2 na vybrané míry TREC (k2).

Všechny typy rozšíření vedly ke zlepšení úplnosti i přesnosti, zlepšení dalších měřítech TREC (zejména *map*) bylo obtížné. Další odstavce se budou zabývat zvyšováním *map*.

5.2.2 Testování míry Map

Rozšiřování dotazu - varianta v4:

Rozšiřování dotazu o slova podobná více slovům dotazu nevede ke znatelnému zvýšení *P* a *R*. Tato varianta se ale zdá být odolná vůči ztrátě střední průměrné přesnosti (*map*). Ve statistickém souboru 200 konfigurací (EN, ST, LSA, Coals, počet *N* nejlepších slov [1, 2, 5, 10, 20, 50], spoluvýskyt se 2, 3 nebo všemi slovy, minimální kosinová vzdálenost 0, 0,1 a 0,2) zkoušených na trénovací sadě se *map* pohyboval na intevalu 0,1691 – 0,1914, *P* na intervalu 0,02 – 0,021, R na intervalu 0,52 – 0,54, *rel_ret* na intervalu 195 – 204, kde horní hranice intervalů odpovídala baseline a zároveň nejlepším testovaným konfiguracím. Počet přidaných slov byl až 90.

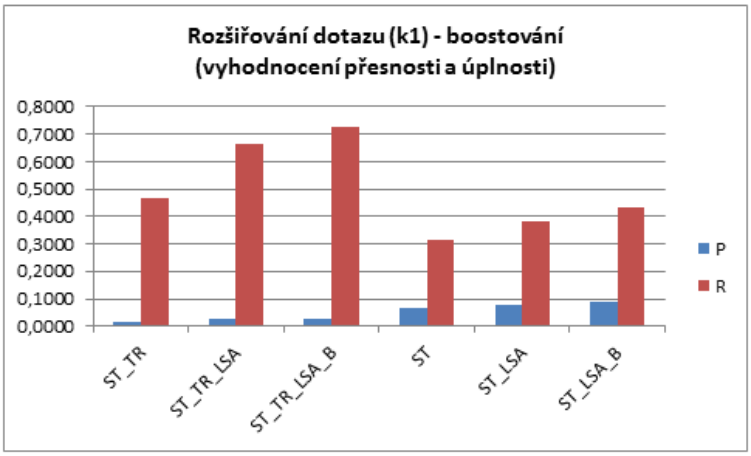
Rozšiřování dotazu – varianta k1 - boostování

Boostování bylo otestováno na variantách k1. Boost pro slova dotazu byl nastaven na hodnotu 8 a bylo zapnuto pokročilé boostování, které na základě IDF zvýhodňuje přidaná slova.

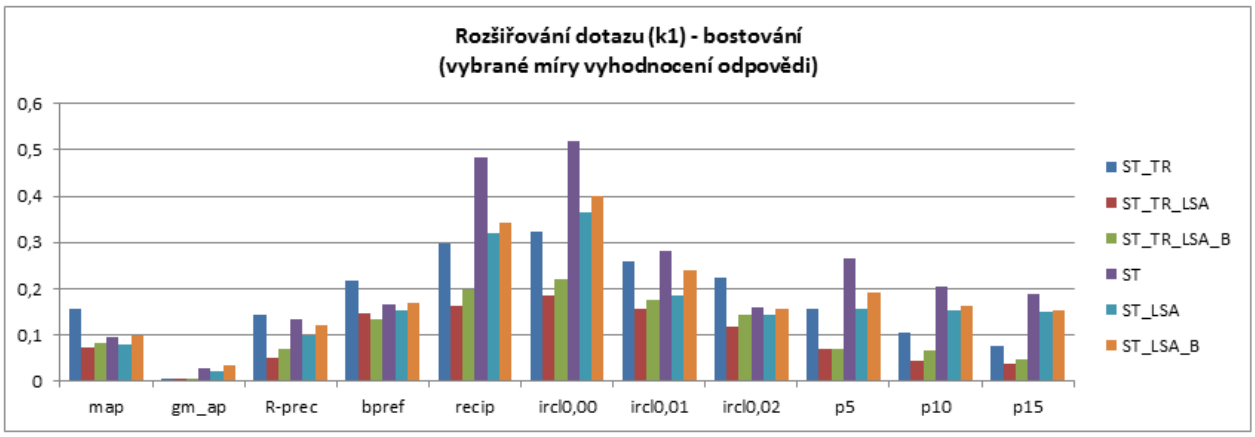
č.	A	SP	B	Sada	LE	LRQ	LR	ret	rel	rel_ret	map	P	R	PS
1	ST			TR				9734	375	174	0,1563	0,0179	0,4640	
2	ST	LSA		TR	5	2000	1000	9945	375	250	0,0732	0,0251	0,6667	278
3	ST	LSA	8	TR	5	2000	1000	9945	375	273	0,0838	0,0275	0,7280	278
4	ST			TST				9978	2063	652	0,0939	0,0653	0,3160	
5	ST	LSA		TST	5	2000	1000	10000	2063	789	0,0803	0,0789	0,3825	285
6	ST	LSA	8	TST	5	2000	1000	10000	2063	893	0,0977	0,0893	0,4329	285

Tabulka 5-6: Konfigurace a naměřené výsledky (k1) - boostování

Na obrázku 5-9 je vidět, že metoda boostování zvýšila *P* a *R* jak v trénovací, tak v testovací sadě. Z obrázku 5-10 lze vyčíst vzrůstající tendenci většiny měřítek TREC. V případě č.6 byla oproti č.4 zvýšena *map*.



Obrázek 5-9: Vliv boostování na přesnost a úplnost (k1) - boostování.



Obrázek 5-10: Vliv boostování na vybrané míry TREC (k1) - boostování.

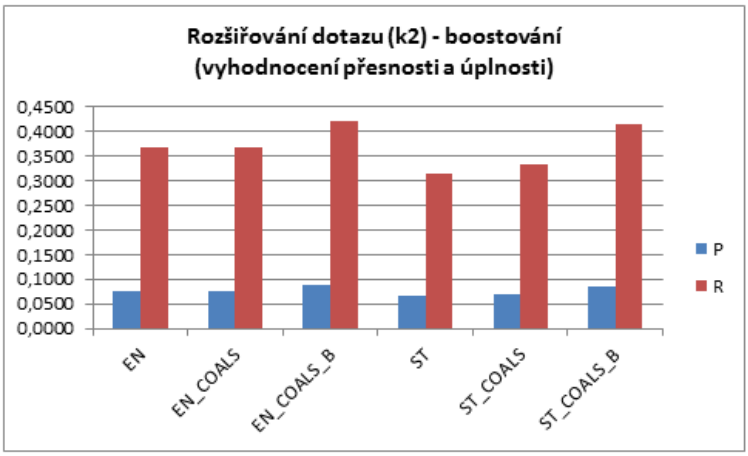
Rozšiřování dotazu – varianta k2 - boostování

Obrázky 5-11 a 5-12 znázorňují vliv boostování na rozšíření typu k2.

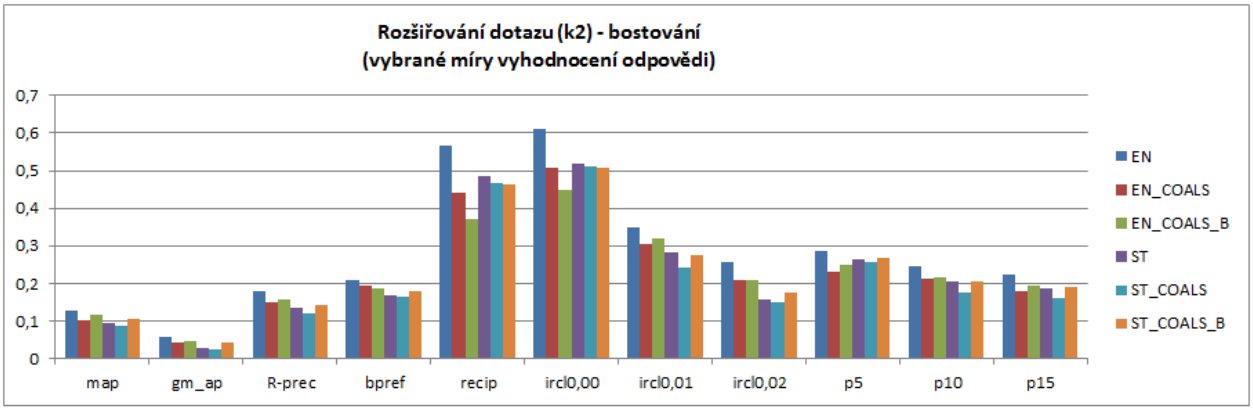
č.	A	SP	B	Sada	LE	LO	LRQ	LR	SC	ret	rel	rel_ret	map	P	R	PS
1	EN			TST						10000	2063	757	0,1259	0,0757	0,3669	
2	EN	Coals		TST	50	2	7000	1500	0,1	9850	2063	761	0,1026	0,0773	0,3689	130
3	EN	Coals	8	TST	50	2	7000	1500	0,1	9850	2063	872	0,1158	0,0885	0,4227	130
4	ST			TST						9978	2063	652	0,0939	0,0653	0,3160	
5	ST	Coals		TST	50	2	7000	1500	0,1	9978	2063	688	0,087	0,0690	0,3335	156
6	ST	Coals	8	TST	50	2	7000	1500	0,1	9978	2063	857	0,1062	0,0859	0,4154	156

Tabulka 5-7: Konfigurace a naměřené výsledky (k2) – boostování.

Na prvním obrázku je vidět výrazné zvýšení *R* i *P*. Z obrázku 5-12 lze vyčíst vzrůstající tendenci většiny měřítek vlivem boostování. V případě ST je původní *map* vylepšena o 12%.



Obrázek 5-11: Vliv boostování na přesnost a úplnost (k2) - boostování.



Obrázek 5-12: Vliv boostování na vybrané míry TREC (k2) - boostování.

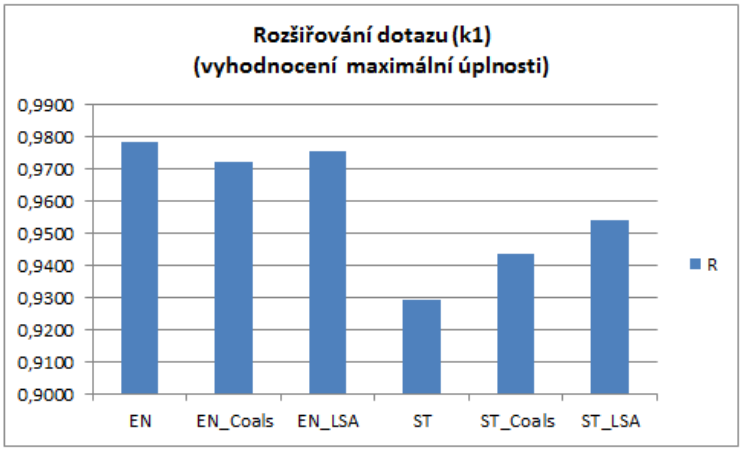
5.2.3 Testování maximální úplnosti (zvyšování Hits)

Pro testování maximální úplnosti jsem zvolil variantu k1, protože dosáhla nejvyšších zlepšení na *R*. K dosažení maximální úplnosti *R* byl nastaven maximální *hits*. V tabulce 5-13 je vidět, že rozšíření dotazu vede ke zvýšení počtu získaných dokumentů *ret* oproti baseline.

č.	A	SP	B	Sada	LE	LRQ	LR	ret	rel	rel_ret	map	P	R	PS
1	EN			TST				930999	2063	2019	0,1101	0,0022	0,9787	0
2	EN	Coals	8	TST	5	2000	1000	966239	2063	2006	0,0624	0,0021	0,9724	124
3	EN	LSA	8	TST	5	2000	1000	946431	2063	2013	0,0606	0,0021	0,9758	185
4	ST			TST				586091	2063	1917	0,0784	0,0033	0,9292	0
5	ST	Coals	8	TST	5	2000	1000	636385	2063	1947	0,0332	0,0031	0,9438	150
6	ST	LSA	8	TST	5	2000	1000	623003	2063	1968	0,0379	0,0032	0,9540	285

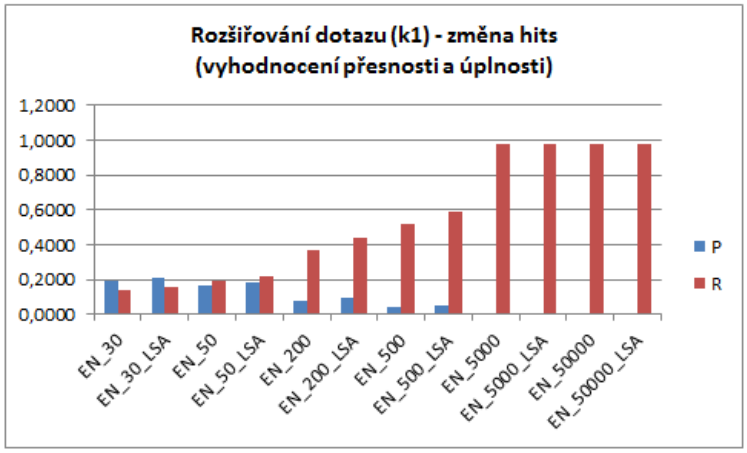
Tabulka 5-8: Konfigurace a naměřené výsledky (k1) – maximální úplnost.

Na obrázku 5-13 jsou znázorněny maximální *R* pro různé konfigurace.



Obrázek 5-13: Maximální úplnosti jednotlivých konfigurací (k1).

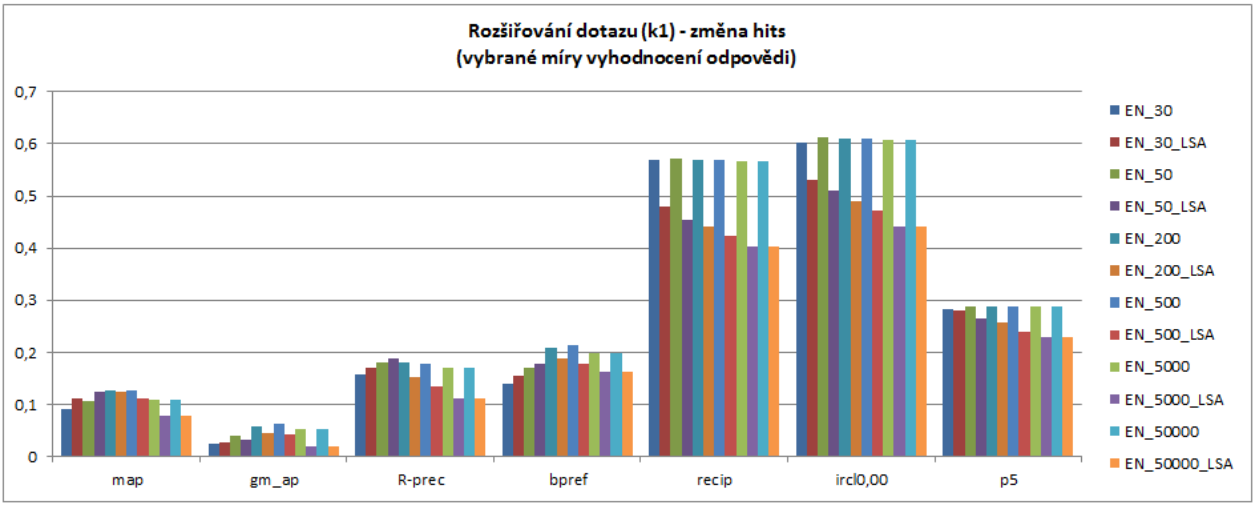
Nejvyšší *R* dosáhla kombinace č. 3, proto na této kombinaci testuji zvyšování *hits*. Na obrázku 5-14 je vidět vliv změny počtu získaných dokumentů k dotazu (*hits*) na *P* a *R*. Úplnost roste přímo úměrně s *hits*, přesnost se zvyšujícím se počtem získaných dokumentů (*ret*) klesá.



Obrázek 5-14: Vliv změny Hits na přesnost a úplnost (k1).

Zvyšování *hits* v případě rozšíření dotazu konverguje k úplnosti R = 1 rychleji než v případě baseline. Této úplnosti se ale v žádném z testovaných typů rozšíření nepodařilo dosáhnout.

Obrázek 5-15 znázorňuje vliv *hits* na ostatní míry TREC. S narůstajícím *hits* se snižují.



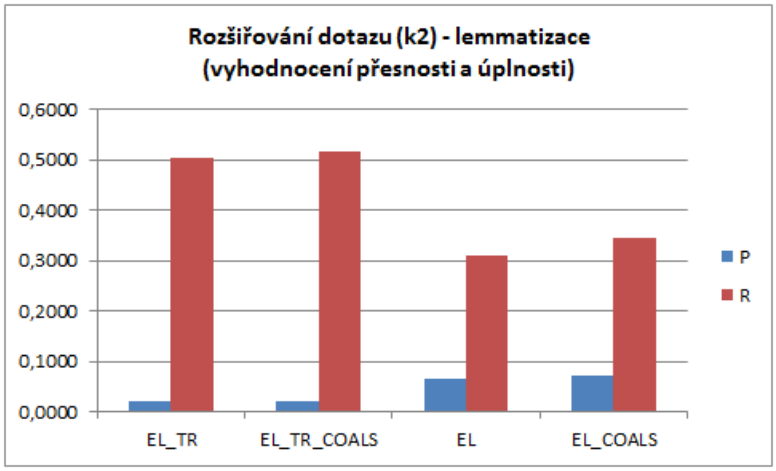
Obrázek 5-15: Vliv změny Hits na vybrané míry TREC (k1).

5.2.4 Testování Lemmatizátoru

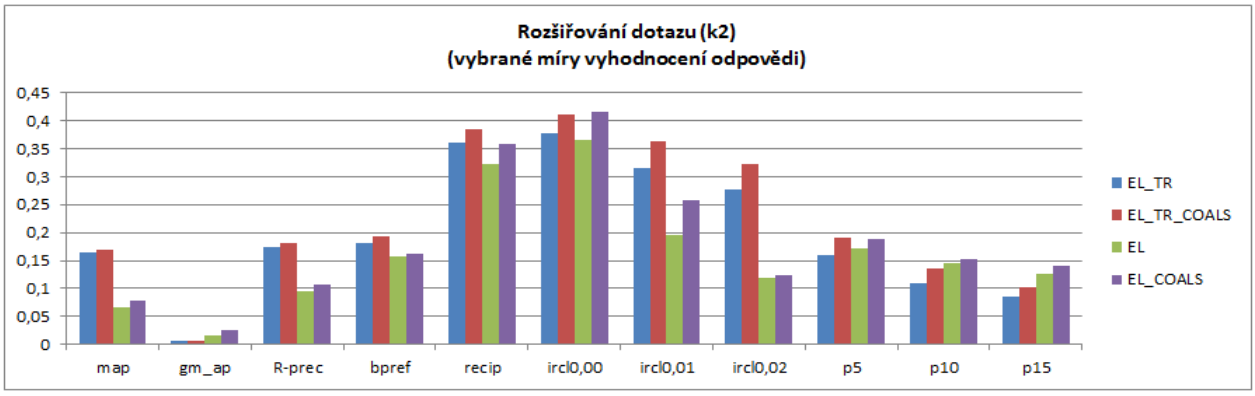
Na obrázku 5-17 je vidět, že i přes relativně velké množství slov, která byla přidána k trénovací i testovací sadě, *map* roste. Rostou i ostatní míry TREC, *P* a *R* ale méně než v případě ST a EN (viz obrázek 5-16).

č.	A	SP	B	Sada	LE	LO	LRQ	LR	SC	ret	rel	rel_ret	map	P	R	PS
1	EL			TR						9335	365	184	0,165	0,0197	0,5041	
2	EL	Coals	8	TR	20	2	50000	3000	0,1	9335	365	189	0,1695	0,0202	0,5178	298
3	EL			TST						9808	2063	638	0,0666	0,0650	0,3093	
4	EL	Colas	8	TST	20	2	50000	3000	0,1	9808	2063	715	0,0771	0,0729	0,3466	254

Tabulka 5-9: Konfigurace a naměřené výsledky (k2) – lemmatizace.



Obrázek 5-16: Vliv rozšíření pomocí lemmatizace na přesnost a úplnost (k2).



Obrázek 5-17: Vliv rozšíření pomocí lemmatizace na vybrané míry TREC (k2).

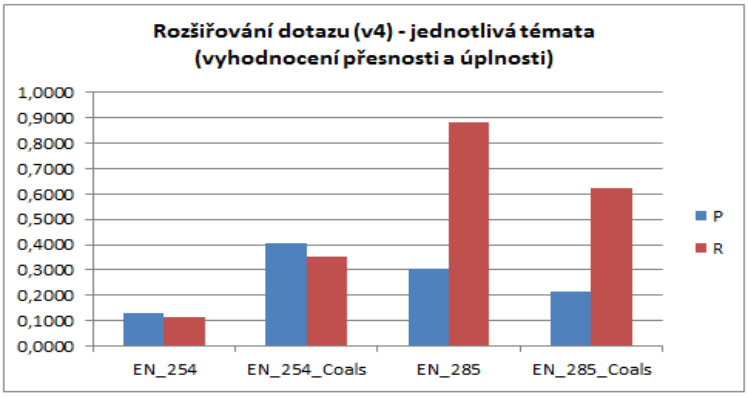
5.2.5 Testování jednotlivých témat

Rozšiřování dotazu - varianta v4:

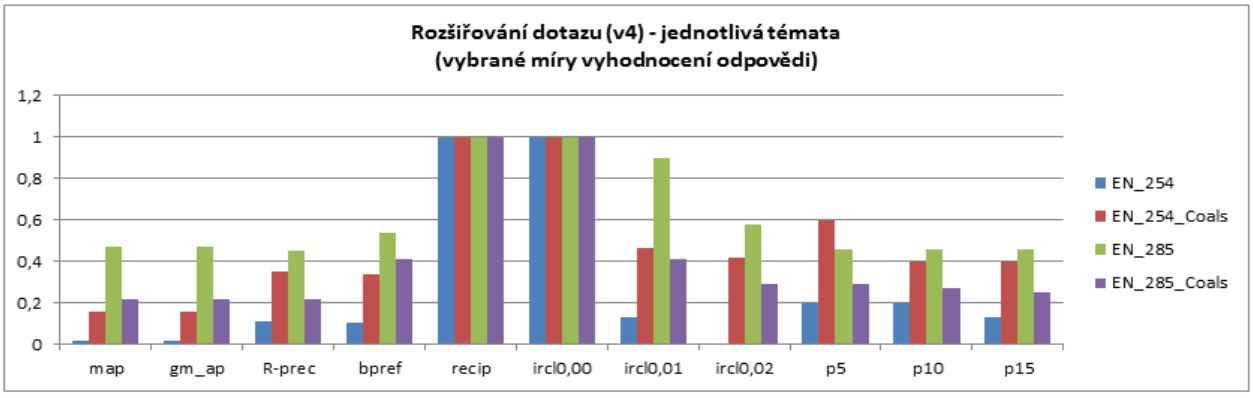
Ná obrázcích 5-18 a 5-19 je vidět, že rozšíření dotazu č. 254 (*Earthquake Damage*) vedlo k výraznému zvýšení všech měřítek. Rozšíření dotazu č. 285 (*Anti-Abortion Movements*) vedlo naopak k jejich výraznému snížení.

č.	A	SP	Téma	LE	LO	SC	ret	rel	rel_ret	map	P	R	PS
1	EN		254				200	229	26	0,0179	0,0206	0,5440	
2	EN	Coals	254	150	2	0,1	200	229	81	0,1557	0,0210	0,5600	82
3	EN		285				200	69	61	0,4718	0,0179	0,4640	
4	EN	Coals	285	150	2	0,1	200	69	43	0,2173	0,0245	0,6507	79

Tabulka 5-10: Konfigurace a naměřené výsledky (v4) – jednotlivá témata.



Obrázek 5-18: Vliv rozšiřování vybraných dotazů na přesnost a úplnost (v4).



Obrázek 5-19: Vliv rozšiřování vybraných dotazů na vybrané míry TREC (v4).

6 ZÁVĚR

Myslím si, že hlavní cíle diplomové práce byly splněny. Aplikace je uživatelsky přívětivá, rozšiřitelná a umožňuje implementovat nové typy rozšíření dotazů. Navržené typy rozšíření dotazu byly otestovány alespoň na některých konfiguracích a výsledky prokázaly, že pomocí sémantických prostorů lze do určité míry (a za cenu prodloužení odezvy) zlepšit kvalitu odpovědi systému na zadaný dotaz.

6.1 Zhodnocení dosažených výsledků

Všemi navrženými typy rozšiřování dotazu bylo možné zvýšit přesnost a úplnost. Rozšíření založená na vzácnosti slov vedla spíše ke zvýšení *P* a *R*, rozšíření založená na podobnosti přidaného slova s více slovy dotazy vedla ke zvýšení *map*, případně dalších měřítek souvisejících s přesností. Kombinování jednotlivých typů rozšíření dotazu vedlo k dalšímu zlepšení naměřených výsledků.

Map se ve většině případech nepodařilo vylepšit. Tato míra byla zvýšena až metodou pokročilého boostování, která vedla ke znatelnému zlepšení všech měřítek. Boostování v některých případech zlepšilo výsledky i oproti baseline. Míry *map* a *P* jsou přímo ovlivněny počtem získaných dokumentů (čím větší *hits*, tím větší *ret*). *Map* dosahovala nejlepších výsledků v okolí hits 200 (viz obrázek 5-15).

Jednoduché typy rozšiřování dotazu dosahovaly lepších výsledků, když byl použit ST a LSA; kombinované varianty dosahovaly lepších výsledků spíše s EN a Coals. Tato skutečnost ale zdaleka neplatila obecně, byla značně závislá na konfiguraci daného typu rozšíření dotazu.

Na kvalitu odpovědi má vliv také počet slov, která jsou k dotazu přidána. Obecně platilo, že čím více slov, tím větší *P* a *R*, ale nižší *map*. *P* roste proto, že počet získaných dokumentů je omezen *hits*. *Map* klesá, protože na rozdíl od *P* vyjadřuje také pozici, na které byl relevantní dokument nalezen. Toto ale neplatilo při použití lemmatizátoru, který *map* mírně zvyšoval i při přidání 300 slov k 50 tématům sady.

Očekávalo se, že použití normalizačních metod předzpracování textu pro vytvoření SP a vyhledávání v indexu (viz oddíl 2.4) povede ke zvýšení úplnosti a snížení přesnosti (předpokladem je neomezený *hits*) [2]. Na neomezeném *hits* byl předpoklad potvrzen, úplnost roste s každým dalším nalezeným relevantním dokumentem, přesnost s každým dalším získaným dokumentem klesá. Použití lemmatizátoru ale vedlo k pomalejšímu růstu *R* i pomalejšímu snižování *P*.

Obecně lze také říci, že zlepšení výsledků na trénovací sadě vedlo ve většině případů i k podobnému zlepšení výsledků na testovací sadě. To znamená, že se jednotlivé míry rozšiřováním dotazu nemění nahodile.

Vyhledávací systém umožňuje využívat i další typy SP (připraven je HAL), je implementováno měření vzdálenosti dvou vektorů pomocí Euklidovské vzdálenosti a Pearsonovy korelace a je implementována tokenizace pro zpracování textu (WS). Tyto možnosti z časových důvodů nebylo možné vyzkoušet.

6.2 Další možnosti rozšíření aplikace

Z průběhu práce vyplynuly i další možnosti rozšiřování dotazu. Bylo by například zajímavé hledat taková slova, která se vyskytují ve více různých SP. Další zpřesnění odpovědi by mohlo být dosaženo vážením přidaných slov nejen podle IDF ale i podle jejich koeficientu podobnosti. Dále by se mohla vybírat slova, související s více slovy dotazu taková, že alespoň v jednom případě jejich koeficient podobnosti přesáhne stanovený limit (v práci je implementována pouze možnost, že je limit dosažen ve všech případech).

Metriky odvozené od přesnosti (např. *map*) se ve většině případech nepodařilo vylepšit. To je zřejmě způsobeno tím, že byly implementovány pouze normalizační metody zpracování textu, které snižují přesnost [2]. Bylo by proto zajímavé hlouběji prozkoumat možnosti knihovny Stanford Postagger, která kromě lemmatizace umí slova také anotovat. Kombinace anotace a normalizace textu by mohla vést k dalšímu zlepšení měřítek souvisejících s přesností i úplností.

Vzhledem k tomu, že stejné konfigurace rozšíření dotazu se mohou v jednotlivých případech projevit naprosto odlišně (viz odstavec 5.2.5), dalšího zlepšení by bylo možné dosáhnout rozpoznáním dotazů, které se budou rozšiřovat, a dotazů, které se rozšiřovat nebudou. Případně rozpoznat, jaké rozšíření pro daný dotaz zvolit.

Tato rozšíření z časových důvodů nebyla implementována. Aplikace je ale navržena tak, že by jejich implementace neměla být příliš náročná. Aplikaci lze dále snadno rozšířit o další SP, analyzátory, míry sémantické podobnosti apod. (viz oddíl 4.7).

Samostatnou úlohou, která by úlohu rozšiřování dotazu mohla značně ovlivnit, by bylo experimentování s vytvářením různých typů SP a s různými parametry pro vytvoření SP. Testovat jaká další slova kromě stopslov vynechat, jak velké nastavit okénko či dimenzi pro SVD rozklad. Tomuto problému jsem se v této práci nevěnoval vůbec.

Také by bylo vhodné zamyslet se nad zvýšením rychlosti rozšiřování dotazu, protože rozšíření jednoho dotazu zabírá řádově vteřiny a přitom korpus není nijak zvlášť velký. Rozšiřování dotazu nad větším korpusem by mohlo trvat poměrně déle.

Přehled zkratek a pojmů

Baseline – základní stav (vyhledávání bez rozšiřování dotazu), oproti kterému se testuje zlepšení rozšiřováním dotazu.
CLEF (*Cross-Language Evaluation Forum*) – organizace, která se zabývá výzkumem, rozvojem a inovací přístupu k textovým informacím. Hlavní důraz klade na multijazyčné informace.
Coals (*Correlated Occurrence Analogue to Lexical Semantic*) – algoritmus pro vytváření SP.
DIS (*Dokumentografický informační systém*) – prostředek sloužící k nalezení dokumentů psaných v přirozeném jazyce z nějaké definované oblasti problematiky.
Dokument – jazykově zpracovaný dokument připravený k indexaci (obsahuje Pole).
EL – zkratka pro EnglishLemmaAnalyzer (součástí knihovny Stanford Postagger).
EN – zkratka pro EnglishAnalyzer (součástí knihoven Apache Lucene).
HAL (*Hyperspace Analogue to Language*) – algoritmus pro vytváření SP.
Index – seznam výskytů všech slov korpusu v jednotlivých dokumentech.
Kontext – označuje věty, odstavce či jiný úsek textu, které slovo, výraz nebo spojení obklopují. Kontextem je význam sdělení a jeho vztah k ostatním částem sdělení.
Korpus – kolekce textů v přirozeném jazyce, který slouží pro lingvistický výzkum v daném jazyce anebo například jako základ pro tvorbu slovníků, ontologií, překladačů apod.
LSA (*Latent Semantic Analysis*) – algoritmus pro vytváření SP.
Odpověď – seznam takových dokumentů vrácených k danému dotazu, které vyhledávací systém považuje za relevantní.
Pole (*Field*) – každý indexovaný dokument se skládá z polí (např. text, popisek, datum apod.).
Přesnost – pravděpodobnost, že získaný dokument je relevantní.
Relevance odpovědi – relevance všech dokumentů získaných na základě vyhledávacího dotazu.
S-Space – balík knihoven S-Space obsahuje algoritmy pro vytváření SP.
SGML (*Standard Generalized Markup Language*) – značkovací jazyk, předchůdce XML.
Slovo – základní jednotka přirozeného jazyka nesoucí význam.
SVD *(Singular Value Decomposition)* – algoritmus pro redukci matice (snížení její dimenze).
ST – zkratka pro StandardAnalyzer (součástí knihoven Apache Lucene).
T-D matice – zkratka pro matici typu Term-Document.
Term – libovolný zpracovávaný řetězec (slovo, sousloví, fráze a jiné řetězce).
Testovací sada – sada témat, na které se ověřují konfigurace získané na trénovacích datech.
TF-IDF (*Term Frequency - Inverse Document Frequency*) – algoritmus pro převažování prvků matice výskytů.
TREC (*Text Retrieval Conference*) – série seminářů, zabývající se různými oblastmi výzkumu na poli vyhledávání informací. V textu je zmiňována aplikace TREC, která slouží k vyhodnocení odpovědi na vyhledávací dotaz (součástí korpusu CLEF).
Trénovací sada – sada témat korpusu, na této sadě se provádí experimenty.
Úplnost – pravděpodobnost, že bude získán relevantní dokument.
SP (*Vector Space Model*) – sémantický prostor.
W-C matice – zkratka pro matici typu Word-Context.
WS – zkratka pro WhitespaceAnalyzer (součástí knihoven Apache Lucene).

Reference

[1] Krčmář, L. (2011). *Aplikace sémantických prostorů v úloze rozšiřování dotazu*. Odborná práce. Plzeň. ZČU, FAV.

[2] Turney, P. D., & Pantel, P. (2010). *From Frequency to Meaning: Vector Space Models of Semantics*. Journal of Artificial Intelligence Research 37, 141-188.

[3] Krömer, P. *Dokumentografické informační systémy*. Skripta. Brno, VSB

[4] Osgood, C., Suci, G., & Tannenbaum, P. (1957). *The measurement of meaning.* University of Illinois Press.

[5] Sahlgren, M. (2006). *The Word-Space Model*. Stockholm University.

[6] Firth, J. R. (1951). *Papers in Linguistics.* Oxford University Press.

[7] Budanitsky, A., & Hirst, G. (2006). *Evaluating wordnet-based measures of semantic distance*. Computational Linguistics. 32(1), 13–47.

[8] Salem, T. S. (2012). *Automatické zodpovídání dotazů založené na sumarizaci textů*. Diplomová práce. Plzeň. ZČU, FAV.

[9] Rohde, D. T., Gonnerman, L., & Plaut, D. (2004). *An improved method for deriving word meaning from lexical co-occurrence*. Cognitive Science.

[10] Landauer, T., & Dumais, S. (1997). *The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge.* American Psychological Associations, Inc.

[11] Lund, K., & Burgess, C. (1996). *Producing high-dimensional semantic spaces from lexical co-occurrence*. Behavior Research Methods, 203-208.

[12] Jurgens, D., & Stevens, K. (2010). *The S-Space Package: An Open Source Package for Word Space Models*. Proceedings of the ACL 2010 System Demonstrations 30-35.

[13] Matejovič, P. (2011). *Aplikace pro řízení a sledování výroby*. Bakalářská práce. Plzeň. ZČU.

[14] Krátký, P. (2007). *Vyhledávání v českých dokumentech nad nativním XML úložištěm*. Diplomová práce. Brno. MUNI, FI.

[15] Hatcher, E., Gospodnetic, O., McCandless, M. (2009). *Lucene in Action*. Second Edition. Manning Publications.

PŘÍLOHA A: KONFIGURACE APLIKACE

A.1 Prerekvizity

Pro správný chod aplikace je třeba mít v systému nainstalovány:

- Java JDK[81] v6 a vyšší,
- Apache Tomcat v7[82],
- MinGW[83] pro aplikaci TREC,
- Eclipse IDE for Java EE[84] (pro další vývoj aplikace).

Všechny prerekvizity jsou součástí přiloženého Blu-ray.

A.2 Instalace a nastavení systému

Aplikace je dodána jako balík `war`, který je nutné umístit do `%TOMCAT_PATH%\webapps\`, kde `%TOMCAT_PATH%` znamená cestu k nainstalovanému Tomcat serveru. Tomcat server lze přidat přímo do vývojového prostředí (záložka Servery v Eclipse). Pracuje v JVM, která má implicitně nastaveno 96MB pracovní paměti, proto je třeba na serveru (případně na serveru integrovaném v Eclipse) nastavit výchozí argumenty pro provoz aplikace, např.:

`-Xmx1280m`, pro 32bit Javu
`-Xmx8g -d64`, pro 64bit Javu

Do systémové cesty operačního systému musí být přidáno `%MINGW_PATH%\bin; %MINGW_PATH%\msys\1.0\bin;` kde `%MINGW_PATH%` značí cestu k adresáři MinGW (např. `c:\MinGW`).

A.3 Struktura dat

V `dp.properties`, který je součástí dodaného `war` balíčku) musí být nastavena cesta ke kořenovému adresáři s daty (např. `c:\\DP`). Následující struktura kořenového adresáře musí být zachována[85]:

`\cmd`	Obsahuje dávky, které jsou spouštěny z aplikace (přípona BAT).
`\conf`	Obsahuje konfigurační soubory, seznamy analyzátorů, SP apod. (CONF).
`\data`	Obsahuje XML soubory s články pro vytvoření indexu.
`\index`	Obsahuje indexy rozdělené podadresáři podle typů analyzátorů (ST, EN, WS).
`\lemma`	Obsahuje slovníky pro lemmatizaci.
`\log`	Obsahuje LOG soubory pro každou vyhodnocenou odpověď na vyhledávací dotaz.
`\sspace`	Obsahuje jednotlivé SP (přípona SSPACE).

[81] http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
[82] http://tomcat.apache.org/
[83] Kompilátor jazyka C++, http://sourceforge.net/projects/mingw/files/.
[84] http://www.eclipse.org/
[85] Měnit ji lze definicí ve třídě Settings.

`\trec`	Obsahuje aplikaci `TREC.exe` a soubory pro vyhodnocení relevance (`qrels`). Odtud aplikace načítá dávkový soubor `batch.csv` a sem také ukládá výstup `results.csv`.
`config.ini`	Konfigurační soubor vstupních parametrů aplikace.
`activity.log`	Přírůstkový log událostí v aplikaci.

Všechna potřebná data jsou součástí přiloženého Blu-ray.

A.4 Konfigurace

Vstupní konfigurace aplikace se provádí v souborech:

- `dp.properties` – je součástí projektu a definuje kořenový adresář s uživatelskými daty.
- `config.ini` – pro implicitní nastavení parametrů rozšiřování dotazu ve tvaru dvojic `<parametr>=<hodnota>`.

Parametr	Typ	Vysvětlení
`globalAnalyzer`	`int`	Analyzátor (`analyzer.conf`).
`parseType`	`int`	Typ parsingu (`parser.conf`).
`limit`	`int`	Maximální počet vyhledaných dokumentů.
`expansionType`	`int`	Typ rozšíření dotazu (`expansion.conf`).
`sspace`	`String`	SP (`sspace.conf`).
`similarityType`	`int`	Typ podobnosti (`similarity.conf`).
`similarityCoef`	`float`	Koeficient podobnosti `<0;1>` (zahrne slova s vyšší podobností).
`limitExpansion`	`int`	Maximální počet nalezených nejpodobnějších slov.
`limitOccurence`	`int`	Počet slov dotazu, se kterými slovo musí souviset (4 - všechny).
`limitRareWordQuery`	`int`	Definice vzácného slova dotazu (maximální IDF).
`limitRareWord`	`int`	Definice vzácného slova (maximální IDF).
`boost`	`float`	Koeficient boostu původního dotazu.
`advancedBoost`	`String`	Boostování slov podle jejich IDF (`true`, `false`).
`trainSet`	`String`	Trénovací nebo testovací sada témat (`true`, `false`).
`enableBoost`	`String`	Zapnutí boostovaní (`true`, `false`).
`limitExclusion`	`int`	Horní hranice výskytu málo častých slov (pro vytvoření exclusion listu pro vytvoření SP).
`searchType`	`int`	Předdefinované typy vyhledávání (`search.conf`).

Dojde-li ke změně kořenového adresáře, musí se patřičně změnit i cesty v dávkových souborech v adresáři DP\cmd. Adresář DP včetně dat je součástí přiloženého Blu-ray.

A.5 Spuštění aplikace

Program lze spustit jako webové rozhraní s úplnou funkcionalitou:

`http://<IP nebo FQDN>:8080/DP/` (např. `http://localhost:8080/DP/`),

nebo jako samostatnou JAR knihovnu, která umožňuje pouze dávkové zpracování jednotlivých konfigurací rozšiřování dotazu s výstupem do CSV souboru:

`java.exe -d64 -Xmx3g -jar dp.jar [-v]`,

kde volitelný parametr `-v` (`-verbose`) znamená, že budou na výstup vypisovány všechny runtime informace. Takto spuštěná aplikace ke svému běhu používá soubor `\trec\batch.csv`.

JAR knihovna nepodporuje lemmatizaci. Z časových důvodů a z důvodu chyby v metodě `edu.stanford.nlp.io.IOUtils.findStreamInClasspathOrFileSystem()` ji nebylo možné v příkazové řádce zprovoznit.

JAR knihovna je součástí přiloženého Blu-ray.

A.6 Vytváření dávek

Dávky hromadného zpracování testovacích sad se definují prostřednictvím souboru `\DP\trec\batch.csv`, který je ve tvaru (jednotlivé řádky znamenají jednu konfiguraci rozšíření sady dotazů, sloupce korespondují s parametry rozšiřování dotazu - viz oddíl A.4):

```
analyzer; parser; hits; expansion type; sspace; similarity; similarity coef; limit expansion; limit occurrence; limit rare query; limit rare; boost; boost advanced; test set
```

A	P	H	ET	M	S	SC	LE	LO	LRQ	LR	B	BA	T
1	2	200	0	0	0	0.0	0	0	0	0	0	false	true
1	2	200	2	ST_coals.sspace	0	0.0	10	0	0	0	0	false	true
1	2	200	2	ST_coals.sspace	0	0.1	10	0	0	0	0	false	true
1	2	200	7	ST_coals.sspace	0	0.2	10	2	2000	500	0	false	true
0	2	500	7	EN_coals.sspace	0	0.1	20	2	2000	750	0	false	true
0	2	500	7	EN_coals.sspace	0	0.1	20	2	2000	1000	2	false	true
0	2	500	7	EN_lsa.sspace	0	0.0	20	2	2000	1500	2	false	true
0	2	500	7	EN_lsa.sspace	0	0.0	10	2	2000	3000	2	true	true
2	2	200	7	WS_coals.sspace	0	0.0	50	2	5000	300	2	true	true
2	2	200	7	WS_lsa.sspace	0	0.2	50	2	5000	400	2	true	false

Vyhodnocení odpovědi pro jednotlivé konfigurace dávky, jsou uloženy do souboru `\DP\trec\results.csv`. První sloupec jednoznačně identifikuje výsledek[86], další sloupce obsahují výstupy z aplikace TREC a v posledních sloupcích je uložena původní konfigurace (z `batch.csv`).

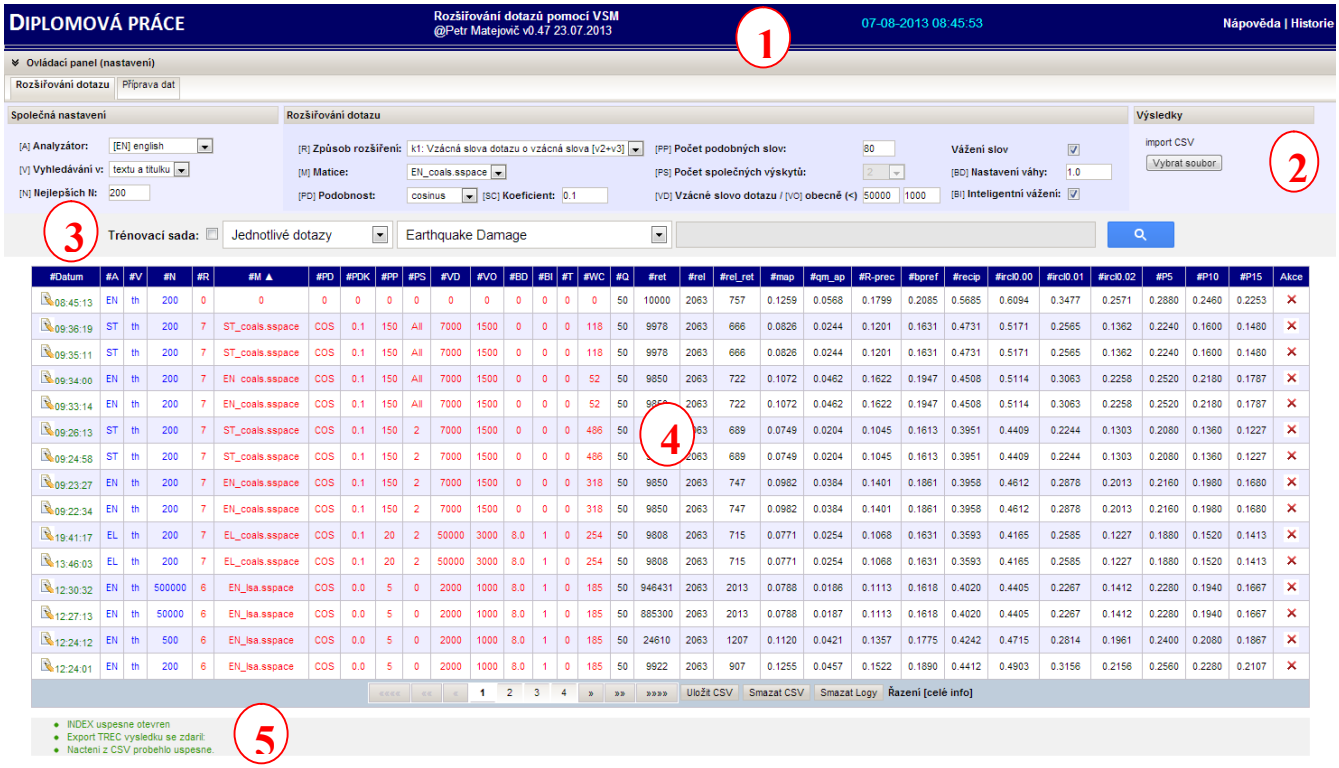
[86] Pomocí tohoto identifikátoru lze pak v aplikaci zobrazit příslušný LOG soubor.

PŘÍLOHA B: UŽIVATELSKÁ DOKUMENTACE

B.1 Popis aplikace

Hlavní obrazovka je rozdělena do pěti hlavních částí (obrázek A-1):

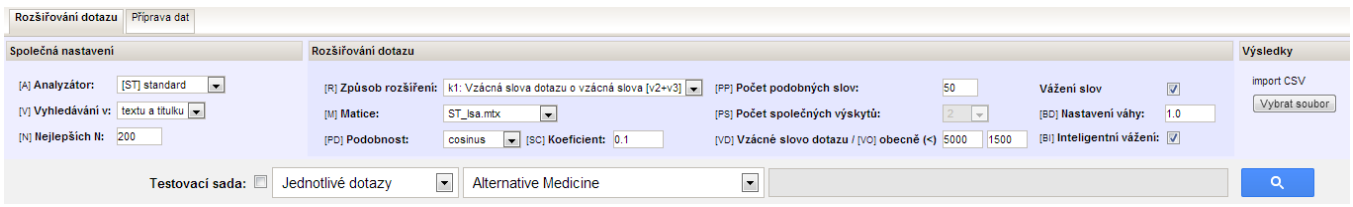
1) *Hlavička* obsahuje základní informace o aplikaci.
2) *Ovládací panel* obsahuje záložky pro rozšiřování dotazu (oddíl B.2) a přípravu dat (B.3).
3) *Vyhledávací panel* pro obsluhu samotného vyhledávání.
4) *Tabulku výsledků* vyhledávání (statistiky nebo nalezené dokumenty).
5) *Stavový panel*, kde jsou zobrazována hlášení o stavu požadavku.



Obrázek B-1: Hlavní obrazovka. Tabulka výsledků je rozlišena barevně, zelená pro datum (datum je současně jednoznačným identifikátorem záznamu s výsledky), modrá pro parametry vyhledávání, červená pro parametry rozšíření dotazu a černá pro výsledky z aplikace TREC.

B.2 Rozhraní pro vyhledávání a rozšiřování dotazu

Záložka *rozšiřování dotazu* uživateli umožňuje nastavit konfiguraci (odstavec A.4) pro rozšíření dotazu (obrázek B-2). Výběr analyzátoru ovlivňuje výběr indexu, nad kterým budou slova dotazu vyhledávána, a také výběr SP; způsob rozšíření dotazu má vliv na další pole formuláře (související pole budou aktivní, nepotřebná vypnutá – zešedlá).



Obrázek B-2: Nastavení rozšíření dotazu.

V pravé části formuláře je odkaz pro otevření a zobrazení souboru `results.csv`, tlačítkem vybrat soubor lze vybrat další uložené soubory ve stejném formátu jako `results.csv` z adresáře `\DP\trec\results\`.

B.3 Rozhraní pro přípravu dat

Záložka *příprava dat* poskytuje funkce pro zpracování XML souborů[87]:

- Vytvoření validních `xml` souborů s příponou REPAIRED.
- Přejmenování `repaired` na XML.
- Vymazání obsahu adresáře `\DP\data`.

Dále pro práci s indexem[88]:

- Smazání původního indexu.
- Vytvoření nového indexu z připravených XML souborů.

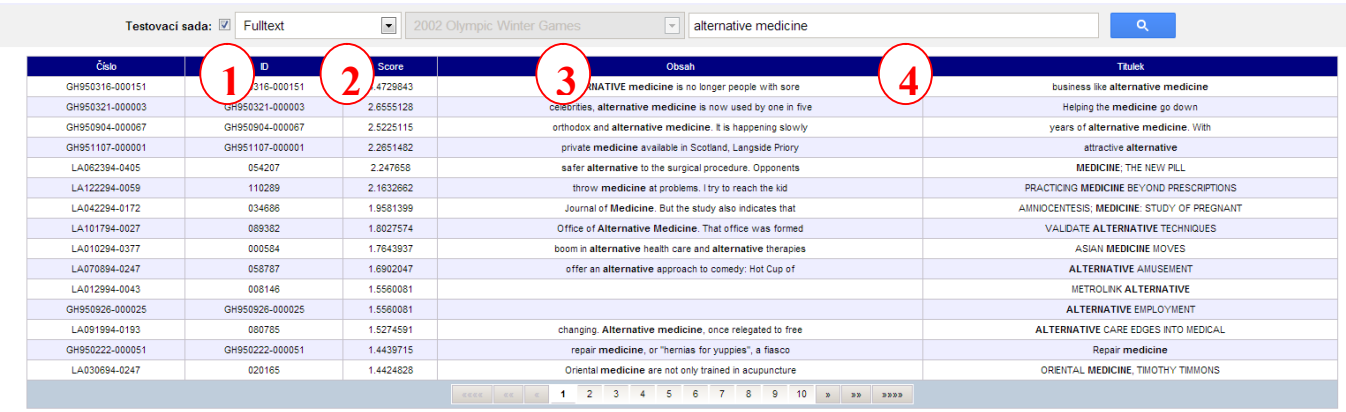
A vytvoření seznamu četností slov v indexu (IDF) a seznamu výjimek (slov, která budou vyloučena při vytváření SP).

B.4 Vyhledání dotazu a zobrazení odpovědi

Vyhledávací panel umožňuje vybrat mezi trénovací a testovací sadou témat (checkbox **1**) a mezi způsoby vyhledávání (**2**):

- Vyhledání předdefinovaného tématu (**3**).
- Vyhodnocení celé sady témat.
- Spuštění předdefinované dávky (`batch.csv`).
- Vyhledání nenalezených dokumentů k předdefinovanému tématu.
- Fulltextové vyhledávání libovolného dotazu (**4**).

Výstupem prvních tří způsobů vyhledávání je souhrnná tabulka (obrázek B-1) s možností zobrazení podrobných informací (kliknutím na ikonu vlevo) a smazání řádku (ikona vpravo), jinak je zobrazen seznam nalezených článků s možností náhledu na článek (kliknutím na jeho obsah).



Obrázek B-4: Zobrazení nalezených dokumentů.

[87] Soubory musí být umístěny v adresáři DP\data.

[88] Index souvisí s analyzátorem, na základě výběru analyzátoru je vybrán i příslušný index, se kterým se pracuje.

PŘÍLOHA C: OBSAH PŘILOŽENÉHO BLU-RAY

Přiložené Blu-ray obsahuje:

- zdrojové kódy projektu,
- všechny potřebné knihovny (soubory s příponou JAR),
- kompletní projekt vyexportovaný do souboru WAR (DP.WAR),
- spustitelný JAR soubor projektu (DP.JAR),
- instalační soubor Apache Tomcat v7,
- instalační soubor Java JDK v7 64bit,
- zabalený MinGW,
- Eclipse for Java EE (indigo) 64bit,
- všechna potřebná data (SP, XML, indexy, slovníky pro lemmatizaci atd.),
- zkompilovaná verze aplikace TREC a zdrojové soubory,
- dokumentaci ve formátu DOC i PDF,
- ukázkové soubory `batch.csv` a `results.csv`.