TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

DIPLOMOVÁ PRÁCE

2008

Jan Pleva

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N 2612 – Elektrotechnika a informatika

Studijní obor: 1802T007 – Informační technologie

FULLTEXTOVÉ PROHLEDÁVÁNÍ MULTIMEDIÁLNÍCH DATABÁZÍ

FULLTEXT SEARCH OF MULTIMEDIA DATABASES

DIPLOMOVÁ PRÁCE

Autor: **Bc. Jan Pleva**

Vedoucí DP: **Ing. Jindřich Žďánský Ph.D.**

V Liberci, 16. 5. 2008

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Ústav informačních technologií a elektroniky Akademický rok: 2007/08

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení: **Jan Pleva**

studijní program: N 2612 - Elektrotechnika a informatika

obor: 1802T007 - Informační technologie

Vedoucí ústavu Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto diplomovou práci:

Název tématu: **Fulltextové prohledávání multimediálních databází**

Zásady pro vypracování:

1. Seznamte se s možnostmi automatického zpracování multimédií pro účely indexace jejich obsahu.
2. Seznamte se s principy fulltextového vyhledávání v textových dokumentech a tvorbou odpovídajících databází.
3. Navrhněte databázovou strukturu vhodnou pro indexaci multimediálního obsahu.
4. Vytvořte uživatelské rozhraní k navržené databázi.

Prohlášení:

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 16. 5. 2008

Podpis Jan Pleva

Poděkování:

Děkuji za cenné rady, podměty a připomínky vedoucímu diplomové práce Ing. Jindřichovi Žďánskému. Bylo mi ctí podílet se na realizaci dalšího kousku puzzle propojujícího velké úsilí ústavu rozpoznávání řeči a lidí, kteří mohou mít užitek z činnosti tohoto ústavu.

Neméně díků patří i mým rodičům za neklesající důvěru a podporu. Děkuji kamarádům, kteří za mnou stáli, Helče M, Honzovi B, Jakubovi Z, a celé naší mládeži. Nemohu opomenout ani věrné spolubydlící Mariána a Standu, kteří mě povzbuzují již nějaký ten rok. Poslední, ale nejdůležitější dík patří Ježíši, bez kterého bych tu nebyl.

Abstrakt

Cílem diplomové práce je vytvořit funkční multimediální vyhledávač. Práce pojednává o základních principech a pravidlech tvorby webového vyhledávače, zejména oddělenosti jednotlivých vrstev. V práci je rozebrána problematika a zpracování multimediálních dat. Podrobně je popsán jazykový model češtiny, který zajišťuje skloňování slov a tak výrazně zlepšuje výsledky hledání. Jsou zde rozebrány a vysvětleny základní druhy vyhledávacích algoritmů a jejich fungování. Práce detailně popisuje strukturu a tvorbu uživatelského prostředí.

Vyhledávač má implementovány dva vyhledávací algoritmy. Syntaxe dotazovacího jazyka je tvořena jednoduchými prvky, na které si uživatel snadno a rychle zvykne. Též je možno využít přehledného a intuitivního pokročilého vyhledávání. Vyhledávač disponuje nadstandardními možnostmi omezení výsledků, což přispívá k zpřesnění hledaných dokumentů. Moderní prostředí vyhledávače je navrženo přátelsky pro uživatele. Použity jsou moderní prvky webové technologie.

Výsledkem práce je funkční multimediální vyhledávač.

Klíčová slova: *Vyhledávač, Multimediální data*

Abstract

The goal of this thesis is to create a functional multimedia search engine. The work deals with the basic principles and rules to design a Web search engine, and particularly, it discusses the single layers separateness. This thesis examines the issue of multimedia data processing. It describes in detail the Czech language model which allows word declination and by this means it improves significantly the search results. The basic types of search algorithms and their functions are analyzed and explained. The structure and implementation of the user interface are described.

The search engine, there are two search algorithms integrated. The query language syntax is composed of simple elements that make user's adaptation easy and fast. It is also possible to use a well organized and transparent advanced search. The search engine offers extra options to restrict the outcome of the results and this helps in specifying of desired documents. The search engine modern look is user-friendly and the latest Web technology is used.

The result of this work is a functional multimedia search engine.

Key words: *Search engine, Multimedia data*

Obsah

Seznam obrázků

Seznam tabulek

Seznam rovnic

Slovníček

Score – jistota rozpoznání. Určuje, do jaké míry si můžeme být jisti, že daná věc (slovo, mluvčí, téma) byla správně rozpoznána.

Query – obecně dotaz, může obsahovat slova, nebo značky či speciální syntaxi. Používá se v souvislosti s databázemi nebo jednoduchým dotazovacím jazykem.

Term Weighting – váha termu, číselné vyjádření důležitosti termu

Search engine – vyhledávací stroj, neboli vyhledávač (př.: Gogole).

Wildcard – divoký znak, zastupuje libovolný znak nebo více znaků v řetězci

Úvod

Historie internetu se začíná psát od roku 1983, kdy se internetová síť rozrůstá v průběhu několika málo let z pouhého tisíce počítačů na víc než milion (okolo roku 1992). Internetová – zatím akademická - síť je čím dál více využívána komercí a je zaplavována reklamou, a to rychlosti růstu a šíření internetu jen napomáhá. V těchto letech přichází na svět revoluční novinka v podobě hypertextového dokumentu od společnosti CERN „*World Wide Web*" (volně přeloženo Celosvětová pavučina), neboli „*WWW*". Tím začíná éra internetu zhruba v takové podobě, jak ho známe dnes.

Spolu s internetem se rozvíjejí i vyhledávací stroje [*search engine*]. Jedním z prvních vyhledávačů byl Archie. Fungoval na principu regulárních výrazů a pomáhal uživatelům nalézt příslušné odkazy na stránky. S rostoucím množstvím www stránek se uživatelé internetu začínají stále více ztrácet v nekonečném světě informací a textu. Obtížné hledání požadovaných informací se stává problémem.

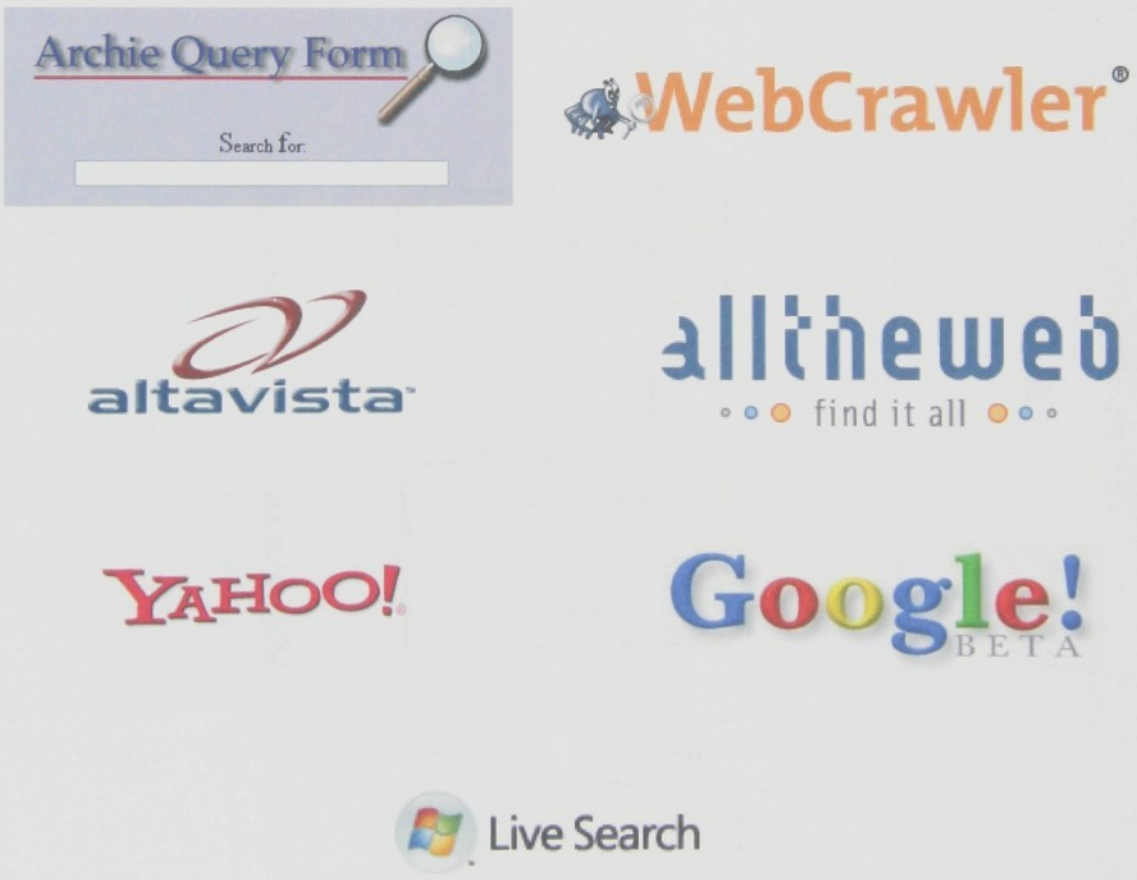
Okolo roku 1994 zavádějí největší portálové společnosti, jako Yahoo! a další, seznamy oblíbených stránek [tzv. *Yahoo! dictionary*]. Ve stejné době neexistovalo mnoho vyhledávačů, které by vyhledávaly v obsahu webových stránek a pakliže ano - jako např. *WebClawler* - tak stejně nedávaly uživateli příliš přesné výsledky, a ještě se na ně muselo vcelku dlouho čekat.

Průkopníkem ve světě chaosu informací byl roku 1995 vyhledávač *AltaVista* od společnosti *Digital Equipment Corporation*. Umožňoval fulltextové vyhledávání s pokročilými funkcemi. Záhy ho koupilo Yahoo!, stejně jako velmi rychlý vyhledávač o pár let později *Alltheweb*. Ve stejné době Larry Page a Sergey Brin začínají společný projekt vyhledávače, založeného na relevanci odkazů s názvem *BackRub*, později *Google beta*. Výsledky takto postaveného vyhledávače byly překvapivě přesné. O tři roky později za pomoci kamarádů a sponzorů zakládají Page s Brinem vlastní společnost s názvem *Google Inc.* Také *Microsoft* se snaží svým *Life Search* z roku 2007 zařadit mezi oblíbené vyhledávače.

Množství informací stále roste. Spolu s novými technologiemi a algoritmy se otevírají dveře k novým zdrojům informací. Jsou jimi právě multimediální data, zvuk, video, obraz, atd. Na internetu jsou uloženy tisíce hodin zvukových a televizních nahrávek. Dosud jediná možnost, jak se v nich orientovat, byly názvy dokumentů a

popisky. Informace ukryté v těchto multimediálních dokumentech však jsou uživatelům skryty. Jedinou možností, jak se k nim dostat, je přehrát tyto záznamy. Při hledání konkrétní věci je to proces velmi zdlouhavý a obtížný, mnohdy se člověk ani nedobere výsledku. Vzniká tak potřeba snadnějšího přístupu k těmto informacím. Je to možnost poskytnout uživatelům informace, které byly dosud nedostupné a obtížně vyhledatelné.

Multimediální vyhledávač je dalším milníkem na cestě k těmto nedostupným informacím. Cílem práce je vytvořit multimediální vyhledávač, který by uživatelům zpřístupnil tisíce hodin audio-video dokumentů z multimediálních archívů, volně přístupných na internetu.

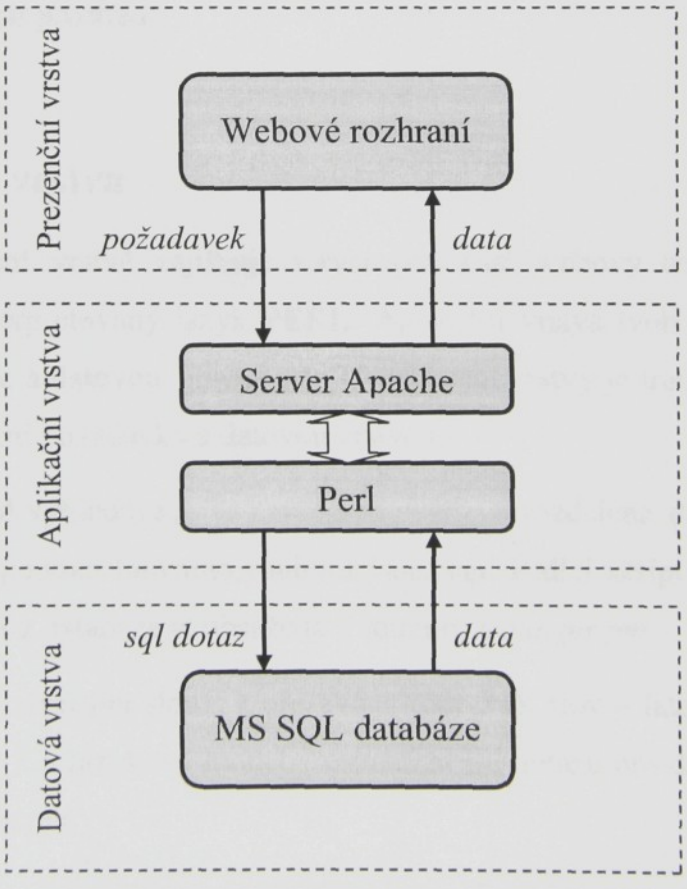


Obrázek 0-1 - loga vyhledávačů

1 Třívrstvý model multimediálního vyhledávače

Multimediální vyhledávač (dále jen MM vyhledávač) je postaven na třívrstvé architektuře, prezenční, aplikační a datové. Výhodou této architektury je možnost změnit či přepracovat určitou vrstvu bez nutnosti zásadně měnit jiné vrstvy. Cílem je, aby byly vrstvy separované, a to jde samozřejmě jen do určité míry. Komunikační rozhraní mezi vrstvami je vždy ovlivněno samotnou technologií té které vrstvy a také vrstev sousedních. I přesto je velmi výhodné vrstvy udržovat oddělené.

Jednotlivé vrstvy mají své úlohy. Prezenční vrstva zajišťuje komunikaci s uživatelem přes webové rozhraní a prezentaci dat. Na aplikační vrstvě je uložena logika vyhledávače a datová vrstva obsahuje uložená data.



Obrázek 1-1 – Schéma třívrstvého modelu MM Vyhledávače

1.1 Prezenční vrstva

Oddělení logiky od vzhledu je velmi důležité. Tento postup totiž umožňuje změnu vzhledu, aniž bychom zároveň poškodili logiku celého systému a naopak, je možná změna logiky systému, aniž bychom se tím dotkli jeho aktuálního vzhledu.

MM vyhledávač je webovou aplikací a proto je grafický vzhled zajištěn pomocí **HTML** a **CSS** stylů.

Pro šablony vzhledu jsem zvolil systém *Tenjin*. Je to velmi výkonný a rychlý templatovací systém dostupný ve více jazycích. Pro naše účely se hodil nelépe v jazyce *Perl* (*plTenjin*).

Efekty zajišťuje na stránkách vyhledávače technologie zvaná **AJAX** (*Asynchronous JavaScript and XML*). Umožňuje měnit obsah stránek bez nutnosti jejich opětovného načítání. Často se využívají již naprogramované ajaxové knihovny, jako v našem případě jQuery. Tento krok je velmi výhodný pro uživatele, neboť pro něj znamená přívětivější prostředí.

1.2 Aplikační vrstva

Na aplikační vrstvě zajišťuje serverovou část webový server **APACHE** a výkonnou část interpretovaný jazyk **PERL**. Aplikační vrstva tvoří prostředníka mezi vrstvou prezentační a datovou. Hlavní náplní aplikační vrstvy je transformace dat mezi vstupně - výstupními požadavky a datovou vrstvou.

Logika MM vyhledávače je naprogramována a rozdělena do několika skriptů, které jsou volány pomocí hlavního souboru *index.cgi*. Řídící skript stránek, parsování dotazu a úprava dat z databáze je obsažena v souboru *manager.pm*.

Soubor *koncovky.pm* slouží k ořezávání koncovek slov – tato záležitost je blíže popsána v kapitole 3.3 Jazykový model. Poskládání sql dotazu pro databázi je zajištěno souborem *sql_dotaz.pm*.

Pro snadnou přenositelnost jsou veškeré specifické informace uloženy v souboru *info.pm*. Jedná se např. o údaje pro připojení k databázi, umístění souborů, atd.

1.3 Datová vrstva

Datová vrstva slouží jako datová základna, úložiště dat. Může jím být souborový systém nebo relační databáze či jiné úložiště. Pro MM vyhledávač jsme zvolili databázi **MS SQL 2005**. Důvodem byla možnost použít tuto databázi pro nekomerční účely a vyzkoušet její vlastnosti.

Tato databáze je velmi propracovaná. Umožňuje sledování dotazů pomocí *SQL Query Profileru* a analýzu pomocí *Database Engine Tuning Advisoru*, a na základě této analýzy jsou pak doporučovány případné změny databáze.

2 Problematika multimediálních dat

Multimediální data jsou automaticky přepisována do textové podoby. Princip je popsán v kapitole 2.1 Rozpoznávač spojité řeči. Spolu s textovými daty jsou v souborech obsaženy i další užitečné informace. Tyto informace pomáhají uživatelům zpřesnit údaje, které hledají.

Každé slovo má své vlastní, jedinečné parametry, které je nutno uchovat. Jedná se o čas začátku, který slouží při přehrávání videa, dále jistotu rozpoznání, která určuje, s jakou jistotou je slovo správně rozpoznáno. Dalším parametrem je příznak bloku mluvčích, který říká, do jakého bloku mluvčích slovo patří (blok obsahuje jednoho či více potenciálních mluvčích). Posledním parametrem je příznak bloku tematického, určující jeho tematickou povahu (každému je přiřazeno jedno či více témat). Tyto unikátní informace ztěžují běžné využívání databázového fulltextového vyhledávání.

V zásadě jsou možné tři způsoby přístupu k ukládání multimediálních dat. První z nich je varianta využití xml databáze, která by sama indexovala celé xml soubory. Odpadla by tak nutnost složitějšího nahrávání dat do databáze. Nevýhodou však je omezená složitost dotazovacího jazyka a nepropracovanost některých částí xml databází.

Další možností by bylo vytvořit z textu fulltext a ke každé pozici slov v dokumentu přiřadit vektor parametrů. Výhodou by mohla být rychlost hledání a úspora místa. Komplikovanější by ovšem bylo zacházení s vektory a hledání omezujících podmínek.

Poslední možností je vytvořit vlastní strukturu v databázi, ve které by byly zakomponovány všechny parametry slova přímo do struktury. Výhodou takovéto struktury je komplexnost a možnost snadno omezit výsledky hledání. Umožňuje též použití jednoho dotazu, a to je pro databázi velmi efektivní.

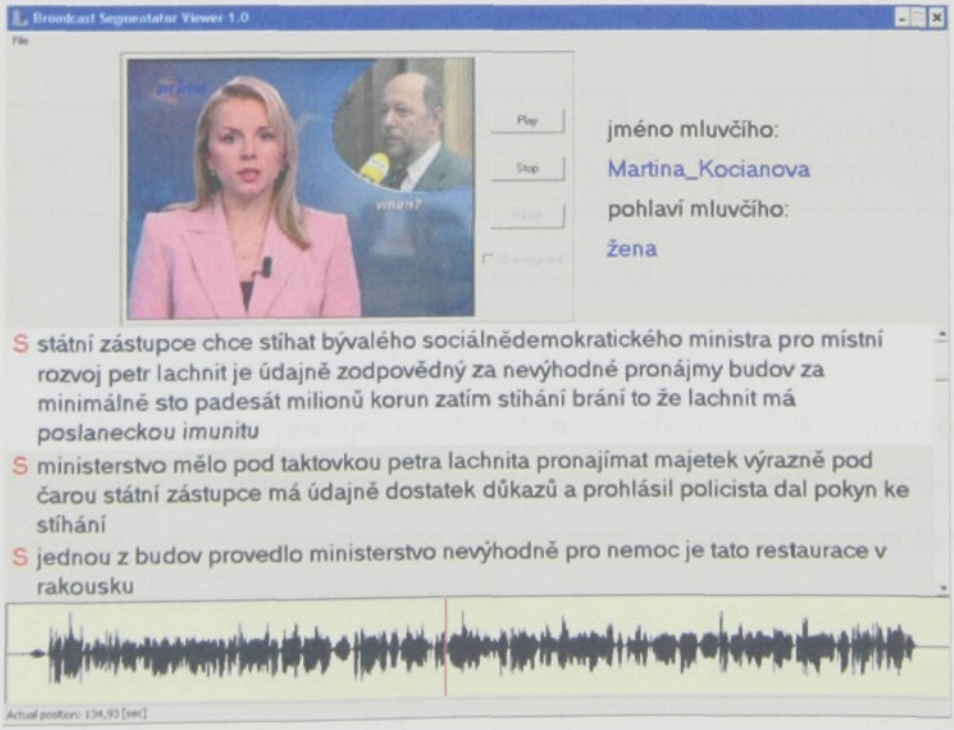
Pro všechny tyto způsoby provedení však platí, že vyhledávací algoritmy jsou jednak velmi jednoduché a dále, nejsou příliš přesné. Zpravidla fungují na principu booleovské metody – viz. kapitola 3.5. Databázové a vyhledávací algoritmy.

Vhodným rozšířením pro všechny varianty jistě bude externí engine s pokročilým vyhledávacím algoritmem, který zajistí lepší výsledky hledání. Problematickou částí bude vždy časová náročnost těchto pokročilých algoritmů.

Cílem této diplomové práce nebylo porovnávat či srovnávat výhody a nevýhody různých druhů řešení, ale vytvořit funkční vyhledávač za použití vlastní databázové struktury, v které bude implementován vyhledávací algoritmus na úrovni databáze, který bude provádět vyhledávání pomocí jednoho dotazu.

2.1 Rozpoznávač spojité řeči

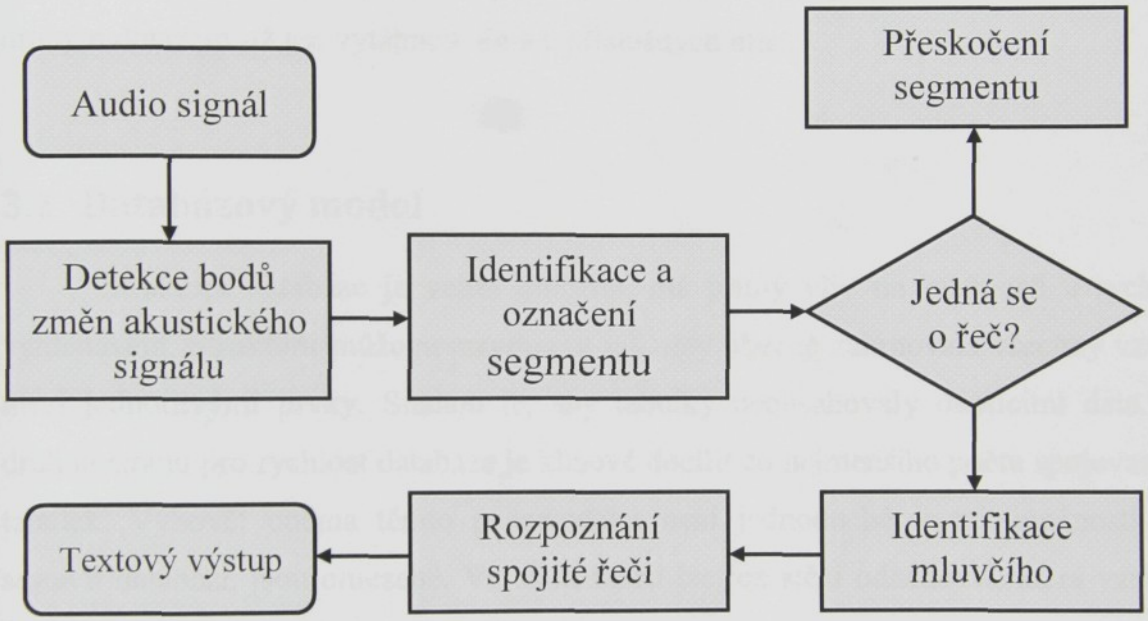
Tým z ITE [Ústavu informačních technologií a elektrotechniky] na Technické Universitě v Liberci, který vede profesor Jan Nouza, vytvořil rozpoznávač řeči pro český jazyk. Výzkum postupně pokročil od jednoslovného rozpoznávání až k složitějšímu rozpoznávání spojité řeči. Tento pokrok otevřel cestu pro vytvoření automatického systému pro přepis televizních a rozhlasových pořadů.



Obrázek 2-1 - Aplikace pro automatický přepis zpráv

Systém provádí následující operace (viz. obrázek 2-2): Nejprve rozčlení záznam celého zpravodajského pořadu na části - tzv. segmenty - obsahující řeč a jiný akustický signál (zejména hudbu, znělky, atd.) Dále rozčlení jednotlivé zpravodajské příspěvky podle charakteru akustického signálu, na části mluvené různými osobami (tzv. mluvčími). U těchto osob lze provést jejich identifikaci, a to přichází v úvahu především u moderátorů a často se vyskytujících reportérů či významných osob.

Jednotlivé příspěvky jsou pak předávány do modulu rozpoznávání řeči, na jehož výstupu se postupně objevuje textový přepis ve formátu XML. Tento přepis je doplněný o seznam pravděpodobných mluvčích u každého segmentu.



Obrázek 2-2 – Průběh rozpoznávání spojité řeči a mluvčího

Současná verze pracuje se slovníkem 320.000 nejfrekventovanějších slov. Úspěšnost takového systému se podle světových standardů měří procentem správně rozpoznaných slov. U automatického přepisu zpráv se toto procento pohybuje kolem 82% v rámci celého zpravodajského bloku.

Výrazně vyšší procento rozpoznání (85 - 95%) je u příspěvků namluvených profesionálními mluvčími ve studiu či v málo rušném prostředí, nízké je naopak u příspěvků, které mají v pozadí hudbu, nebo kde lidé hovoří na rušné ulici, mluví nespisovně atd.

3 Tvorba Multimediálního vyhledávače

Multimediální vyhledávač lze vytvořit mnoha způsoby, protože metod, jak přistupovat ke vznikajícím problémům, je více. Problém ukládání dat je možné realizovat například pomocí XML databáze, kterých existuje celá řada. Po podrobnějším zkoumání jsem však došel k názoru, že pro naše účely stále nejsou dosti propracované. XML databáze mají problémy především, co se týče rychlostí a složitosti dotazů. Od samého začátku však bylo naším cílem vybrat všechna data pomocí jednoho dotazu, proto jsme zvolili databázi klasickou. Je to opět jedno z možných řešení problému nalezení nejpřesnějších hledaných míst. Jiným řešením by mohlo být vybrat z databáze potenciální místa a ty pak lépe matematicky zpracovat mimo databázi a druhým dotazem již jen vytáhnout data z příslušných míst.

3.1 Databázový model

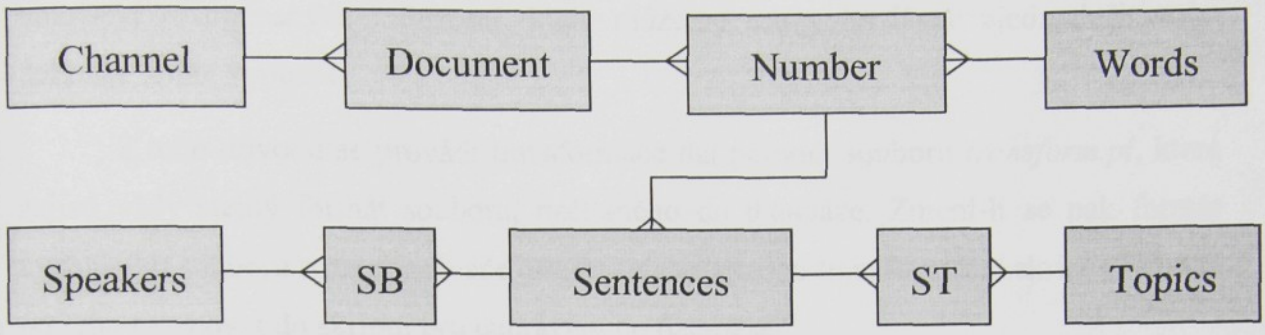
Struktura databáze je velmi důležitá, má přímý vliv na možnosti a rychlost vyhledávání. Strukturu můžeme navrhnout tak, aby obecně zahrnovala všechny vztahy mezi jednotlivými prvky. Snahou je, aby tabulky neobsahovaly duplicitní data. Na druhou stranu pro rychlost databáze je klíčové docílit co nejmenšího počtu spojovaných tabulek. Vyhovět oběma těmto požadavkům není jednoduché, a tak možnosti, jak sestavit databázi, jsou omezené. Ve skutečnosti lze jen stěží odhadovat, která varianta řešení bude pro vyhledávání rychlejší.

Jednou z možností je též propojit strukturu databáze s algoritmem vyhledávání. Spočívá v tom, že si spočteme potřebné údaje do algoritmu předem a uložíme je do tabulky. Výhodou tohoto řešení je zvýšení rychlosti, provádíme-li veškeré algoritmy na úrovni databáze, jako v našem případě. Pro řešení s algoritmem mimo databázovou úroveň nebude toto řešení příliš výhodné.

Jako základní algoritmus pro vyhledávání jsme zvolili relevanci n-tic. Tento algoritmus je podrobněji popsán v kapitole 3.5 Databázové a vyhledávací algoritmy.

Základem tohoto algoritmu je počet slov v dokumentu. Pro tento účel byla vytvořena tabulka Number, ve které jsou zapsány počty výskytů slov v dokumentech. Napojena je na slova (Words), dokumenty (Document) a text dokumentů (Senteces).

Pro návrh databázového modelu jsem použil program CASE Studio 2. Na následujícím obrázku 3-1 je vyobrazena zjednodušená struktura databáze pomocí E-R diagramu. (kompletní struktura viz. Příloha A).



Obrázek 3-1 – E-R diagram struktury databáze

V tabulce *Channel* jsou uloženy názvy všech kanálů, na kterých byly dokumenty vysílány. Do tabulky *Document* jsem uložil všechny klíčové informace o dokumentu, jeho název, umístění na internetu, délku dokumentu, datum vysílání a další pomocné informace. V tabulce *Words* jsou uloženy slova, která jsou spojena s tabulkou *Number*. Tabulku *Number* obsahuje napočítaný výskyt stejných slov v jednotlivých dokumentech, oproti tabulce *Senteces* je přibližně o polovinu menší.

Tabulka *Sentences* je ze všech tabulek nejobsáhlejší, je v ní uložena posloupnost slov dokumentů s parametry. Obsahuje ukazatele na slova a dokumenty přes identifikátor z tabulky *Number* a všechny další informace týkající se potenciálních možných mluvčí a témat. V tabulkách *Speakers* a *Topics* jsou uloženy jména mluvčích a názvy témat. Přes tabulky *SB* a *ST* jsou pak spojeny s tabulkou *Sentences*. Tyto tabulky umožňují asociovat bloky slov s více mluvčími a tématy.

Možnosti rozšíření

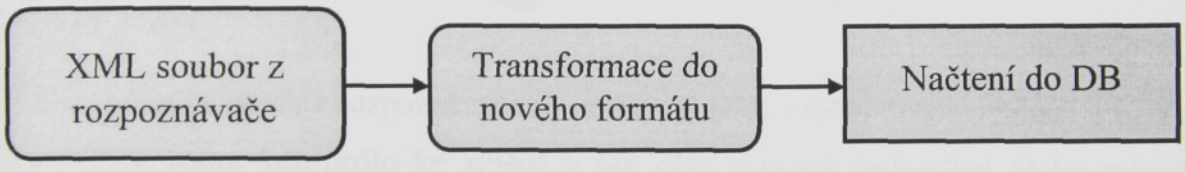
Požadavkem na databázovou strukturu byla možnost rozšíření o další informace, které budeme časem schopni získávat z rozpoznávání nejen audio, ale i video signálu. Jedná se především o obrazová data. Rozšíření spočívá v možnosti navěšování dalších tabulek na tabulku *Sentences*.

3.2 Plnění databáze

3.2.1 Transformace výstupních dat z rozpoznávače

Textový výstup z rozpoznávače je ve formátu **XML**. Tento soubor obsahuje množství rozpoznaných informací, které můžeme pro vyhledávač zjednodušit nebo dokonce úplně vypustit.

Z toho důvodu se provádí transformace dat pomocí souboru *transform.pl*, která zajistí vždy stejný formát souboru, načítaného do databáze. Změní-li se pak formát výstupního souboru z rozpoznávače, přepracuje se snadno transformační skript a nebude potřeba zasahovat do skriptu pro nahrávání do databáze.



Obrázek 3-2 – Zpracování výstupních XML souborů z rozpoznávače

XML soubor z rozpoznávače se skládá z hlavičky (*head*), informační hlavičky (*userinfo*), bodů změn (*changepoint*) a bloků. Tyto bloky jsou v zásadě dva druhy, *speach* – řeč nebo *noise* – hluk. Bloky s hlukem můžeme při zpracování vypustit, neobsahují pro vyhledávač užitečné informace.

Výše zmíněná hlavička (*head*) obsahuje informace o použitém způsobu rozpoznávání, verzi rozpoznávače, verzi a velikosti použitého slovního korpusu a další informace, nepotřebné pro vyhledávač.

V informační hlavičce (*userinfo*) jsou údaje o rozpoznaném dokumentu, které jsou již užitečnější. Obsahuje informace o umístění dokumentu (*source*), jeho názvu (*name*), vysílacím kanálu (*channel*), datu vysílání (*date_trans*) a délky dokumentu (*duration*).

Po transformaci vypadá hlavička následovně.

```
<adocinfo>
	<name>168 hodin</name>
	<channel>ČT24</channel>
	<date_create>2008-04-20 14:28:33</date_create>
	<date_trans>2008-02-03 22:32:00</date_trans>
	<duration>00:26:48</duration>
	<n_of_block>134</n_of_block>
	<source>http://ct1streaming.visual.cz/new/asx/high/168Hodin-
030208.asx</source>
	<start_time>00:00:00</start_time>
</adocinfo>
```

Oproti původním informacím je hlavička obohacena o údaje o počtu bloků (*n_of_block*), datu přetransformování (*date_create*) a času začátku dokumentu (*start_time*).

V souborech z rozpoznávače se nacházejí body změny (*changepoint*), které nám označují místa, kde došlo ke změně a tak nám označují jednotlivé bloky mluvčích v dokumentu. Tyto bloky jsou definovány časově.

Témata pro tematický blok se získají projitím textu bloku a určením klíčových slov. Podle těchto slov se následně vyberou a přiřadí bloku příslušná témata. Zpravidla jeden tematický blok obsahuje více bloků mluvčích. Příkladem může být třeba Zpravodajství, kde více lidí promlouvá na určité téma.

```
<topic score="-32.7571">volby</topic>
```

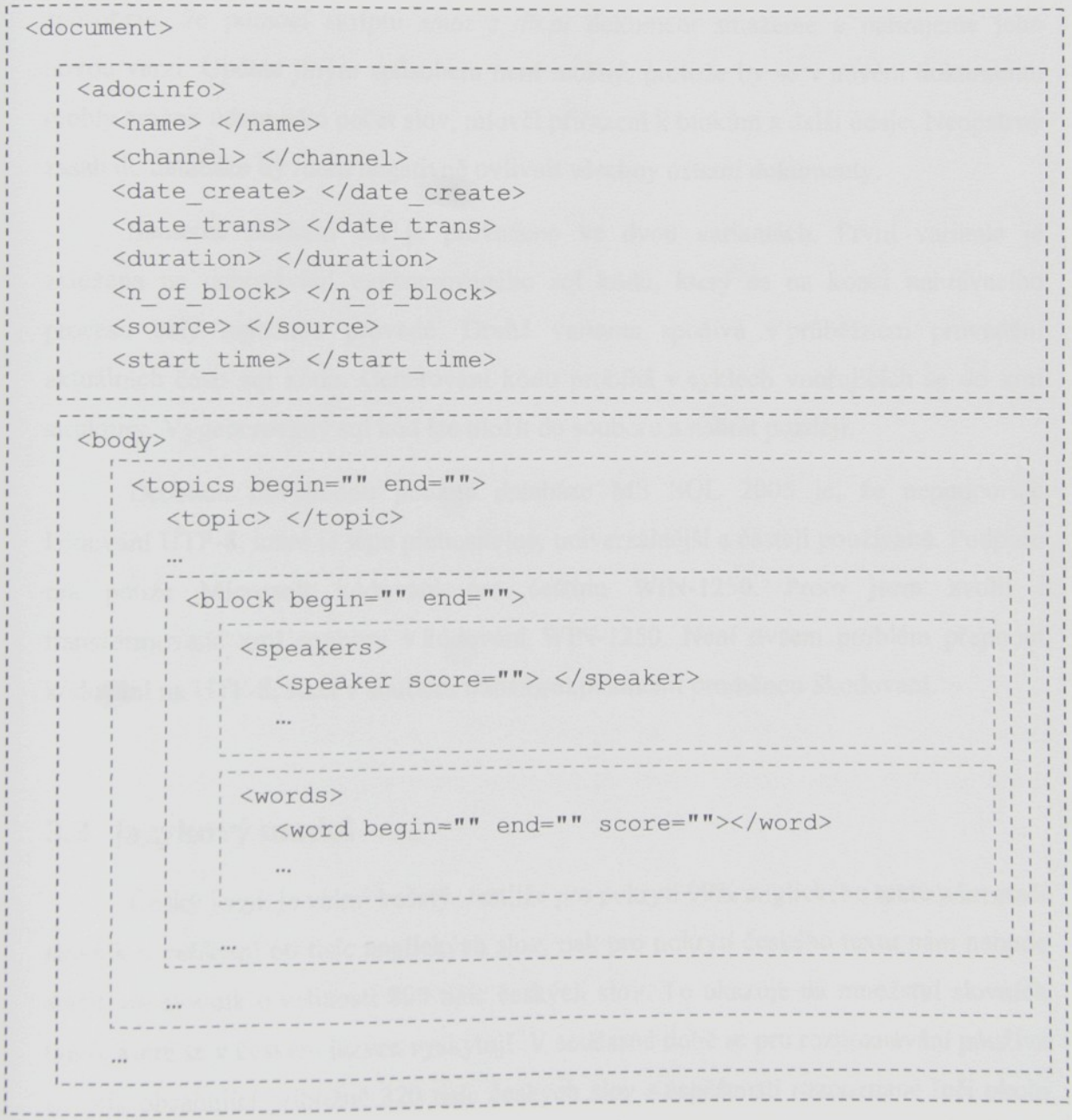
Blok mluvčích obsahuje seznam potencionálních mluvčích, kteří pravděpodobně řekli příslušná slova. Každý mluvčí nebo téma má určité číslo - *score*, vyjadřuje míru jistoty, že právě tento mluvčí řekl příslušná slova nebo že právě toto téma patří k příslušnému textu. V modelu rozpoznávání je zahrnut model tzv. obecného mluvčího, ženy a muže. Tito obecní mluvčí slouží jako měřítko pro ostatní mluvčí. Podle jejich *score* se ořezávají mluvčí jen na ty, kteří svým *score* překonali obecného mluvčího.

```
<speaker score="-35.7839">vaclav_klaus</speaker>
<speaker score="-38.9412">muž</speaker>
```

Bloky mluvčích také obsahuje, kromě seznamu potenciálních mluvčích, posloupnost slov. V posloupnosti slov nejsou obsažena žádná interpunkční znaménka (čárky ve větách, tečky, otazníky,...). Každé slovo má tři parametry. Čas začátku a konce, a score určující jistotu rozpoznání. Časy začátků a konců jsou v setinách sekund.

```
<word begin="1598" end="1630" score="79.0675">dobrý</word>
```

Struktura nového přetransformovaného xml dokumentu je znázorněna na obrázku. Tři tečky symbolizují opakující se tagy a bloky.



Obrázek 3-3 – Struktura XML souboru

3.2.2 Načítání dat do SQL databáze

Nově přetransformovaný xml soubor se nahrává do databáze pomocí nahrávacího skriptu *nacti_do_db.pl*. V tomto souboru se nejdříve kontroluje, zda daný dokument již není obsažen v databázi. Pro každý dokument je unikátní jeho uložení (*source*), které obsahuje v názvu cestu a identifikační prvky jako datum a název dokumentu.

Př: **http://ct1streaming.visual.cz/new/asx/high/168Hodin-030208.asx**

Je-li dokument již obsažen v databázi, můžeme provést jeho update a to tím způsobem, že pomocí skriptu *smaz_z_db.pl* dokument smažeme a nahrajeme jeho novou verzi. Update jiným způsobem není možný, protože by se v novém dokumentu mohly změnit údaje jako počet slov, mluvčí přiřazení k blokům a další údaje. Neopatrný zásah do databáze by mohl negativně ovlivnit všechny ostatní dokumenty.

Samotné načítání dat je provedeno ve dvou variantách. První varianta je založena na uchovávání vygenerovaného sql kódu, který se na konci nahrávacího procesu celý najednou provede. Druhá varianta spočívá v průběžném provádění aktuálních částí sql kódu. Generování kódu probíhá v cyklech vnořujících se do xml struktury. Vygenerovaný sql kód lze uložit do souboru a nahrát později.

Drobnou nevýhodou použité databáze MS SQL 2005 je, že nepodporuje kódování UTF-8, které je lépe přenositelné, univerzálnější a častěji používané. Podporu má pouze Microsoft kódování pro češtinu WIN-1250. Proto jsem zvolil i transformované xml soubory v kódování WIN-1250. Není ovšem problém přepnout kódování na UTF-8, stačí v souboru transform.pl změnit proměnou $kodovani.

3.3 Jazykový model

Český jazyk je velmi košatý. Jestliže pro pokrytí 99% anglického textu nám stačí slovník o velikosti 60 tisíc anglických slov, pak pro pokrytí českého textu nám nebude stačit ani slovník o velikosti 800 tisíc českých slov. To ukazuje na množství slovních tvarů, které se v českém jazyce vyskytují. V současné době se pro rozpoznávání používá slovník obsahující přibližně 320 tisíc českých slov s úspěšností rozpoznané řeči okolo 82%.

Množství slovních tvarů je obecně problém vyhledávačů. Většina českých slov se skloňuje tak, že se ke slovu přidávají různé koncovky. Jsou ovšem i slova, kterým se při skloňování mění kmen slova, např. 1.p „den", 3.p „dni".

Tyto problémy se slovními tvary se mohou řešit pomocí jazykového modelu. Pro komplexní řešení se používají dva nástroje. *Lematizátor*, převádí slova na základní slovníkový tvar a *Derivátor*, provádí inverzní úlohu, vypisuje množinu všech možných tvarů, které přicházejí v úvahu.

Řešením by mohla být databáze obsahující základní slovní tvary s vazbou na všechny ostatní tvary slova. Vyhledáním slova by nám pak databáze vrátila všechny jeho tvary. Vytvoření takovéhoto jazykového korpusu by bylo náročné, ovšem fungování stoprocentní. Na stránkách www.pravidla.cz je možno vidět, jak by takový jazykový model fungoval.

Zabývat se problémem komplexního jazykového modelu by však bylo příliš náročné a dalece by to přesáhlo meze této diplomová práce. Cílem této práce bylo vytvořit funkční vyhledávač. Pro splnění tohoto úkolu jsem se rozhodl použít méně náročnou metodu, která by aspoň částečně umožňovala automaticky odstraňovat koncovky slov.

Princip fungování je jednoduchý. Základem je cyklus s omezujícími podmínkami pro jednotlivá slova. Vyhoví-li slovo všem podmínkám a je-li nalezena shoda konce slova s koncovkou, je tato koncovka oddělena a nahrazena divokým znakem[1] (wildcard) pro databázi.

První podmínkou pro vstup do jazykového modelu je *„čisté české slovo"*, tedy nesmí předem obsahovat divoké znaky, číslovky či jiná nečeská písmena. Splnění první podmínky hlídá parser, popsaný v kapitole 3.4 Parsování dotazu, který rozděluje dotaz uživatele do příslušných slovních tříd.

Další podmínku, kterou bylo nutno zavést, je omezení délky slova. Slov kratších než čtyři písmena je velmi mnoho a často nemají pro hledání velký význam, spíše naopak - zahlcují výsledky hledání a přetěžují databázi svým častým výskytem. Proto

[1] Divoké znaky [anglicky wildcard] používají se jako náhrada neznámého libovolného písmene nebo několika písmen. Pravděpodobně pochází z regulárních výrazů. Obecně rozšířené v podvědomí jsou: * = libovolný počet znaků, ? = jeden libovolný znak. V MSSQL databázi jsou ekvivalenty % místo * a _ místo ?.

jsem stanovil, že pro tři a méně písmenná slova se nebude zkoumat, zda mají koncovku, a tato koncovka nebude nahrazována.

Někdo by mohl namítnout, že existuje celá řada „*normálních*" třípísmenných slov, jako třeba „*hra*", „*dům*" a další. To je pravda, ovšem praktické zkušenosti ukazují, že slova s touto délkou se do vyhledávání nedostávají až tak často. Jejich vyřazení nám pomůže ušetřit časové prostředky serveru při zpracovávání dotazu, které by jinak zabralo procházení dlouhého *stop listu*.

Zmíněný Stop list je další podmínkou jazykového modelu. Navzdory výše uvedenému je skutečností, že existují slova, která zdánlivě končí koncovkou, ale při odstranění koncovky by došlo k odstranění příliš velké části slova. Proto je zaveden Stop list, v kterém jsou problémová slova uložena.

Poslední podmínka je zabudovaná přímo ve vnitřním cyklu, který zkouší přiřadit na konec slova správnou koncovku. Pokud se mu podaří najít koncovku na slově, tuto koncovku odstraní, nahradí ji divokým znakem a přejde na ořezávání dalšího slova. Pokud je ovšem slovo po oříznutí kratší než dva znaky, ořezávání se neprovede. Pravděpodobně se totiž jedná o chybné oříznutí. Koncovky totiž mohou být v některých případech až čtyř písmenné.

Všechny důležité koncovky jsem vypsal z *Příruční mluvnice češtiny*. Vynechal jsem jen několik výjimečně se vyskytujících tvarů, které by úspěšné ořezávání koncovek jen zhoršilo[2]. Koncovky jsem seřadil do pole podle jejich délky. Při cyklu se snažím pomocí regulárních výrazů najít na konci slova koncovku, kterou mám v poli. Zkoušení se provádí od nejdelších koncovek, proto je důležité seřazené pole. Delší koncovky totiž obsahují části, které následně mohou být samostatnými koncovkami. Pokud by se provedlo odříznutí kratší koncovky dříve, než koncovky delší, ona delší by již nebyla nalezena. Po odříznutí koncovky se přeruší cyklus a pokračuje se opět dalším slovem.

Slovům, která vyhověla všem podmínkám, ale nebyla u nich nalezena koncovka, se na konci divoký znak přidá také.

Do jazykového modelu, implementovaného jako funkce, je posíláno pole slov. Výstupem je opět pole slov s odstraněnými koncovkami, nahrazenými divokým znakem.

---

[2] Konkrétně se jedná o koncovku „c", která je velmi ojedinělá. Její výskyt je při skloňování přijatých slov ze slovanských jazyků, převážně jmen měst. Druhou výjimku jsem udělal u skloňování podle typu„kuře". Nezohledňuji kmenotvornou příponu –et– a v množném čísle – at– (př. 2.p kuř–et–e).

Důležité je zmínit fakt, že tento jazykový model koncovek je velmi závislý na českých znacích (diakritických značkách). Nikoli v tom smyslu, že by koncovku ořízl špatně, jako spíše to, že koncovku neořízne. Je to způsobeno tím, že koncovky jsou velmi závislé na podobě a délce písmen. Pokud bychom zahrnuli do jazykového modelu i tvary bez diakritiky, ořezávalo by nám to velmi často špatné části. Jako příklad může posloužit slovo „přepych". Koncovka je „-ých", ale pokud odstraníme diakritiku, model ořízne slovo „přepych" na pouhé nesmyslné „přep".

Účinnost tohoto modelu je odhadnuta pro běžné dotazy až na 75% úspěšných ořiznutí. Bude-li uživatel psát dotazy bez diakritiky, účinnost klesne odhadem na 40%. Poměr jednoduchosti a účinnosti je velmi slibný. Narazí-li uživatel přece jen na problém, lze jazykový model vypnout a použít divoké znaky.

Jazykový model byl odzkoušen na prvních 600 nejfrekventovanějších českých slovech a na tvarosloví z Příruční mluvnice češtiny.

Příklad výstupu při testování ořezávání koncovek:

růže - růž%
růží - růž%
růžím - růž%
růžemi - růž%

3.4 Parsování dotazu

Pojem parsování je odvozený od nástroje, kterému se říká parser. Parser se používá v souvislosti s XML nebo regulárními výrazy. Parse v překladu z angličtiny znamená rozebrat, analyzovat, oddělit, a to také parser dělá.

Dotaz uživatele vzniká na úrovni uživatelského rozhraní, kde ho uživatel sestaví a odešle. Sestavení dotazu může být prostým napsáním slov, která chce uživatel najít nebo pro složitější dotaz použije omezující podmínky, či upřesňující formulace (*uvozovky – přesná fráze, negace, použití OR*). Pro poskládání složitějšího dotazu je ovšem již zapotřebí znát syntaxi, a to by pro běžného uživatele mohlo být překážkou. Uživatelé tedy mohou využít možnosti *pokročilého hledání*, které poskládá složitější dotaz za ně.

Dotaz se obecně může skládat ze dvou základních entit[3], kterými jsou termy[4] a omezující podmínky. V uvedeném příkladě se vyskytují dva termy „*prezident*" a „*volby*" a dvě omezující podmínky data - „*datum:13/11/07*" a vysílacího kanálu - „*kanal:ČT24*".

Př: **prezident volby datum:13/11/07 kanal:ČT24**

3.4.1 Syntaxe

Dotazovací syntaxi jsem se snažil navrhnout tak, aby pokud možno co nejméně mátla uživatele a pokud možno byla co nejvíce podobná již užívané syntaxi jiných celosvětových vyhledávačů (viz. Google). Základní prvky, jako je přesná fráze pomocí uvozovek, negace pomocí slova NOT nebo mínusu, OR čili nebo, jsem zachoval naprosto stejné. Tak jako v ostatních vyhledávačích i tady platí, že jsou všechny termy mezi sebou ve vztahu AND.

Př: **"prezidentské volby" Švejnar OR Klaus NOT Havel**

Syntaxe pro omezující podmínky mi již dávala větší prostor pro zvolení formátu. Rozhodl jsem se použít snadno pamatovatelný formát, který bude možné aplikovat na všechny omezující podmínku. Základem je jméno omezení a za dvojtečkou následuje ono omezení. Omezení se mohou řetězit pomocí pomlček.

Př: **název:omezení-další_omezení-a_další**

Př: **mluvčí:Klaus-Jan_Švejnar**

3.4.2 Třídy termů

Výše popsané možnosti upřesňování dotazu spolu s podmínkami nám dávají dost možností, jak kombinovat a upřesňovat hledané informace.

Pro generování sql dotazu již není příliš výhodné mít tolik možností. Pro každou změnu - namísto nejjednoduššího vypsání slov za sebou - se musí speciálně vytvářet či přetvářet generovaný sql kód. Proto jsem vytvořil pět tříd pro termy, které zpracovávají vždy stejný typ termů.

---

[3] Entitou je chápán objekt, množina dat.
[4] Pojmem term je chápána jednotka entity.

První třídou, která se parsuje z dotazu, jsou termy „přesné“[5]. Jako jeden term je chápán celý výraz ohraničený uvozovkami. Těchto termů může dotaz obsahovat i více.

Př: **“prezidentské volby“**

V druhé třídě termů jsou všechny druhy omezujících podmínek. V následující tabulce jsou uvedeny možné tvary, druhy, jejich zápis a vysvětlení.

Název	Zápis	Vysvětlivka
dokument	dokument:Zprávy	*Omezuje hledání v dokumentech*
kanál	kanal:ČT24	*Omezuje hledání pro vysílací kanál*
datum	datum:12/01/07-02/12/07 datum: -02/12/07 datum:12/01/07 datum:12:00/12/01/07-20:00/02/12/07 datum: -20:00/02/12/07 datum: 20:00/02/12/07	*Omezení data vysílaných dokumentů. Pomlčka a datum znamená do data, bez pomlčky od data.*
délka	delka: 00:30-01:00 delka: -01:00 delka: 00:30	*Omezení délky dokumentu. Pomlčka má stejný význam jako u předchozího data.*
mluvčí	mluvci:Václav_Klaus	*Omezení na mluvčí.*
téma	tema:politika	*Omezení na témata.*
skóre slova	ss:60	*Omezení s jakou jistotou bylo slovo rozpoznáno.[v %]*
skóre mluvčího	sm:60	*Jistota rozpoznání. Stejné jako u skóre slova.[v %]*
skóre tématu	st:60	*Jistota rozpoznání. Stejné jako u skóre slova.[v %]*

Tabulka 1 - druhy omezujících podmínek

[5] Slovem „přesné“ je vyjádřeno, že se hledané slovo či fráze bude hledat přesně tak, jak bylo zadáno.

Třetí třídou termů jsou všechna slova, která obsahují divoké znaky (* zastupuje libovolný počet znaků, ? zastupuje jeden znak). Divoké znaky mohou být umístěny libovolně před slovem, ve slově nebo za slovem.

Př: **letadlo *letělo** *(vzletělo, přeletělo, doletělo, letělo...)*

Čtvrtou třídou termů jsou obyčejná slova. Tyto slova jsou při zapnutém ořezávání koncovek slov vpuštěny do jazykového modelu. Spolu s nimi je vpuštěn i upřesňující prvek OR. Tento prvek však neprojde dalšími podmínkami jazykového modelu, a tak vyjde nezměněn.

Poslední pátou třídou termů jsou slova, která nevyhověla ani jedné výše zmíněné třídě. Mohou to být číslovky nebo jiné značky a znaky. Stav nynější databáze obsahuje slovní přepis všech číslovek. Již se chystá transformace těchto slovních čísel na řecké číslice.

Zmínit se ještě musím o upřesňujícím prvku NOT (neboli mínus „–"). Tento prvek, lze použít u každé třídy. Výjimky jsou pouze upřednostňující podmínky pro skóre slova, tématu a mluvčího.

3.4.3 Parser

Srdcem parseru jsou regulární výrazy, pomocí kterých se z dotazu vysekávají příslušné třídy. Parser pracuje v cyklech jednotlivých tříd. Před vstupem dotazu do parseru jsou provedeny dvě důležité akce.

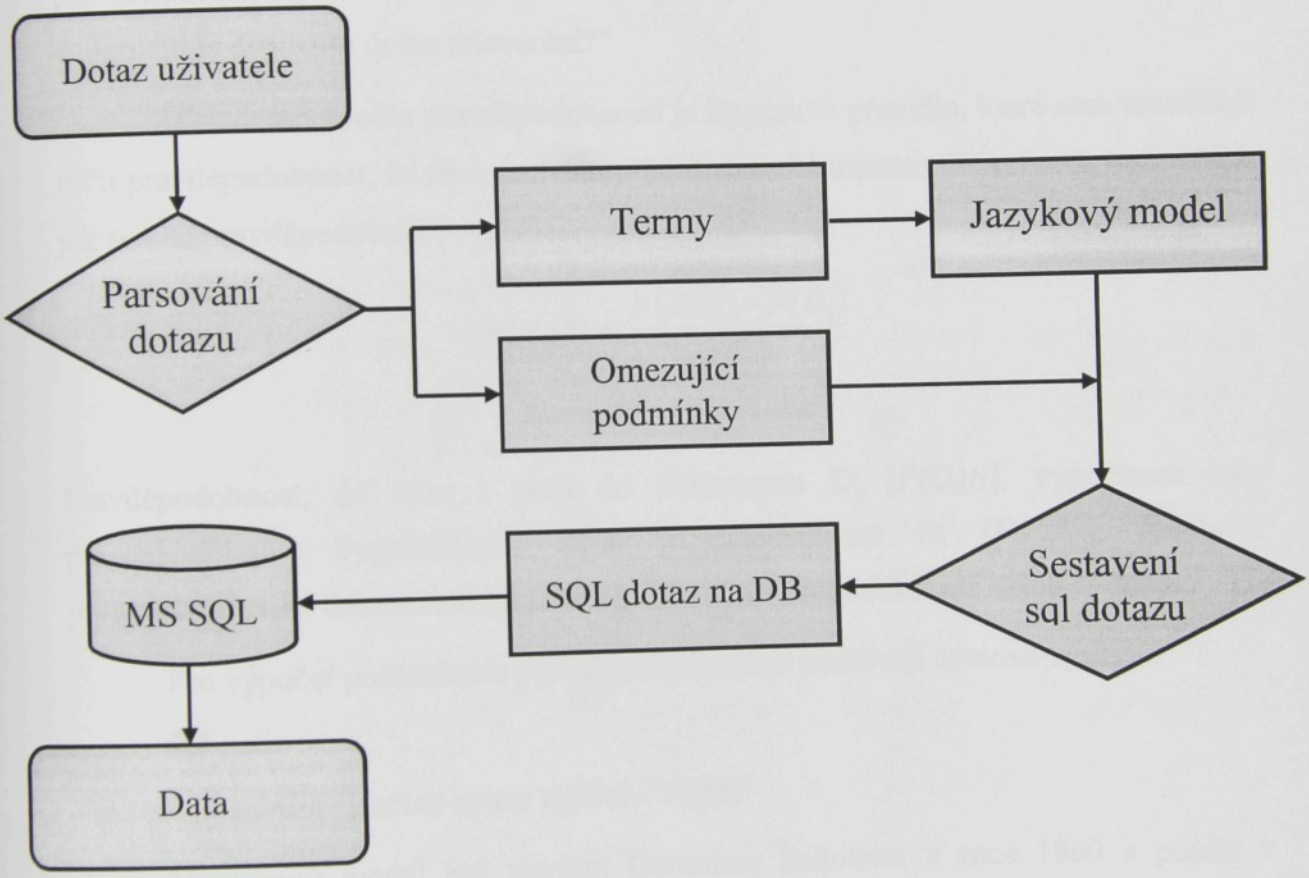
Jsou odstraněny všechny nebezpečné znaky, které by mohly narušit regulární výraz a podobné značky se nahrazují jednotnou symbolikou. Příkladem může být, pokud uživatel použije jednoduché uvozovky namísto dvojitých.

Druhou akcí je zmenšení všech písmen v dotazu. Vyhledávač je **Case Insensitive**, což znamená, že nedělá rozdíly mezi velkými a malými písmeny. Je to umožněno především díky nastavení databáze, která je také case insensitive.

Průběh parsování tříd je následující. Pomocí regulárního výrazu parser zjistí, zda dotaz obsahuje prvek z příslušné třídy, pokud obsahuje, vyjme ho a načte do pole. Do druhého pole uloží příznak, byl-li před termem zápor a pokračuje ve snaze najít další výskyty. Pokud neuspěje, přejde na další třídu.

Po dokončení parsování odesílá jednotlivá pole tříd na zpracování souboru sql_dotaz.pl, který posílá zpět příslušný sql kód. Parser je pro všechny druhy hledání stejný. Podle typu hledání se liší sql kód vráceny ze souboru sql_dotaz.pl. Vrácený kód se poskládá do komplexního dotazu a odešle se na databázi.

Vrácená data z databáze prochází korekturou, zvýrazněním hledaných míst. Data se ukládají do hashe[6], která je posílána na prezenční vrstvu. Prezenční vrstva se stará o prezentaci dat.



Obrázek 3-4 – Průběh zpracování dotazu uživatele

[6] Hash je v Perlu datová struktura tvořená klíčem a hodnotou („key"=>value)

3.5 Databázové a vyhledávací algoritmy

Algoritmů pro nalezení nejrelevantnějších dokumentů je celá řada. Liší se svoji složitostí a výpočetní náročností. V principu fungování algoritmů se dají vysledovat tři základní metody přístupu, booleovský, vektorový nebo pravděpodobnostní. Tyto základní metody přístupu se většinou dále kombinují s jinými algoritmy, které mají za úkol zpřesnit základní metodu.

Pravděpodobnostní model

Pravděpodobnostní model, jak již sám název napovídá, je postaven na pravděpodobnosti. Hlavní otázkou této metody je: „Jaká je pravděpodobnost, že dokument je pro tento dotaz relevantní?"

Základem výpočtu pravděpodobností je Bayesovo pravidlo, které nám umožňuje určit pravděpodobnost, že term patří do příslušného dokumentu. Rovnice 3-1 nám říká, jak se tato pravděpodobnost vypočte.

$$P(D_i|t) = \frac{P(t|D_i) * P(D_i)}{P(t)}$$

Rovnice 3-1 - Bayesova věta

Pravděpodobnost, že term t patří do dokumentu D_i $[P(D_i|t)]$, vypočteme jako pravděpodobnost vygenerování termu t dokumentem D_i $[P(t|D_i)]$, násobeno pravděpodobností dokumentu Di $[P(D_i)]$, děleno pravděpodobností termu t $[P(t)]$.

Pro výpočet počátečních pravděpodobností se používají četnosti výskytů.

Vektorový model (Vector space model - VSM)

Vektorový model byl vyvinut Gerardem Saltonem v roce 1960 a použit v takzvaném SMART-systému.

V modelu vektorového prostoru je každý dokument či dotaz reprezentován vektorem vah. Výpočet váhy objektu nejčastěji vychází ze vzorce Saltonovy váhy, který je dán vztahem Rovnice 3-2, kde tf_i je počet termů i v dokumentu, df_i je počet dokumentů obsahujících term i a D je počet všech dokumentů.

$$w_i = tf_i * \log\left(\frac{D}{df_i}\right)$$

Rovnice 3-2 - Salmonova váha

Při zadání dotazu se měří podobnost vektoru dokumentu s vektorem dotazu. Podobnost Sd,q mezi dokumentem d a dotazem q může být spočítána pomocí Rovnice 3-3, kde $w_{d,t}$ a $w_{q,t}$ představují číselnou váhu termů t v dokumentu d a dotazu q.

$$S_{d,q} = \sum_{t=0}^{d+q} w_{d,t} * w_{q,t}$$

Rovnice 3-3 - Rovnice podobnosti vektorů

Dokumenty se seřadí podle vypočítané podobnosti pro každý dokument.

Booleovská metoda

Booleovská metoda vyhledávání je založena na booleovské algebře. Při booleovském vyhledávání jsou termy proloženy logickými spojkami AND, OR, NOT. Dotazy je možné znázornit pomocí Vennových diagramů[7]. Booleovská metoda vyhledávání je metoda „přesného hledání" (*Exact Match*), daný dokument jasně splňuje nebo nesplňuje požadavky na výskyt hledaných termů v dokumentu. Nevýhodou booleovského přístupu je fakt, že existují pouze dva stavy ([nalezeno, nenalezeno], [1,0]).

Metoda zvaná „*Best Match*" zlepšuje tuto nevýhodu pomocí rozšířené logické hodnoty, a kombinuje vážený systém *Weighting Methods* s logickou metodou, tak aby dokumenty s lepším hodnocením byly zařazeny nejvýše. Dotazem se vyberou dobře a nejlépe odpovídající dokumenty, výsledkem je seřazený list ohodnocených dokumentů.

Výsledný seznam může zahrnovat i odhad kvality. Těžko se rozlišuje mezi přesným a nejlepším výsledkem. Dokument přesně odpovídající hledanému dotazu nemusí být nejlepší, hledané části mohou být od sebe výrazně vzdáleny. Metoda *Best Match* je jednou z nejčastěji používaných metod. Obecně se dá říct, že tato metoda má nejlepší výsledky v poměru výpočetní náročnost : přesnost.

[7] Vennův diagram je množina charakterizovaná uzavřenou křivkou. Pomocí těchto množin lze vyjádřit matematický nebo logický vztah mezi věcmi.

3.5.1 Hledání v dokumentech

Jako základní algoritmus pro hledání v databázi jsem zvolil metodu zvanou *Best Match* popsanou výše. Jedná se o metodu, která umožňuje, díky své výpočetní nenáročnosti, implementaci již na úrovni databáze. Zaručuje se tak nejkratší doba pro zpracovávání dotazů.

Nevýhodou je, jak již bylo zmíněno, problematické určení nejlepšího výskytu. Tento problém částečně řeší tzv. relevance dat. Hledáme takový výskyt, kde bude největší množství všech hledaných termů. Term s nejmenším počtem výskytů pro nás bude hranice. Uvažujeme teoreticky, že termy spolu tvoří n-tice a my hledáme největší výskyt n-tic v dokumentech. Bude-li nějaký z termů převyšovat četnost výskytů nejméně se vyskytujícího termu, nebude se na to brát ohled.

Nevýhodou však zůstává fakt, že budou-li se v dokumentu nacházet všechny termy, ale některé jen na začátku a některé jen na konci dokumentu, dokument bude i tak zařazen jako vyhovující, ale přesto nebude obsahovat informace, požadované uživatelem.

Na obrázku je znázorněn příklad vyhledávání tří termů. V bublině jedna je vidět počet nalezených n-tic a v bublině dvě je počet n-tic.

1 2

Results | Messages

	rn	cnt	id	id_num	match	nice	pocet	text	date
1	1	65	27	32339	46	3	506	se další zajímavostí pět ekonomické naviděnou a...	03.11.07 - 15:(
2	2	65	25	28667	43	3	606	průzkum hospodářské komory dvě třetiny pozdrav...	03.11.07 - 09:(
3	3	65	26	30490	43	3	714	přehled zpráv sledujete ČT dvacet čtyři o v snad ...	05.11.07 - 21:(
4	4	65	44	50723	36	3	539	ty okamžiky dobrý den vítejte u dalších vydání m...	03.11.07 - 18:(
5	5	65	40	46162	31	3	524	i když oba Čína včera dobrý večer začíná další p...	03.11.07 - 20:(
6	6	65	33	41803	29	3	1475	pokračujeme hned po stručných zprávách v touto...	04.11.07 - 13:(
7	7	65	18	19324	26	3	592	díky za pozornost a dobro nad opoziční sociální d...	03.11.07 - 19:(
8	8	65	55	68216	26	3	1233	počasí a sportem naviděnou zítra ráno oslavovaly...	04.11.07 - 20:(
9	9	65	32	38841	25	3	1536	za o vítejte na hezkou neděli vám všem ještě pře...	04.11.07 - 12:(
10	10	65	50	60832	18	3	672	devět sedm čtyři sedm tři sedm šest pět jedna ale ...	04.11.07 - 19:(

Obrázek 3-5 – Výstup informací z databáze metodou hledání v dokumentech

3.5.2 Hledání oblasti v dokumentech

Druhý algoritmus implementovaný do vyhledávače slouží především pro podrobnější zobrazení již nalezeného dokumentu. Ke vzniku druhého algoritmu, který hledá oblasti v dokumentu, vedl nápad že, uživatelé vlastně nejčastěji hledají určitou oblast v dokumentu, ve které je nedlouze zmíněno hledané téma. Velikost oblasti se dá libovolně měnit. Implementace tohoto algoritmu do databázové úrovně bylo již obtížnější.

Nevýhodou této metody je podstatně delší doba hledání, výhoda je v tom, že je o něco přesnější. Princip spočívá v tom, že se prochází tabulka nalezených výskytu a hledají se nejbližší sousedé, kteří se označí číslem jako skupina. Počet termů ve skupině je určující pro seřazení.

Ani tato metoda však není bezchybná. Je-li v oblasti větší výskyt jednoho z hledaných termů, je oblast posunuta výše, aniž by nutně obsahovala další termy.

Nevýhoda delšího zpracování a častého výskytu jednoho termu se kompenzuje kombinací této metody s předešlou (*Best Match*). Dokumenty jsou omezeny pouze na ty dokumenty, ve kterých se aspoň jednou vyskytují všechny termy.

Pro drobné zpřesnění se ještě provádí rozptyl v oblastech.

1 2

Results | Messages

	begin	cislo	id_document	pocet	rozptyl	text
1	450	128670	50	4	1110,58734460645	právníci soudci a celá občanská společnost nebude.
2	1397	131063	50	4	1135,2946754037	zakázané město zná celý svět a Číňané jsou na ní p.
3	1276	130760	50	3	904,906809197534	britského archeologa Howarda Cartera který pro nov.
4	958	129960	50	3	1170,59400875537	zvolte proto opačnou cestu a vyzývá prostřednictví...
5	49	127692	50	3	2147,02841154932	Kennedy odhalena tvář faraona Tutanchamona pro ..
6	394	128533	50	2	329,511760032931	Tereza Krásenská nyní cítí česká televize nové Ha..
7	279	128236	50	2	464,569155239562	a židle a cestou vytvořili barikádu policisté na to reag.
8	1234	130656	50	2	519,016377390926	asi padesát vyvolených faraonova mumie stará přes .
9	1302	130789	50	2	554,371716450253	mladý zřejmě poté co si zlomil stehenní kost a do rán.
10	675	129270	50	2	1159,65512114594	v úterý na celostátní konference lidovců a zjistit jestli.
11	317	128328	50	2	1394,41457249987	už nebudu vystupovat co znamená že dojde k podle
12	1068	130250	50	2	2154,55436227541	za volantem řeší celá Evropa například tvrdí projekt.
13	590	129045	50	1	NULL	tom říct pro cizince atraktivní proto protože tady je v.
14	164	127963	50	1	NULL	povinnosti ze zákona cennými papíry a odstupující v

Obrázek 3-6 – Výstup informací z databáze metodou oblasti

Na obrázku Obrázek 3-6 je ukázka dat z databáze. Bublina číslo jedna označuje počet nalezených slov v oblasti a druhá bublina zpřesňuje pomocí rozptylu, která oblast má termy blíže k sobě.

3.5.3 Realizace sql dotazů

Při konstrukci sql dotazů jsem se snažil sestavit tyto dotazy tak, aby byly maximálně optimalizované na rychlost. Při tomto sestavování jsem musel hodně experimentovat a postupně měřit rychlost jednotlivých variant provedení. Pokusím se shrnout zkušenosti, které jsem načerpal.

Dotazy musely být konstruovány jako vnořené, a to je poměrně rychlý způsob.

Př: select * from (select * from word where slovo='slov%') as word

Snažil jsem se vyhnout vnořování v podmínce, které bylo časově velmi pomalé.

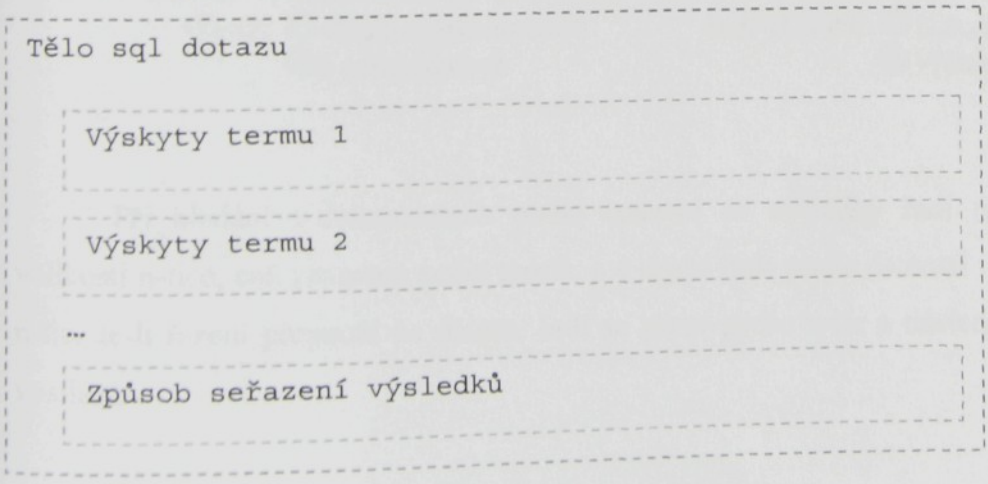
Př: select * from Number where id_word in (select id_word from word where slovo='slov%')

Nedoporučuji ani příliš používat konstrukce s příkazem HAVING a též je velmi brzdící podmínkou konstrukce s NOT IN. Proto jsem řešil zápor hledaných slov pomocí sloupce příznaku záporu, který po sečtení s ostatními hledanými termy signalizuje, zda dokument nebo oblast je bez chtěného slova nebo s chtěným slovem.

Struktura sql dotazů

Obecně se struktura sql dotazů skládá ze tří základních částí. První část tvoří jednotlivé termy hledané zvlášť a pospojované pomocí příkazu UNION. Další částí je tělo, které obaluje první část. V něm se provádí různá spojení a ořezávání promocí GROUP a HAVING za použití SUM, MIN, MAX, COUNT. Poslední částí je způsob řazení záznamů popsané v 3.5.5 Řazení výsledků.

Tělo sql dotazu je pro oba druhy hledání různé.



Obrázek 3-7 – Struktura sql dotazu

3.5.4 Procedury a funkce DB

Databáze MS SQL 2005 je výbornou databází. Ovšem v porovnání s databází MySQL ji chybí některé užitečné příkazy a ty je proto nutné nahrazovat jinými postupy.

Jedním z důležitých příkazů, které MS SQL postrádá, je group_concat(). Tato funkce způsobí v MySQL, že jednotlivá slova ve sloupci jsou poskládána za sebe do řádku do jedné buňky.

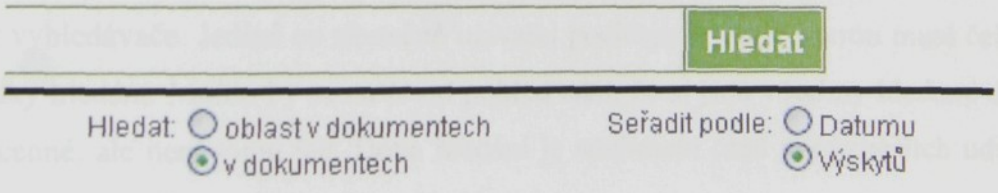
Realizace v MS SQL může být buď pomocí WITH a CASE nebo pomocí funkce s příkazem CURSOR. V našem případě jsem použil funkci.

Do funkce vstupuje číslo začátku požadovaného textu vzaté z tabulky *sentences*. Funkce pak pomocí kurzoru CURSOR a cyklu WHILE projde nalezenou posloupnost textu a pospojuje ho. Výstupem z funkce je pospojovaný text. Tuto funkci jsem nazval „word_list".

Další pomocnou funkcí je „to_char", slouží k převodu data do formátované textové podoby. Do této funkce zašleme datum v základním tvaru, který vystupuje z databáze, a formát ve kterém chceme datum vrátit. Výstupem je formátované datum.

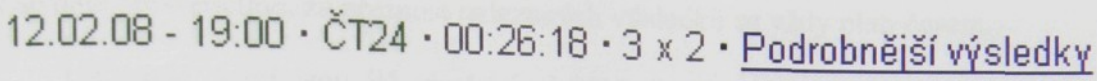
3.5.5 Řazení výsledků a rychlost nalezení

Výsledky hledání je možné řadit buď podle četnosti výskytu, nebo podle data.



Při hledání v dokumentech podle četnosti se výsledky řadí nejdříve podle velikosti n-tice, což znamená počet termů, a v druhé řadě podle četnosti výskytu těchto n-tic. Je-li řazení přepnuté na datum, řadí se první podle n-tic a následně podle data vysílání.

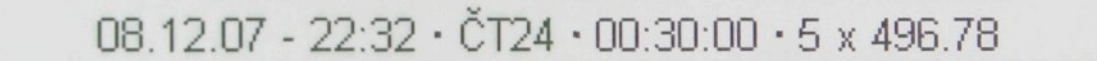
Na obrázku 3-9 můžeme vidět informace o dokumentu. Zleva jsou informace datum vysílání, čas vysílání, název kanálu, na kterém byl dokument vysílán, délku dokumentu, velikost n-tice a počet n-tic nalezených v dokumentu.



Obrázek 3-9 – Informace o nalezených výskytech v dokumentech

Při hledání oblasti v dokumentech se výsledky seřadí podle počtu nalezených termů v oblasti a jako druhé řazení se používá rozptyl těchto termů.

Při přepnutém řazení na datum se první seřadí seznam podle počtu nalezených termů a jako druhé řazení se použije datum vysílání. Informace o dokumentu jsou obdobné s předchozím obrázkem. Rozdíl je jen v předposledním a posledním čísle, která udávají počet termů v oblasti a rozptyl termů.



Obrázek 3-10 – Informace o nalezených výskytech v oblastech dokumentů

Rychlost nalezení výsledků

Pro obyčejného uživatele jsou skryty vnitřní procesy a funkcionalita probíhající uvnitř vyhledávače. Jediné co skutečně uživatel pociťuje je doba, kterou musí čekat na výsledky hledání. Mohlo by se na první pohled zdát, že si jsou všechny hledané dotazy rovnocenné, ale není tomu tak. Doba hledání je souhrnem časů nejrůznějších událostí, které se schovávají za celkový čas vyhledávání.

Přímý vliv na dobu hledání má samozřejmě volba algoritmu. Algoritmus má určitou složitost a náročnost, s každým přidaným termem vzrůstá doba provádění algoritmu. Také má samozřejmě vliv četnost výskytu slov, ale pro srovnání našich dvou metod můžeme tento vliv zanedbat. Předpokládejme, že pro účely porovnání použijeme stejné termy pro oba algoritmy. Výsledné časy nejsou nikterak překvapující. Předpokládali jsme, že pomocí metody *best match* bude vyhledávání stále rychlé, což se potvrdilo. Doba hledání devíti termů je přibližně stále stejná a pohybuje se okolo 1

sekundy. Kdežto hledání pomocí metody oblasti je s každým termem stále horší a horší. Náročnost algoritmu roste zhruba exponenciálně s počtem termů. Je to dáno tím, že metoda prochází tutéž tabulku tolikrát, kolik řádků tabulka má. Z původních hledaných dvou termů za 3.5 sekundy se čas hledání zhoršil na 22.77 sekund při devíti termech. Jak se dále přesvědčíme, za přesnost nalezených výsledků se vždy platí časem.

Uživatelé, kteří jsou již zkušenými hledači, ví, že k rychlému a úspěšnému nalezení chtěných informací je klíčem správná volba slov. Bude-li se dotaz skládat z obecných, často se vyskytujících slov, vyhledávač najde množství nic neříkajících nepřesných míst. Term, jakožto slovo, má tím větší informační hodnotu čím méně se vyskytuje. Volbou specifických, originálních, méně často se vyskytujících slov se uživatel dostane mnohem snáze k cíli hledání, zároveň pro vyhledávač resp. jeho algoritmickou část je hledání poněkud méně náročné.

Světové vyhledávače využívají tzv. stop list, který při větším počtu termů znevýhodní či úplně zastaví vyhledávání velmi často[8] se vyskytujících termů. Tyto termy nenesou žádnou informační hodnotu, ale pouze znehodnocují výsledky hledání.

Někdy je užitečnější lépe proškolit hledače, než vylepšovat vyhledávače.

Pokud uživatel má již představu, co chce hledat a tuší jaká správná slova použít, není nic jednoduššího, než zadat slova do vyhledávače a najít požadované informace. Je pravdou, že při použití omezujících podmínek u multimediálního vyhledávače dostane uživatel přesnější výsledky. Je ale také pravdou, že ne všechny druhy omezení jsou stejně časově náročné. Jedná se především o volbu mluvčích, témat a skóre. Tyto parametry jsou vázány velkými tabulkami, s kterými se při zadání těchto omezení musí pracovat. Je to dáno zvolenou strukturou databáze. Jistě by se našel i jiný způsob řešení struktury databáze, ovšem tento má výhodu snadné rozšiřitelnosti. Omezení pomocí mluvčích, témat či skóre jsou však tak výhodná, že se vyplatí je i tak použít.

Ostatní omezující podmínky jsou velmi efektivní a časově nenáročné.

[8] Slovy ‚velmi často se vyskytujících' je myšleno slova (termy), která se nacházejí na prvních místech frekvenčního slovníku. Výskyt těchto slov výrazně převyšuje výskyt průměrného slova.

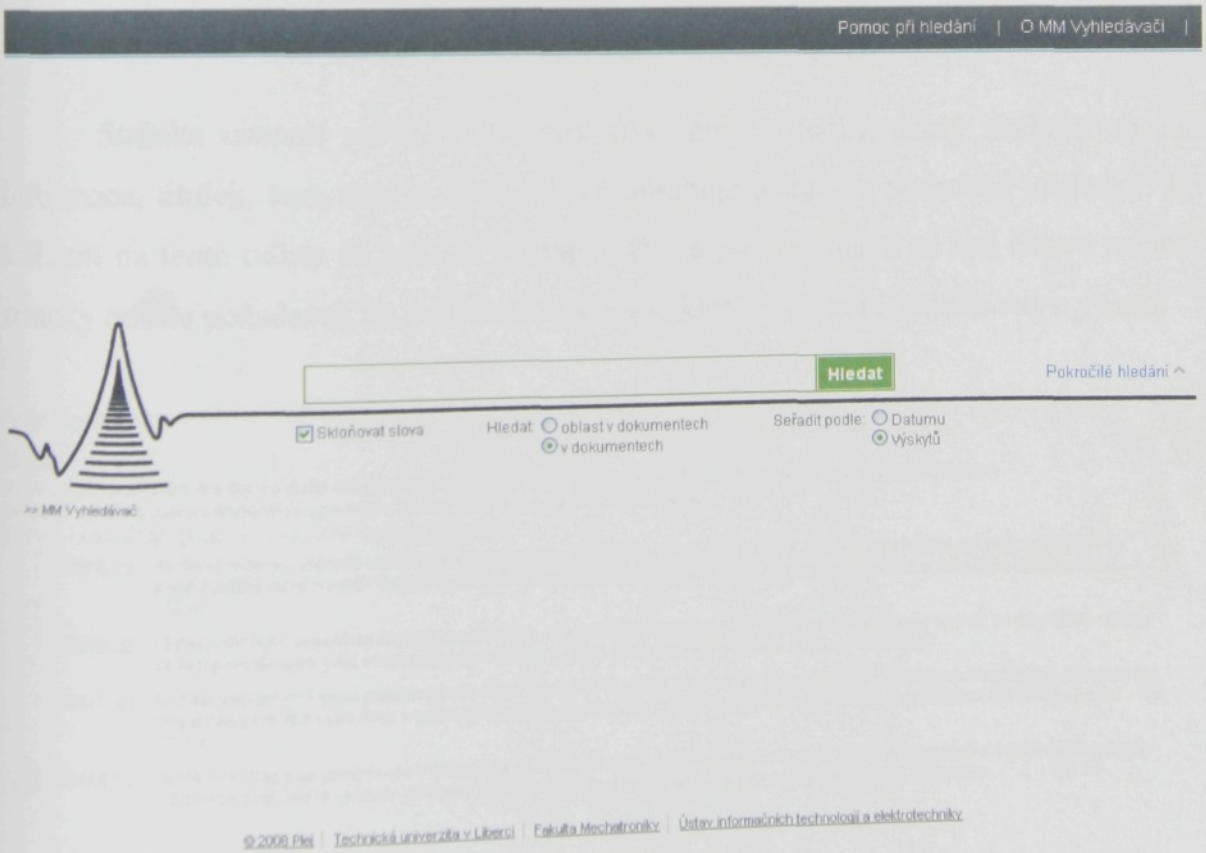
3.6 Uživatelské rozhraní

Uživatelské rozhraní je navrženo co nejjednodušeji a nejstandardněji, aby se v něm uživatel dobře orientoval. Použity jsou efektové prvky z ajaxové knihovny JQuery. Stává se, že uživatelé na tyto efekty nejsou zvyklí a tak občas jednají zbrkle a opakovaně klikají na odkazy v domnění, že stránka nereaguje. Jedná se především o efekt načítání dat bez znovunačtení stránky. Tomuto nedorozumění uživatele s prostředím lze zabránit velmi jednoduše, a to znázorněním symbolu, který značí, že systém pracuje a načítá.

Uživatelské rozhraní je nezávislé na aplikační vrstvě. Použit byl templatovací systém Tenjin pro jazyk Perl.

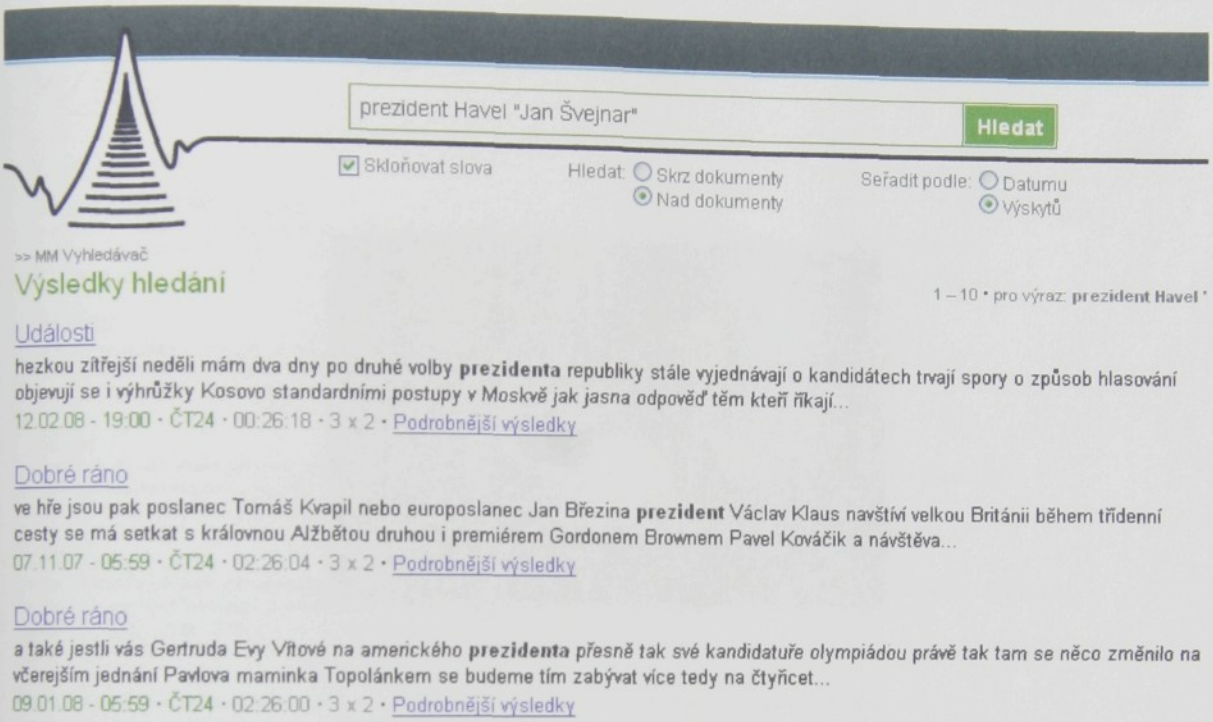
3.6.1 Stránky

Struktura stránek je velmi jednoduchá. Základem všech stránek je univerzální tělo webového dokumentu tzv. *layout*. Tento layout slouží jako hostitel pro další vnořené stránky. Při načtení základní stránky *default* se používá hlavička pro hledání *head_search*. Tato stránka má charakter uvítací stránky.



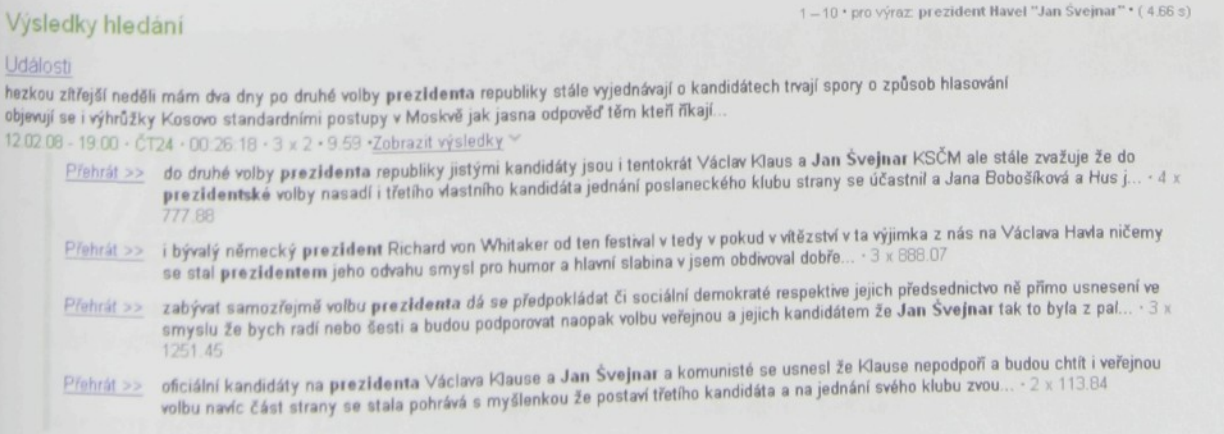
Obrázek 3-10 – Uvítací stránka MM vyhledávače - default

Po zadání dotazu pomocí jednořádkové syntaxe nebo pomocného pokročilého hledání odešle uživatel dotaz na databázi, která vrátí výsledky hledání. Pro tyto výsledky se načte stránka *content* také s hlavičkou *search_head.* Tato stránka slouží jako hlavní stránka zobrazující výsledky hledání.



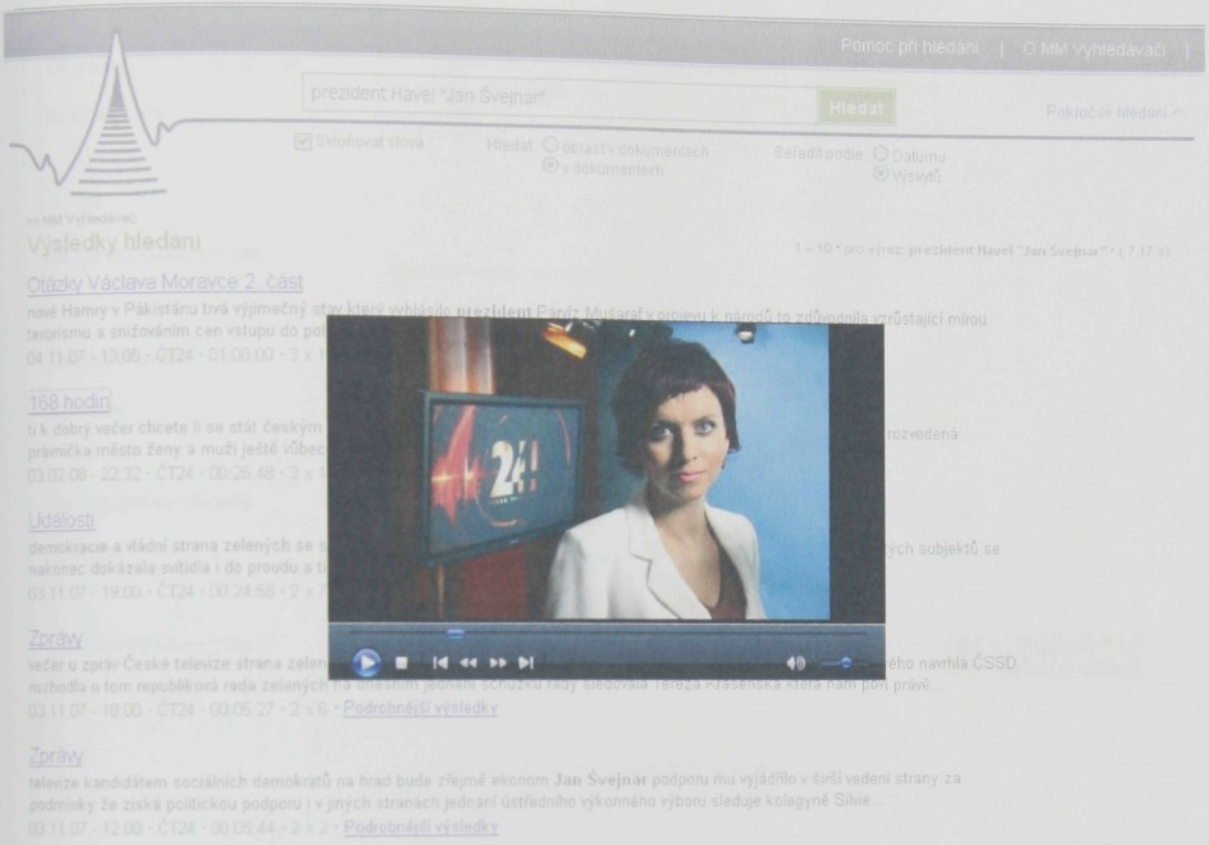
Obrázek 3-11 – Hlavní stránka vyhledávače – kontent

Stránka *content* má základní část blok pro výsledky, který osahuje obecné informace, titulek, text, datum, kanál. Také obsahuje odkaz na podrobné výsledky. Po kliknutí na tento odkaz se zobrazí spinner a pomocí ajaxu se bez znovu načtení stránky odešle požadavek na podrobné výsledky, které zobrazuje stránka *more_result.*



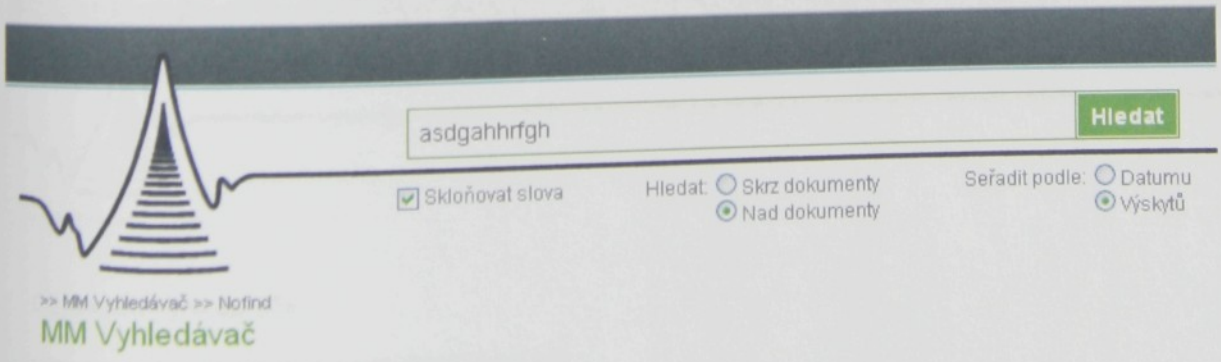
Obrázek 3-12 – Podrobnější výpis nalezených míst – more_result

Po kliknutí na odkaz se obrazovka prohlížeče zesvětlí a uprostřed prohlížeče se spustí požadované video přesně v místě, které uživatel hledal. Při dvojkliku se video roztáhne na celou obrazovku. V dolní části jsou pak připojeny titulky. Po přehrání stačí kliknout na klávesu escape, nebo kliknout mimo video.

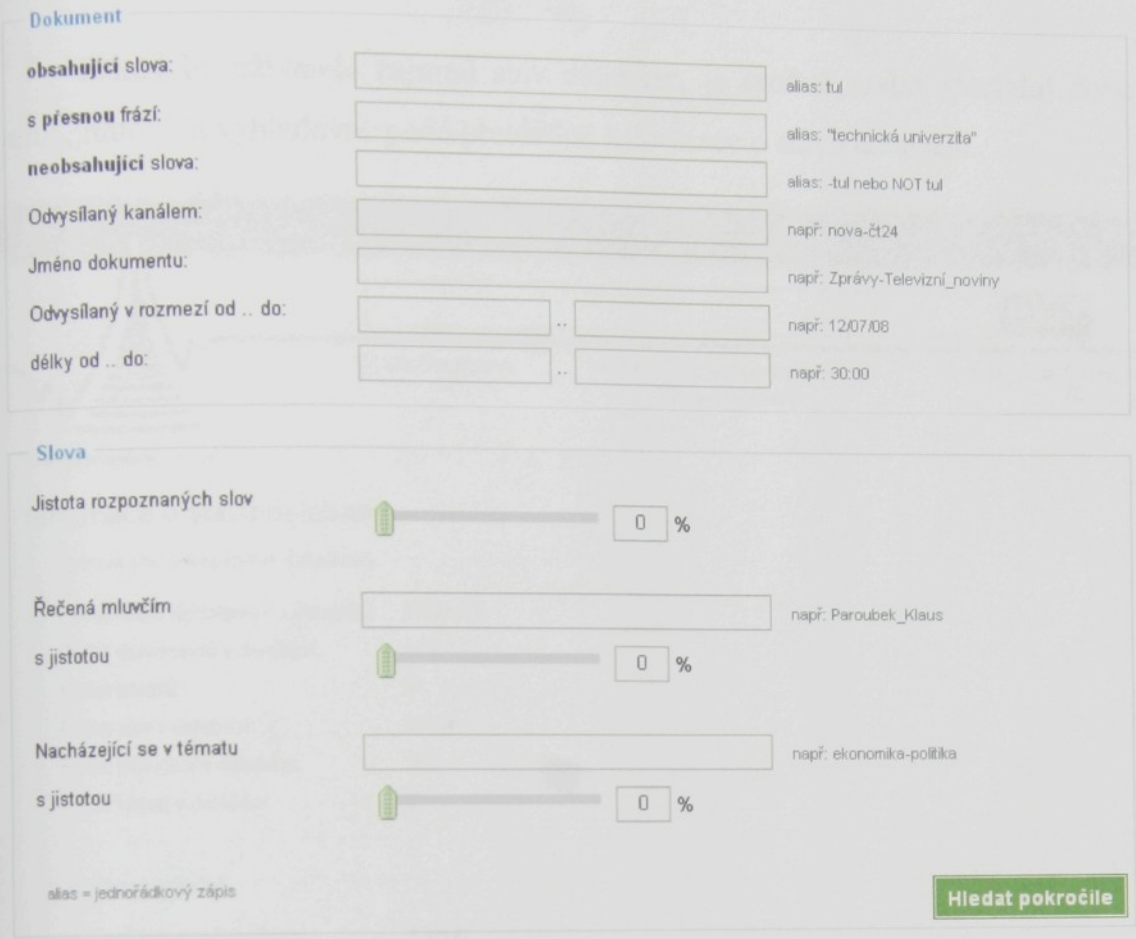


Obrázek 3-13 – Přehrávání videa

Pokud uživatel použije špatnou syntaxi, nebo nejsou pro hledané termy žádné výsledky, zobrazí se stránka *nofind*, která uživatele požádá o jinou kombinaci hledaných termů, nebo použití *pokročilého hledání*.

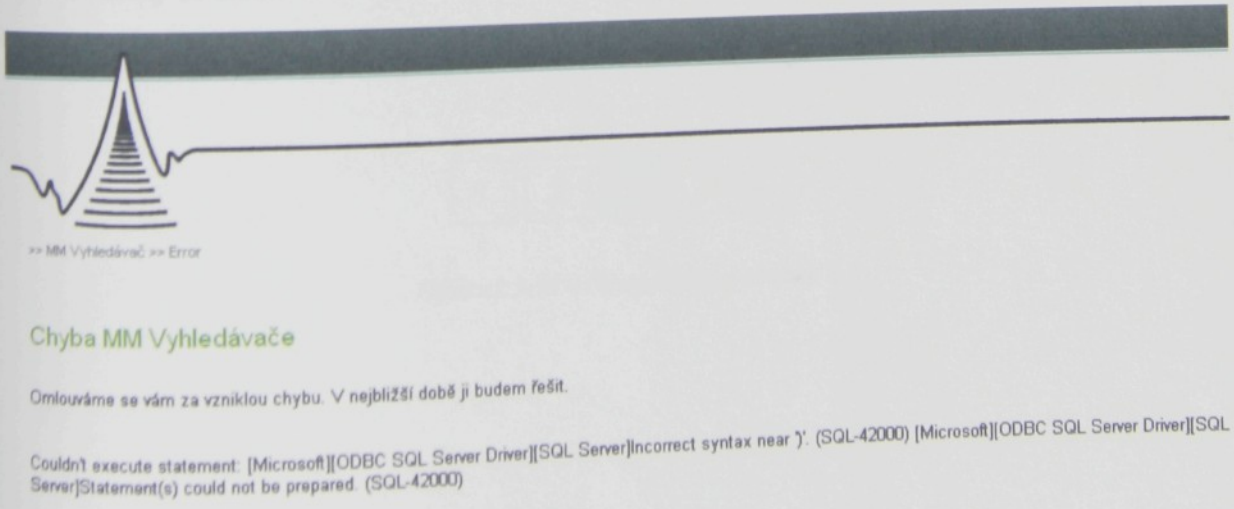


Obrázek 3-14 – Stránka pro nenalezené výsledky – nofind



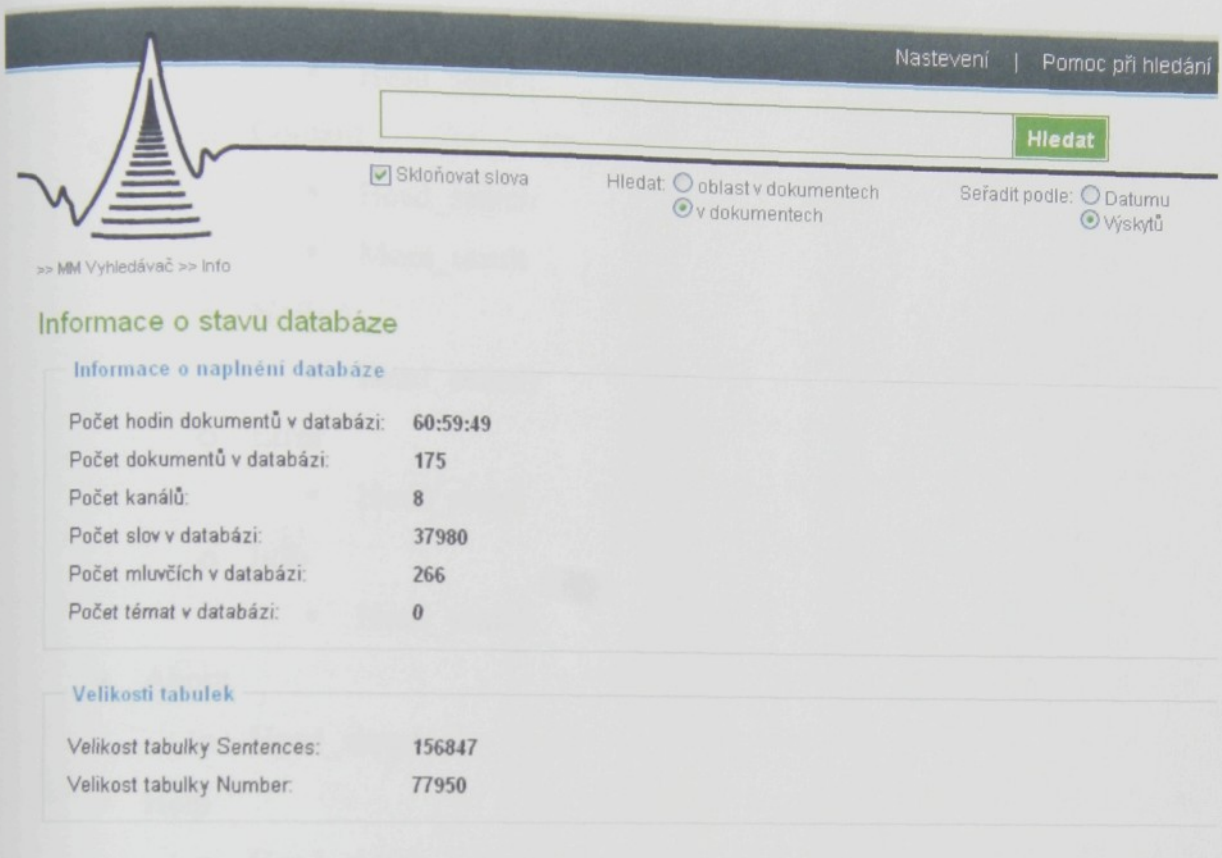
Obrázek 3-15 – pokročilé hledání

Nastane-li problém ve vyhledávači, například v případě výpadku databázového serveru, či jiná událost, zobrazí se stránka *error* s hlavičkou *search_simpl* bez možnosti zadávat dotaz. Pravděpodobně vznikl někde problém ne vinou uživatele a bude nejlépe počat, než se problém opraví.



Obrázek 3-16 – Stránka s chybovým hlášením – error

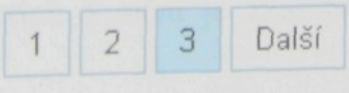
Pokud by uživatele zajímal stav databáze, je možné zaslat specielní dotaz ve tvaru „::info::“ a vyhledávač podá přehledné informace o stavu databáze.



Obrázek 3-17 – Stránka s informacemi o databázi – info

Stránky *about* a *help* jsou součástí stránky *layout* jako záložky. Při kliknutí na tyto záložky se načte obsah záložek z externích souborů *about* nebo *help*.

Stránka *content* při výskytu více výsledků obsahuje v dolní části navigační blok.



Obrázek 3-18 – Navigační část stránky

Schéma propojení a struktury templatů je znázorněno následujícím víceúrovňovým seznamem.

- Layout
  - Default
    - Head_search
  - Content
    - Head_search
    - More_result
  - Nofind
    - Head_search
  - Error
    - Head_simpl
  - Info
    - Head_search
- About
  - Head_simpl
- Help
  - Head_simpl

3.7 Uvedení systému do provozu

Zkušební verze systému byla uvedena do provozu 1.5.2008 a byla instalována na školním počítači.

V průběhu testování se ukázaly drobné obtíže při nahrávání velkých souborů. Proto byl upraven načítací skript. Systém je zkušebně dostupný online na adrese http://quadira.ite.tul.cz

4 Možnosti budoucího zlepšení

Multimediální vyhledávač, který jsem navrhl, je možno nadále vyvíjet a jsem přesvědčen, že má budoucnost. Zejména by bylo vhodné jej rozšířit o zpřesňující metodu k základnímu vyhledávání *best match*. Toto vyhledávání je velmi rychlé, ale mohlo by sloužit jako předvýběr a poté by bylo možno provést přesnější metodu na úrovni aplikační vrstvy.

Pro implementaci náročnější metody by bylo třeba jít dále, než umožňuje databázová vrstva, a využít výkonnostní možnosti vrstvy aplikační. Pro získání výsledných dat potom bude pravděpodobně nutné použít nikoli jeden, ale dva dotazy. První bude sloužit pro výpočet nejlepších míst v dokumentech, druhý bude již jen vyzvedávat výsledky hledání a jejich slovní okolí. Výhodou tohoto postupu by mohlo být snadné uložení výsledků s následnou možností rychlého zobrazení dalších stran.

V zásadě by samozřejmě bylo možno použít i dotaz jediný, ve kterém by vše potřebné bylo vybráno předem. Nevýhodou tohoto postupu by však - při vzrůstajícím množství informací - byly příliš velké datové přenosy. Při zpracování by hrozilo nebezpečí, že bude docházet k zahazování velkého množství dat vyhodnocených jako nepotřebná. Tímto postupem by výpočetní a přenosové prostředky byly využívány neefektivně.

Do budoucna jistě vyvstane také otázka množství dat. Při jejich růstu pak pravděpodobně bude nutné systém distribuovat mezi více autonomních uzlů (počítačů) a vytvořit tak vnitřní síť. Podmínkou této sítě bude, aby ji uživatel stále mohl vnímat jako jeden celek.

Moderní webové prostředí se již neobejde bez použití ajaxové technologie. Tato technologie má v sobě velký potenciál pro zpříjemnění uživatelského prostředí. Poskytuje možnosti, umožňující celou řadu dalších vylepšení.

Namátkou lze jmenovat například proužek, který by naznačoval, ze které části dokumentu je výsledek hledání vzat. Výborným pomocným prvkem by též mohl být našeptávač, který by nabízel, jaké dotazy již byly hledány.

Věřím, že velmi užívanou funkcí by byla také nabídka přepisu, tj. možnost nechat si celý dokument vypsat v textové podobě. Všechna tyto zlepšení se týkají spíše uživatelského komfortu, tedy je to záležitost webových technologií, předně ajaxu.

Další zlepšení, o nichž je možné uvažovat, se mohou týkat aplikační vrstvy. Zajímavou pomůckou by jistě byl opravovač překlepů. Ve většině případů jde o jednopísmenné překlepy či vynechání písmene. Pro řešení těchto problémů jsou využívány statistické metody. Problémem bývá časová a výkonnostní náročnost těchto algoritmů.

Věřím, že budoucnost ukládání a efektivního vyhledávání v textových dokumentech leží v xml databázích. Pravdou je, že tyto databáze v dnešní době ještě nedosahují takové úrovně, aby zvládaly všechny funkce a možnosti klasických databázi, nicméně začínají být zajímavou alternativou do budoucna. Vývoj jde stále vpřed a spolu s ním i možnosti.

Budoucnost je otevřena těm, kdo jsou připraveni zkoušet nové způsoby a metody.

Závěr

Hlavním cílem diplomové práce bylo vytvořit funkční multimediální vyhledávač. Tento cíl se podařilo splnit. Beta verze multimediálního vyhledávače funguje online na adrese http://quadira.ite.tul.cz

Základem diplomové práce bylo úspěšné zprovoznění MS SQL databáze 2005 a navázání komunikace přes programovací jazyk Perl.

Navrhl jsem strukturu databáze pro ukládání multimediálních dat s možností rozšíření. Vytvořil jsem transformační skript a zjednodušenou strukturu xml dokumentu pro plnění databáze. Naprogramoval jsem kompletní search engine s dvěma implementovanými vyhledávacími algoritmy, které pracují na úrovni databáze.

Podařilo se mi vytvořit zjednodušený jazykový model češtiny pro ořezávání koncovek slov, který přes svoji jednoduchost pracuje s poměrně vysokou úspěšností. Kompletně jsem navrhl a vytvořil uživatelské prostředí s propracovanou strukturou. Uživatelské prostředí je koncipováno jako user-friendly.

Vytvořil jsem syntaxi dotazovacího jazyka, který je používán uživatelem pro zadávání dotazů, je možno též použít pokročilého hledání, které usnadňuje uživateli zadávání omezujících podmínek.

Vyhledávač disponuje širokými možnostmi omezujících podmínek, díky kterým lze zpřesnit hledané dokumenty. Zapojeny byly nejmodernější prvky webové technologie.

Vyhledávač bude vhodné rozšířit další vrstvou, ve které by se prováděly náročnější algoritmické výpočty. Uživatelské prostředí má velký potenciál pro rozšíření o další funkcionality.

Seznam použitých zdrojů

[1] Wall, Aaron: *History of Search Engines: From 1945 to Google 2007,* [online], 2007
http://www.searchenginehistory.com/

[2] SpeachLab: *Laboratoř počítačového zpracování řeči* [online]
http://itakura.kes.tul.cz/

[3] Hyoung-Gook, Kim: *Nicolas Moreau, Thomas Sikora; MPEG-7 Audio and beyound*, England 2005, ISBN 0-470-09334-X

[4] kolektiv autorů, Grepl, Hladká, Jelínek, Karlík, Krčmová, Nekala, Rusínová, Šlosar: *Příruční mluvnice češtiny*, Brno 2003, Nakladatelství Lidové noviny, ISBN 80-7106-134-4

[5] Sparck, Jones K; Walker, S; Robertson, S.E.: *A probabilistic model of informatik retrival: development and komparative experiments*, Londýn leden 2000, [online]
http://www.soi.city.ac.uk/~ser/blockbuster/pmir-pt1-reprint.pdf

[6] Croft, Bruce: Exact-Match Retrieval, 24.2.98, [online]
http://www.cpe.ku.ac.th/~arnon/Mirror/ir-p/ir6/index.htm

[7] Hiemstra, D.; Robertson, S.: *Relevance Feedback for Best Match Term Weighting Algorithms in Information Retrieval*, Cambridge, [online]
http://www.ercim.org/publication/ws-proceedings/DelNoe02/hiemstra.pdf

[8] Červa, P.: přednášky - *Pravděpodobnost a teorie informace*, Liberec 2007

[9] CPAN Perl [online]
http://www.cpan.org/

[10] MS SQL server 2005 [online]
http://www.microsoft.com/sql/

[11] MySQL [online]
http://www.mysql.com/

[12] Ajax [online]
http://www.ajax.cz/

[13] jQuery [online]
http://jquery.com/

[14] Tenjin [online]
http://www.kuwata-lab.com/tenjin/

Příloha [A] – schéma databáze

[1,1]