Vyhledávání v Internetu
Searching in Internet

Bakalářská práce

Lukáš Křemen
Vedoucí „závěrečné“ práce: Ing. Ladislav Beránek, Csc. , MBA
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské/diplomové práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Anotace

Tato Bakalářská práce zahrnuje problematiku tvorby vlastního crawleru a psaní webových stránek tak, aby crawler snadněji separoval slova, z těchto stránek. Dále se zabývá popisem Internetu jako grafu, indexováním

a charakterizací robotů jako je Google, nebo seznam.

Abstract

This work includes issues of creating one's own crawler as well as question of writing web pages in a way for the crawler to easier separate words from these pages. In then focuses on describing the Internet as a graph, indexing and characterizing robots such as Google or Seznam.

Poděkování

Rád bych poděkoval panu Ing. Ladislavu Beránkovi, CSc. , MBA, za vedení, a rady při psaní mé bakalářské práce.

Obsah

1 Úvod

Dnes již určitě téměř každý uživatel Internetu ví, co je to „vyhledávač“. Ovšem málokdo se již zabývá jak vyhledávače pracují, a nebo jak si napsat svůj vlastní vyhledávač. Tato bakalářská práce se snaží nastínit jak se dá při psaní vlastního vyhledávače postupovat. A jaké problémy to obnáší. Z určité části se také zabývá již mnohokrát popsaným tématem správné tvorby webových stránek. V této práci naleznete ukázku webového crawleru, který má možnost vyhledávat emaily, a nebo zadaný text. Uživatel má zde možnost spustit vyhledávání ihned, nebo vyhledávání naplánovat. Tento crawler ukládá nalezené emaily a odkazy, na kterých byl nalezen, do databáze. Pokud je crawler v režimu vyhledávání slov, ukládají se do databáze odkazy, na nichž byl zadaný text nalezen. Vzhledem k velikosti databáze, která by byla třeba a také k problematice indexování, je tento crawler prost indexování a vyhledává informace takzvaně za běhu. V teoretické části se též věnuji objasnění základních pojmů a principů ohledně Internetu. Dále se zabývám samotnou problematikou vyhledávání a indexování. Což z části zahrnuje i již zmíněná doporučení ohledně tvorby www stránek. Ale také jiné jako například rozpoznávání textu, zpracování dotazu, nebo indexy. Dalším objektem zájmu jsou roboty. Kde se snažím pomocí veřejně dostupných informací popsat, jak roboty pracují obecně a také nastínit jak pracuje google a seznam. Nakonec popisuji praktickou realizaci vyhledávacího robota, která tvoří praktickou část. Využívám při tom poznatky teoretických částí o vyhledávání, separování a další.

2 Popis Internetu

Internet je celosvětově rozšířená počítačová síť. Uživatel se dostane k informacím na konkrétní webové stránce, ale jen pokud zná její adresu. Kvůli tomu se však dostane jen k jedné stránce. Přístup k webovým stránkám pomocí jmenné adresy na kterou jsme zvyklí, je obstaráván takzvaným DNS. Domain name system zajišťuje pomocí DNS serveru a DNS protokolu výměnu informací, čímž dochází k převodu doménových jmen na IP adresy a naopak. Doménová jména mají určitou hierarchii, skládají se z několika částí, které jsou oddělené tečkami. Nejobecnější částí doménového jména je tld (top level domain), ta část na konci vpravo. Nazývá se také doména nejvyšší úrovně. Tato doména je buď generického typu neboli com, net, gov, atd.. Nebo představuje geografickou doménu jako cz, sk, de, ru, atd. Další části směrem doleva jsou více a více konkrétnější adresou k serveru, na kterém se nachází stránky, které jsou tímto doménovým jménem definovány.

Aby se uživatel dostal k informaci, kterou potřebuje musí projít řadu konkrétních adres. To bylo možné na začátku internetu. Aby se usnadnilo vyhledávání konkrétní informace na internetu, proto vznikly vyhledávací mechanismy, pro usnadnění vyhledávání informací v této oblasti.

2.1 Internet jako graf

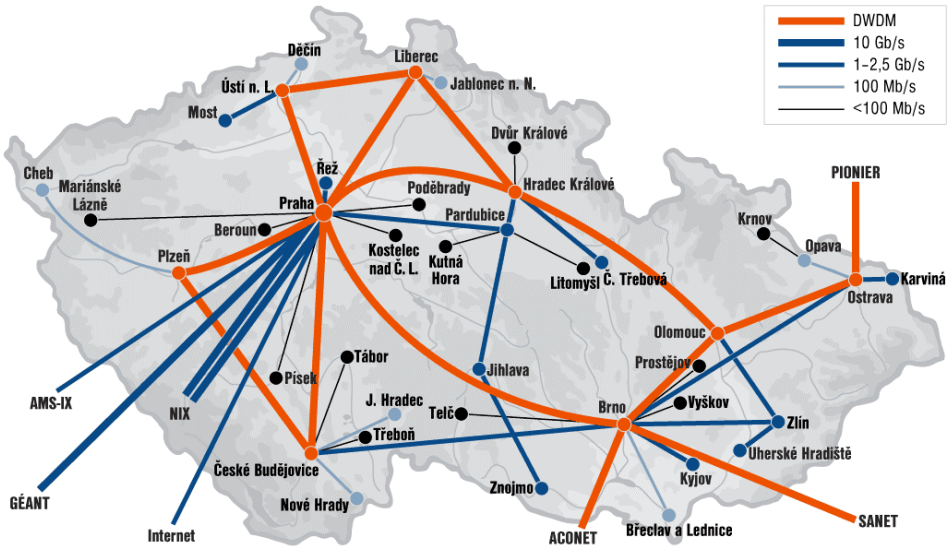
2.1.1 Původní idea Internetu

Internet byl původně určen pro vojenské účely. Armáda postrádala účinnou a rychlou komunikaci během nepřátelského útoku. Vyvinula tedy počítačovou síť, která neměla jen jedno propojení, ale hned několik. Aby při

jednom přerušení mohli ostatní komunikovat i nadále. Tato síť se jmenovala Arpanet. Po nějaké době se Arpanet rozšířil i do Akademické oblasti a později i mezi běžné uživatele. Dnes se této síti říká Internet.

2.1.2 Struktura Internetu jako síť

Struktura této sítě se dá vyjádřit jako graf o velikém počtu vrcholů a hran. Hlavní cesty těchto grafů jsou ve většině případů zajištěny páteřními sítěmi. Existují mezikontinentální páteřní sítě a na tyto mezikontinentální páteřní sítě jsou dále napojeny páteřní sítě, které jsou spravovány jednotlivými organizacemi, které se na daných kontinentech nacházejí. Takto vypadá česká páteřní síť Cesnet2, převzato z [10].



Obr.1: Síť Cesnet2

Cesnet2 je akademická síť. Akademické a komerční sítě, které jsou součástí Internetu, se často prolínají. Cesnet2 má na obrázku vyznačeno propojení s komerčním Internetem který je na obrázku vyznačen slovem Internet. Toto propojení vede přímo do USA jejímž dodavatelem je Telia. Na obrázku jsou vidět i další páteřní sítě, které vedou do sousedních států. Například SANET Slovenská páteřní síť, nebo ACONET páteřní síť Rakouska. Páteřní síť Géant, která je též znázorněna na obrázku, představuje páteřní síť Evropy. Tyto sítě jsou ovšem ryze akademického původu. K propojení s komerčním Internetem a k propojení běžných uživatelů s Internetem z domova, využívají poskytovatelé takzvaná peeringová centra.

2.1.3 Peering

Je propojení počítačových sítí v centrálních bodech, kterým se říká peeringové uzly. Peering se dále dělí na privátní a veřejný. Privátní peering představuje přímé propojení poskytovatele internetových služeb pomocí zvláštní datové linky ke každému sousedovi. Veřejný peering znamená, že vede datová linka jen do peeringového centra.

2.1.4 Peeringová centra

Peeringové centrum je určitý bod propojení například poskytovatelů internetových služeb. Propojovací místa jsou vzájemně propojena, poskytovatelé si tak mohou vybrat odkud se připojit. Díky tomu při výpadku jednoho uzlu, všechny zbylé fungují. Peeringová centra se vyskytují po celém světě. Pro příklad je zde výpis některých z nich.

NIX – Praha, Česká republika

SIX - Bratislava, Slovenská republika

Linx – Londýn, Velká Británie

Parix – Paříž, Francie

TORIX – Toronto,Kanada

MSK – IX – Moskva, Rusko

2.1.5 Graf

Pokud bychom si představili Internet jako graf, dalo by se pomocí ohodnocených hran. Ohodnocení například podle nákladů na provoz, nebo vzdálenosti. Určovat nejefektivnější propojení počítačových sítí, nebo Internetových poskytovatelů. Na takovýto graf by se po té daly použít i matematické metody hledající nejkratší cestu grafem, jako je Dijkstrův algoritmus.

2.2 Internet jako informační médium

2.2.1 Informační služby

První z informačních služeb Internetu je Telnet, která vznikla ještě před rozšířením osobních počítačů. Telnet je emulací terminálu a terminál je střediskový počítač. Což sebou přináší značné nevýhody z pohledu současného uživatele. Uživatel si na PC musí pro napodobení terminálu nainstalovat speciální programové vybavení, které simuluje terminál a jeho funkce. Což znamená ovládání pouze v textovém režimu a nepoužívání myši. K ovládání slouží pouze klávesnice. Další službou byl Gopher, který představoval první službu navigačního typu. Tato služba umožňovala přístup k informačním zdrojům, aniž by uživatel musel znát konkrétní umístění. Gopher pracoval

s hierarchickými nabídkami. Práce s nabídkami byla pro jakéhokoliv uživatele snadno pochopitelná. Gopher umožňoval svůj vlastní prostor, který byl vytvářen externími odkazy z nabídek a byl omezený pouze hranicemi Internetu.

2.2.2 Uspořádanost

Nejprve je nutné říci, že Internet je svou strukturou informačně neuspořádané médium. Ačkoliv se různé předmětové katalogy snaží této neuspořádanosti čelit. Přesto díky jisté volnosti, že kdokoliv může publikovat na Internetu informace dle svého uvážení, bude vždy zůstávat jistý prostor, který se nepodaří uspořádat a dostat pod veřejnou kontrolu. V současné době je ještě stále hluboký neuspořádaný web větší oproti známému a indexovatelnému webu vyhledávači. I do budoucna bude samozřejmě snaha zmapovat pokud možno celý webový prostor. Již dnes se za tímto účelem používají jisté pomůcky. Vyhledávače nabízejí většinou zdarma vyplnění registračního formuláře tvůrcem stránek. Tvůrce zde uvede popis svých stránek, jejich adresu a v některých případech i předpokládanou frekvenci změn stránek. Indexovací robot bude mít tyto stránky ve svém seznamu indexovaných adres a pokud není web napsán nepřijatelnou formou, bude příště zahrnut do vyhledávačem zmapovaného webového prostoru. Z hlediska volné publikace na Internetu je na místě rozhodovat se, která data jsou pravdivá a která ne. Poněvadž nyní už nepublikují informace jen vědecká pracoviště a firmy. Dnes může zveřejnit své názory každý jednotlivec, jenž může být mylně informován, něco opomene, nebo dokonce záměrně lže ve svých publikacích.

2.2.3 Velikost Internetu

V roce 2007 je velikost známého (zmapovaného) webu odhadována na 3 miliardy webových stránek. Nezmapovaných stránek bude pravděpodobně ještě víc. Tento počet se samozřejmě den ode dne mění. Stále ještě přibývají noví uživatelé i nové www stránky. Stále více si začínáme zvykat na vymoženosti, které nám Internet oproti papírové podobě informací nabízí. Například možnost přiložit video k článku o kterém píšeme. Či nějaké animace, nebo nejčerstvější zprávy. Ani nemluvě o šetření stromů k výrobě papíru potřebných. Důsledkem toho lze očekávat i další nárůst uživatelů.

3 Vyhledávání, Indexování

Vyhledávání potřebných informací ve webovém prostoru, pouze pomocí webových adres, je vzhledem k rozsáhlosti webového prostoru, a množství webových stránek nemyslitelné. Je proto zapotřebí použít vyhledávací stroje. V současnosti existuje přes 3 000 těchto vyhledávacích strojů, jiné zdroje naopak uvádějí přes 18 000. Každopádně navzdory těmto číslům není vyhledávání informací v prostředí Internetu nijak dokonalé. Výsledky dotazu nejsou ohodnoceny pouze podle PageRanku, ale také podle pozice hledaného slova v dokumentu. Google navrhl své hodnocení tak, aby žádný jednotlivý faktor nedokázal ovlivnit výsledek příliš velkou měrou. Při vyhodnocování jednoslovného dotazu se zkoumá seznam hitů pro dané slovo. Google si ukládá u každého hitu jeho titulek,URL, text odkazu, obyčejný text malým i velkým písmem. Každý z těchto druhů má pak přiřazenou určitou váhu. Stejným způsobem je ohodnocován i počet hitů pro každý druh. Ohodnocení je zpočátku lineárního charakteru, ale jakmile přeroste určitou mez ohodnocení dále neroste. Ohodnocení relevance se vypočítává skalárním součinem vektoru vah a vektoru ohodnocení počtu výskytů. Kombinací tohoto součinu s PageRankem určuje finálové pořadí dokumentu ve výsledcích. Komplikovanější situace však nastává u víceslovných dotazů. Zde se ohodnocují výsledky na základě vzdálenosti jednotlivých výskytů. Při tomto ohodnocování musíme procházet několik seznamů hitů najednou. U každé nalezené skupiny hitů je spočítána vzdálenost výskytu všech nalezených slov v textu dokumentu a této vzdálenosti se přiřadí jedno z deseti ohodnocení. Dále se počítají výskyty nejen pro jednotlivé druhy hitů, ale také pro dvojice druh a vzdálenost. Tyto údaje jsou dále převedeny na náležející ohodnocení, jejichž skalární součin představuje ohodnocení relevance dokumentu.

3.1 Architektura

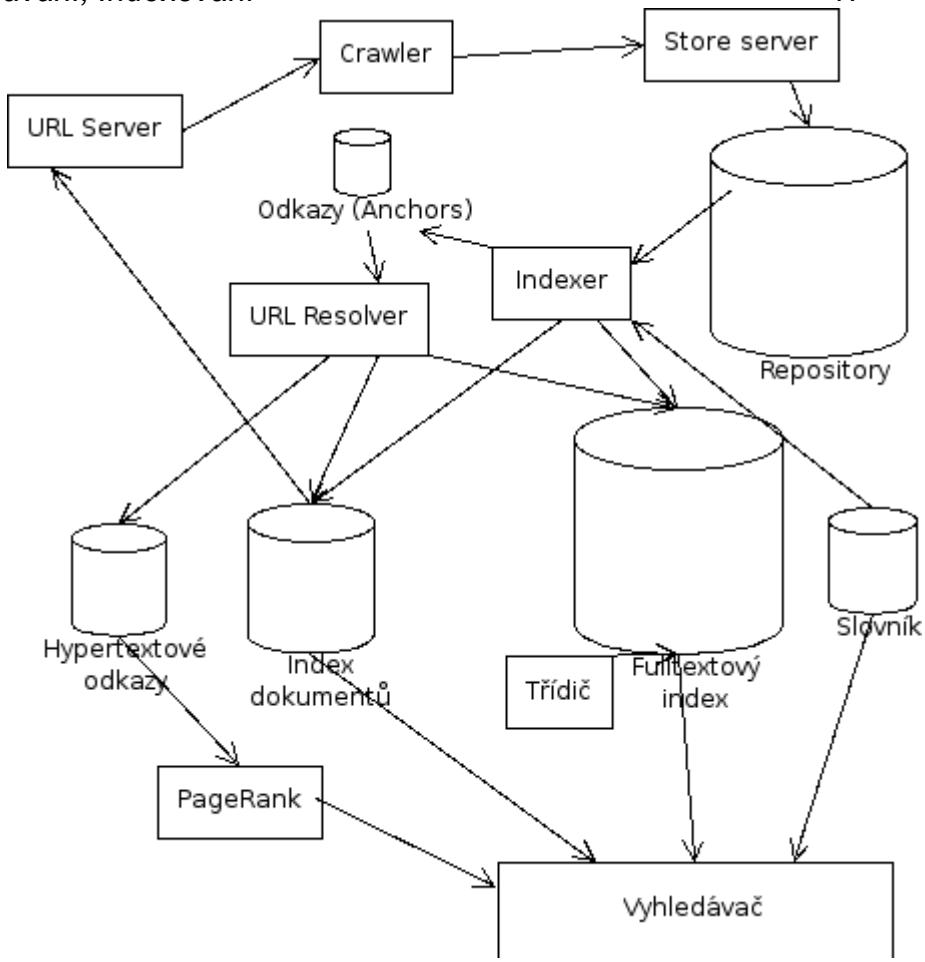
Vyhledávací stroj je tvořen třemi částmi vyhledávacím softwarem(spider, crawler), indexovacím softwarem a softwarem pro vyhledávací engine. Každá z těchto úloh může být prováděna paralelně. Crawler si nejprve zpracuje svou iniciační množinu URL a vyhledá na této množině nové odkazy na webové stránky. Zároveň na těchto stránkách vyhledá klíčová slova, která jsou přidána do indexu nebo katalogu vyhledávacího stroje. Mezi důležité aspekty každého indexu vyhledávacího stroje patří pokrytí indexy, frekvence obměňování a obsah indexovaných polí. Odpověď na dotaz, kterou vrací program vyhledávacího stroje, je uspořádána podle relevance v indexu, kde se většinou nachází i milióny zaznamenaných stran. Vyhledávací stroj obsahuje distribuované crawlery, tyto crawlery pak stahují dokumenty z URL webových stránek, toto URL určil vyhledávacímu stroji URL server. Crawlerů dokáže běžet několik najednou, přitom každý udržuje naráz stovky otevřených spojení. Aby nebyl zdržován čekáním na odpovědi webserverů. Kvůli proměnlivosti internetového obsahu musí být crawler robustní a odolný vůči zvláštním případům, jedním z nich jsou online hry. Crawler každý dokument zkomprimuje a uloží do schránky. Každá stránka má svůj identifikátor. O vyhledání zkomprimované webové stránky a její následnou dekomprimaci se stará indexer vyhledávacího stroje. Který následně parsuje tyto dokumenty do sady hitů. Hit zaznamenává výskyt slova a jeho pozici v dokumentu a relativní velikost písma, kterou je slovo napsáno. Hity se ukládají do zásobníků a tím se vytvoří téměř setříděný index. Dále se z parsovaných dokumentů pomocí indexeru filtrují odkazy a ty se ukládají do předem určeného souboru. O každém odkazu jsou uloženy informace odkud kam vede a jeho text. Pomocí URLresolveru se zpracovává soubor s odkazy na URL a v něm se převádějí relativní cesty na absolutní URL a dále je ukládá do indexu dokumentů, jenž slouží jako zdroj dat pro URLServer. Do indexu se

také přidává ke každému dokumentu text odkazu, který na něj směřuje. Vzájemné odkazy pak slouží k výpočtu relevance daného dokumentu, jenž se nazývá PageRank. Tento ukazatel relevance využívá například Google. Vyhledávač běží na webserveru a pomocí slovníku, zpětného indexu a PageRanků odpovídá na dotaz. Vzhledem k tomu že je vyžadována co nejrychlejší odezva, je snaha číst co nejméně informací z disku.Jelikož pevný disk je mnohokrát pomalejší než přístup do paměti.

3.2 **Zpracování dotazu**

V datacentru je http požadavek přesměrován na webový sever (GWS google web server). Kde je koordinováno zpracovávání dotazů.

Uživatel na závěr od tohoto serveru dostane HTML odpověď, která vznikla zpracováním a formátováním dotazu již zmíněným webserverem. Při zpracování dotazu je nejprve nutné vyhledat odpovídající dokumenty ze zpětného indexu. Pro dosažení adekvátní rychlosti rozdělíme zpětný index na větší počet fragmentů, každému z těchto fragmentů náhodně přidělíme podmnožinu, kterou bude pokrývat. U každého fragmentu zpracovává požadavky více počítačů. Při zpracování požadavku se dotazy rozdělí mezi jednotlivé fragmenty, tak že vždy jeden stroj z každého fragmentu obdrží dotaz. Když všichni vyřídí svou část úkolu, spojí se v celkový výsledek. Tím se značně ušetří čas. Tímto postupem se celý index prohledá v čase prohledání jednoho fragmentu. Koordinaci dotazů má na starosti takzvaný load balancer. Ten zároveň hlídá, zda jsou všechny počítače funkční. Pokud jeden z nich selže, load balancer ho vynechá. Popřípadě jej nahradí jiným strojem.



Obr. 2: Struktura toku dat. Převzato z [2].

Na konci prohledávání by měl být setříděný seznam docID. Dále je nutné přiřadit ke každému docID URL, titulek a úryvek v textu, který odpovídá odkazu. Za tímto účelem jsou využívány dokumentové servery (docservers), ty hledají záznam ve schránce a získávají potřebné informace. Rychlost odezvy je opět řešena pomocí fragmentů jak již bylo uvedeno výše. Jakmile dorazí výsledné informace z docserverů začíná se s vyřazováním duplicit a clusterováním podle doménového jména. Google si pak v takovýchto clusterech uchovává několik kopií celého webu. Clustery se nacházejí na

dokumentových serverech a zajišťují těmito zálohami rychlost odezvy i dostupnost.

3.3 www komunity

Webový prostor by se dal rozdělit na autority a huby. Zatímco autorita se zabývá určitým tématem. Huby jen odkazují na několik autorit. Huby a autority spolu za jistých okolností tvoří takzvanou abstraktní komunitu. Tato komunita má na starosti při nějakém málo specifickém dotazu vracet určité množství webových stránek. Vyhledávací metoda má za úkol najít malé množství autorit, aby byl dotaz zodpovězen co nejpřesněji. Proto je však nutné prozkoumat odkazovou strukturu webu. Problematické však je, jak rozpoznat co je autorita. Jisté řešení tohoto problému navrhl Kleinberg[5], identifikovat huby a autority pomocí matice sousednosti podgrafu webu. Tento iterativní algoritmus nazval HITS (hyperlink induced topic search)[2]:

Pro hledání širokého dotazu algoritmus začíná s kořenovou množinou S stránek, které vrátí textový vyhledávací engine. Množina S indukuje malý podgraf zaměřený na vyhledávací téma. Tento vyvolaný graf je potom expandován tak, že zahrne všechny uzly, které jsou následníky každého uzlu v množině S. Navíc určité množství předchůdců každého uzlu množiny S je také zahrnuto. Nechť G je graf indukovaný uzly v této expandované množině uzlů. Je třeba poznamenat, že linky, které jsou využity čistě pro navigaci v rámci webových stránek nejsou zahrnuty do grafu G.

[2]:

Pro každý uzel x v grafu G ne-negativní autorita váhy a(x) a ne-negativní váha hubu h(x) se spočítá. Váha autority a hubu všech uzlů v grafu G

může být vyjádřena jako vektor a() respektive h(). Elementy vektorů a() a h() jsou inicializovány na jednu. V každé iteraci a(x) je nahrazeno sumou $h(x_i)$' všech uzlů i předcházejících uzlu x a h(x) je nahrazeno sumou $a(x_i)$' všech uzlů j následujících uzlu x. Iterace může být vyjádřena jako

$$a(x) = \sum_{v \to x} h(v)$$

a

$$h(x) = \sum_{x \to w} a(w)$$

$$a(x) = h(v_1) + h(v_2) + h(v_3) + h(v_4),$$

$$(v_i \in pred(x))$$

$$h(x) = a(w_1) + a(w_2) + a(w_3) + a(w_4),$$

$$(w_i \in succ(x))$$

A nyní normalizace autority a hubu [2]:

Autorita a hub jsou normalizovány v každé iteraci tak, že $\sum (a(x))^2 = 1$ a $\sum (h(x))^2 = 1$. Tento iterativní proces konverguje k nalezení vektoru autorit

a hubů pro původní dotaz. Jestliže M je matice sousednosti grafu G, potom iterativní kroky mohou být zobrazeny jako

$a = M^T.h$ a $h = M.a$

Tento krok může být zapsán jako

$a = M^T.h = M^T.M.a = (M^T.M).a$ a

$h = M^T.a = M^T.M.a = (M^T.M).a$

Tedy iterace vektoru a jsou ekvivalentní tomu, násobením původního vektoru a s mocninou $M^T.M$

Podobně vícenásobná iterace normalizovaného vektoru h konverguje k hlavnímu vlastnímu vektoru MM^T. Tedy HITS aplikuje link. založené výpočty pro identifikaci hubů a autorit v tématu dotazu.

3.4 Odpovídající URL a algoritmy hledající téma

Pokud budeme nazírat na topologii webu jako na graf, můžeme využít nových technik vyhledávání informací. Agenty , které by dokázaly obsáhnout linkovou strukturu webu, by mohly doplnit použití vyhledávacích strojů. Bohužel v současnosti je většina databází vyhledávacích strojů statického charakteru, což znamená, že vyhledávání aktualizovaných informací je obtížné. Využitím linkového přístupu by se dalo předejít ovlivňování výpočtu ohodnocení stránky, která je upravena pomocí inflace klíčových slov [2]:

Dean a Henzinger [8] vyvinuly nové vyhledávací paradigma založené na linkové struktuře webu. Navrhli algoritmus pro nalezení webové stránky odpovídající URL. Odpovídající webová stránka je ta, která adresuje stejné téma jako původní stránka, ale je sémanticky rozdílná. Kroky Dean-Henzigerova algoritmu jsou:

1. Vytvořit graf blízkého okolí pro dané URL, tj. uzel U

Graf blízkého okolí je graf s váženými hranami, který zahrnuje URL uzlu U, nahoře k B náhodně vybrané předchůdce uzlu U a pro každý předcházející uzel do Br následníka lišící se od U. Navíc graf zahrnuje F následujíc uzly uzlu U a pro každý následující uzel až do Fb jejich předcházejících uzlů odlišných od U. V grafu blízkého okolí existuje hrana, pokud existuje hyperlink z uzlu v do uzlu w, pokud uzly v a w nepatří stejné webové stránce.

2. Eliminace duplicit a skoro-duplicitních uzlů

Duplicitní uzly jsou stejné webové stránky na zrcadlících se místech nebo různé aliasy pro stejné webové stránky. O dvou uzlech řekneme, že jsou skoro-duplicitní uzly, pokud mají více než 95% linek společných a každý z nich má více než 10 linek. Skoro-duplicitní uzlu nahrazujeme uzlem s linkami, které jsou spojením linek všech skoro duplicitních uzlů.

3. Výpočet vah hran založený na spojení mezi webovými sídly

Hraně mezi uzly na stejném webovém místě je přiřazena hodnota 0. Jestliže míří m1 hran z množiny uzlů na jedné webové stránce na jeden uzel jiné webové stránky, pak každé hraně je přiřazena váha autority 1/m1. Jestliže

míří m2 hran z jednoho uzlu na jedno webové místo na množinu uzlů jiného webového místa, každé hraně je přiřazena hub váha 1/m2. Toto zabraňuje vlivu jednotlivých webových míst ve výpočtu. Obrázek ilustruje přiřazení váhy autority hraně a váhy hubu hraně k vícenásobným hranám z jednoho hostu na druhý.

4. Výpočet skóre hubu a autority pro každý uzel grafu

Deset nejvíce ohodnocených autorit je vráceno jako stránky, které jsou nejvíce relevantní výchozí straně U.

Bharat a Henzinger [9] navrhli rozšířený algoritmus pro vyhledávací stroje, který je založen na webové technologii a který se nazývá Topic distilation. Topic distilation je definován jako proces nalezení kvality webových stránek (více relevantních) odpovídajících předmětu dotazu. Algoritmus topic distilation řeší tři problémy spojené s algoritmem HITS. První problém je vzájemně vynucený vztah mezi webovými sídly (falešná autorita delegující množinou uzlů na jiný uzel). Ostatní dva problémy jsou automaticky generované linky (linky bez lidského posouzení) a přítomnost ne-relevantních uzlů. Jestliže je tam m1 hran směřujících z množiny uzlů na jednom webovém sídle na jeden uzel na jiném webovém sídle, potom každé hraně je přiřazena váha autority (edge_auth_wt) hodnoty 1/m1. Podobně jestliže tam je m2 hran směřujících z jednoho uzlu na jednom webovém sídle na množinu uzlů jiného webového sídla, potom každé hraně je přiřazena hodnota váhy hubu (edge_hub_wf) hodnoty 1/m2. Navíc izolované uzly jsou eliminovány z grafu. Váha autority a hubu každého uzlu je spočítána iterativně jako:

$$\forall u \in V$$

$$a(u) = \sum_{(u,v)\in E} h(v) \times edge\_auth\_wt(u,v)$$

A

$$h(u) = \sum_{(u,v)\in E} a(v) \times edge\_hub\_wt(u,v)$$

Tato modifikace algoritmu HITS eliminuje problém vzájemně vynucených vztahů. Podobnost mezi dotazem a uzlem vracená vyhledávacím strojem je definována jako váha relevance uzlu. Váha relevance každého uzlu se spočítá a uzly, jejichž váha relevance klesne pod prahovou hodnotu, jsou eliminovány. Tato eliminace řeší zbývající dva problémy.

3.5 Katalogy a vyhledávací stroje

3.5.1 Katalog

Nebo také předmětový katalog, se snaží uspořádat informační zdroje v Internetu. Oproti robotům kteří pracují mechanicky a nemají možnost pochopit význam informace, jsou katalogy uspořádávány lidským faktorem, který tyto informace rozděluje do příslušných skupin, podle jejich Významu.

Nevýhodou tohoto přístupu je pomalé rozrůstání počtu odkazů. Ačkoliv je možnost pomocí formuláře nechat zařadit stránky do určité kategorie samotným tvůrcem, stále je toto zařazování odkazů do kategorií ve srovnání s roboty pomalé. V Předmětovém katalogu je možné použít takzvané obecné vyhledávání. Tento postup je vhodné použít, když uživatel přesně neví co hledá, ale má povědomí o kategoriích do kterých by hledanou informaci zařadil. Díky kategoriální struktuře se uživatel postupně dostává

z nejobecnějších kategorií až na kategorie konkrétní. Kde, si sám podle vlastních rozhodnutí volí jakou kategorii chce navštívit.

3.5.2 Vyhledávací stroj

Vyhledává uživatelem zadaný výraz v databázi indexů, do které předtím nashromáždil informace vyhledávací robot. Pokud se ovšem nejedná o vyhledávací stroj, který prohledává www prostor na požádání. Ten vrací výsledky při běhu vyhledávání. Tento způsob ovšem v praxi není téměř používaný. Nicméně praktická část této práce obsahuje právě vyhledávač, který vrací výsledky přímo při procházení webových stránek. A to vzhledem k náročnosti na velikost databáze z hlediska počtu indexů na stránkách se vyskytujících. A zároveň kvůli rychlosti vyhledávání. Indexování není nejrychlejší záležitostí. Takovýto postup se vyplácí pro výkonné servery. Avšak s přibývajícím počtem dokumentů se více hodí Indexování.

3.6 Slovník

Slovník má podobu sekvenčního seznamu slov oddělených nulami a hashovací tabulkou a s jejími ukazateli. U prototypu vyhledávače stačilo na slovník jenž obsahoval 14 miliónů slov pouhých 256 MB operační paměti, díky čemuž nebyl problém s rychlou použitelností. U slovníku je obtížná paralelizace indexování. Za normálního provozu by měly být indexery schopny přidávat nová slova do slovníku. Řešení problému je zafixováním nálezů nových slov, tím že se zapíší do zvláštního souboru a tento soubor je na konci zpracován posledním indexerem. Oproti slovníku není tento soubor nijak objemný, díky tomu jeho zpracování nezabere tolik času. Hity představují většinu z obsahu indexu, proto je nezbytné, aby byly ukládány co možná nejefektivněji. Google má pro toto uložení vyhrazeny dva bajty. Hity

rozdělujeme na dvě kategorie a to obyčejné a důležité. Důležitá jsou slova obsažená v URL, titulku, textu a metatazích a odkazech. V indexech je před každým hitem uložena jeho délka. Zkombinováním wordID(respektive docID ve zpětném indexu) s délkou ušetříme místo. Na délku tím pádem zbývá 8 respektive 5 bitů. V případě že je hitů více bude délka reprezentována escape kódem, načež samotná délka se uloží do následujících dvou bajtů. Index se rozděluje do skupiny kontejnerů a každý kontejner pokrývá určitou část wordID. To znamená že index je tímto postupem částečně setříděný. Obsahuje-li dokument slova, u kterých spadají jejich wordID do daného kontejneru, přidá se do kontejneru jeho docID společně s příslušnými wordID a jejich hitlistem. Tento úkon navzdory potřebě prostoru pro duplikovaná docID významně usnadňuje práci třídiče. Zpětný index osahuje stejné kontejnery jako normální index. Změna je pouze v přetřídění kontejneru podle wordID. Každé platné wordID má ve slovníku doplněn odkaz na kontejner jenž obsahuje seznam dokumentů, které dané slovo obsahují. Kvůli snadnému slučování seznamů při práci s víceslovnými dotazy je vhodné seznam řadit dle docID. Pokud ovšem budeme řadit dokumenty podle PageRanku stanou se jednoslovné odpovědi trivialitou a je možné, že dokonce i odpovědi na víceslovné dotazy budou téměř na začátku. Google používá dvě sady kontejnerů. První je krátká, která obsahuje pouze hity v titulcích a odkazech. Druhá je dlouhá a ta je úplná. Při prohledávání se prochází nejprve první sada a pokud není dostatečné množství nalezených výsledků, prohledává se i druhá sada.

3.7 Indexy

Index přiřazuje klíčová slova k množinám www dokumentů, kde se vyskytují. Pokud se klíčové slovo alespoň jednou v dokumentu vyskytuje,

stává se z něho index. Ze kterých částí dokumentů jsou indexy vytvářeny, záleží na vyhledávacím stroji. Některé stroje procházejí plný text, jiné se zaměřují jen na významnější části jako jsou například název, nadpisy, klíčová slova v meta tazích. Pro průběžnou údržbu indexu si vyhledávací stroj zaznamenává i čas poslední aktualizace. Během indexování se často provádějí některé transformace slov jejichž cílem je vyšší úspěšnost vyhledávání. Dochází ke sjednocování písmen, velká písmena jsou převedena na malá. Navíc jsou vynechávána i stopslova. Což jsou slova, která z hlediska vyhledávání nemají žádný přínos. Za stopslova jsou například považovány předložky, spojky a jiné.

3.8 Index dokumentů

Index dokumentů obsahuje informace o každém jednotlivém dokumentu. Tyto informace zahrnují například aktuální stav dokumentu, kontrolní součet souboru a různé statistiky a ukazatel do schránky. V případě rozparsování dokumentu je v položce také odkaz do souboru ve kterém je URL dokumentu a jeho titulek. Pokud však k rozparsování nedošlo, odkazuje do URLlistu, kde je pouze URL. Výhoda tohoto řešení spočívá v kompaktnosti datové struktury a zároveň schopnosti načíst záznam při jediném přístupu na disk, což znamená urychlení.

3.9 Rozpoznávání textu

Rozpoznávání sémantického obsahu textu je dodnes pro roboty nemyslitelné. Stále ještě není znám postup podle kterého by robot bez

problému rozluštil význam textu a dokázal jej zařadit podle jeho významu, natož si jeho sdělení spojit s jinými texty.

Věcný obsah textu nemusí vždy záležet na slově samotném. Důležitými faktory mohou být počet výskytů slova v daném textu, frekvence výskytu slova v daném textu vztaženou k délce textu, poměr frekvence výskytu slova v daném textu a v ostatních textech, konkrétní poloha výskytu slova v daném textu (např. nadpis celého textu).

3.10 Principy zpracování rozpoznávaných informací

3.10.1 Boolovský model

Různé vyhledávací jazyky dodnes používají pro vyhledávání boolovského modelu. Nejčastěji používanými boolovskými spojkami jsou AND, OR, NOT. Použitím spojky AND, logického součinu, použité mezi dvěmi a více výrazy, říkáme, že tyto výrazy mají být obsaženy najednou v hledaném dokumentu. Oproti tomu spojka OR použitá mezi dvěmi a více výrazy označuje logický součet, kde jako výsledek dotazu postačí výskyt alespoň jednoho z výrazů v prohledávaném dokumentu. Poslední uvedená spojka NOT je ve vyhledávacích jazycích ve většině případů použita jako funkce AND NOT, což při použití mezi dvěma výrazy znamená, že hledáme dokumenty, kde se vyskytuje první z hledaných výrazů a zároveň se v tomto dokumentu nevyskytuje druhý výraz. Použitím těchto logických spojek má uživatel vyhledávacího mechanismu možnost upravovat rozsah jeho dotazu. Použitím spojky AND je možné dotaz upřesnit, ale také omezit na menší počet vrácených výsledků. Naopak užitím spojky OR bychom mohli dotaz rozšířit o jiné tvary hledaného výrazu, například synonyma, nebo jiná slova s naším hledaným výrazem související. Ovšem tyto úpravy spektra působnosti našeho

dotazu sebou nesou i jisté nevýhody. Například u použití spojky OR je možné předpokládat příliš mnoho odpovědí, což může vést k informačnímu přehlcení. Na druhou stranu použitím spojky AND můžeme některé výsledky odfiltrovat a tím nenalezneme všechny relevantní odpovědi. Tento problém znázorňuje následující tabulka [1]:

	Vybráno	Nevybráno	
Relevantní	A	B	A+B
Irelevantní	C	D	C+D
	A+C	B+D	

„Tab. 2-5: Počty dokumentů pro hodnocení úspěšnosti vyhledávání“

„Zde A značí počet vybraných dokumentů relevantních k informačnímu požadavku, B je počet dokumentů sice relevantních k informačnímu požadavku, ale při vyhledávání nevybraných atd.“

V tomto zdroji jsou dále popisovány problémy, které způsobují nevyhledání relevantního dokumentu, nebo naopak vyhledání irelevantního dokumentu. Zdroj těchto problémů pramení v [1]:

- „nedokonalém vyjádření obsahu klíčovými slovy, nedokonalost přitom může pramenit jak z nedokonalých vyjadřovacích schopností klíčových slov, tak i z nedokonale provedeného indexování,“
- „rozdílném vyjádření jednoho tématu indexátorem a uživatelem,“

- „principiálních možnostech vyjádřit pomocí spojek AND, OR a NOT informační požadavek.“

3.10.2 Aktuálnost a pravdivost dat

Pokud se již podaří nalézt přiměřené množství dat, mohou zde nastat i jiné problémy. Tyto například souvisí s aktuálností informací a také pravdivostí. Otázky spojené s aktuálností informací však souvisí s povahou informací a jejich využití. Například bude-li dotaz směřován na historii nějaké firmy, nemusí být informace tak aktuální jako kdyby nás zajímala včerejší finanční transakce dané firmy a my bychom měli nejnovější informace o firmě týden staré. Dalším kritériem je též pravdivost informací, nebo také serióznost zdroje. Žádná data nelze považovat za 100% pravdivá. I když u některých zdrojů lze předpokládat větší míru serióznosti než u jiných.

3.11 Návrh tvorby www stránek

Pro usnadnění práce vyhledávačů, při procházení stránek a separování textu, zde popíši několik známých pravidel pro tvorbu internetových stránek. Porušení některého z těchto pravidel má u některých vyhledávačů za následek špatně přiřazená klíčová slova podle významu. Nebo úplné přeskočení stránek a tím pádem i neindexování jejich obsahu.

3.11.1 Splnění základních podmínek SEO

První z podmínek SEO je používání rámců. Toto používání by mohlo mít za následek nesprávné načtení stránky vyhledávačem, který bude mít

neúplný dokument a tím pádem nedokáže indexovat všechna klíčová slova. Nebo v jiném případě nenalezne odkazy na další stránky. A hlavním důvodem je, že při použití rámců se nemění adresa stránek. Z toho vyplývá, že vyhledávač zvládá pouze načtení hlavní stránky. Na místo rámců je výhodnější používat tabulkové struktury webových stránek. Dalším z kritérií je počet odkazů, které vedou na vaší stránku a počet odkazů, které vedou na jiné stránky, z toho se vypočítává takzvaný page rank. Hodnocení stránek podle tohoto kritéria zároveň i nepatrně předchází vzniku nových neviditelných www stránek. Pokud by na vaše stránky vedl jen jeden odkaz, a to ze stránek, které co nevidět zaniknou, mohly by se vaše webové stránky stát neviditelnými, pokud nejsou ve vyhledávači přímo zaregistrovány.

3.11.2 **Algoritmus PageRank**

Většina vyhledávacích strojů vychází pouze z vyhledávání informací na webu a indexování nalezených klíčových slov a řetězců. Google však jde ještě dál používá grafovou strukturu webu a tím získává lepší výsledky. Využívá k tomu algoritmu, který se jmenuje PageRank. Tento algoritmus bere v úvahu obsah webové stránky a zároveň jeho umístění v grafu webu [2]:

Označení:

u = uzel (vrchol) webového grafu

d+i = odchozí stupeň uzlu i

w1,w2, …, wk = uzly ukazující na uzel u

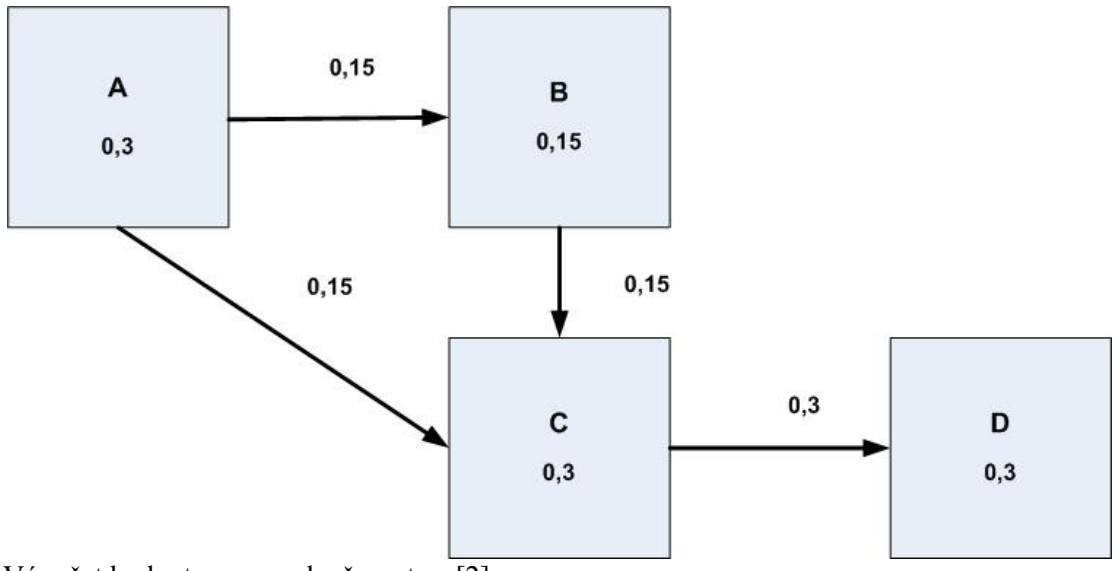
η = normalizační konstanta ($\eta < 1$)

PageRank daného uzlu u je pak dán vztahem

$$PR(u) = (1-\eta) + \eta \cdot \left(\frac{PR(w_1)}{d_1^+} + \frac{PR(w_2)}{d_2^+} + \ldots + \frac{PR(w_k)}{d_k^+} \right)$$

Tento algoritmus přiřadí všem stránkám ohodnocení 1 a po té rekurzivní metodou spočítá PageRank pro všechny stránky. Hodnotu každé stránky rozdělí mezi odchozí hrany rovnoměrným dílem. Pokud stránky které ukazují na naši stránku mají vysoký PageRank, pak má vysoký PageRank i naše stránka. PR(u), jenž je vidět výše na obrázku, znázorňuje pravděpodobnost, že nějaký náhodný surfař narazí na tuto stránku. Normalizační konstanta η představuje pravděpodobnost, že nějaký surfař si vycházející z naší stránky vybere náhodně nějakou jinou stránku.

3.11.3 Výpočet hodnoty PageRank



Obr. 3: Výpočet hodnoty pagerank převzate z [2].

Na obrázku je vidět ukázka výpočtu PageRank. Hodnota stránky A je rovnoměrně rozdělena mezi stránky B a C. Jelikož stránka B má jen jeden odkaz na stránku C stránka C získává celkovou hodnotu B, čímž má ve finále hodnotu 0,6. Stránka D tuto hodnotu opět převzala celou jelikož je jediná, na kterou stránka C ukazuje, čímž má také hodnotu PageRanku 0,6.

3.11.4 Reputace stránky

Výpočet reputace stránek se počítá podle počtu návštěv náhodného uživatele na těchto stránkách. Vychází se z toho, že tento náhodný uživatel hledá informace o určitém zaměření problému a v každém kroku jde na náhodný link, který obsahuje informace o dané problematice, nebo následuje náhodný link z výchozí stránky. Pro tento výpočet se například využívá jednoúrovňový reputační ohodnocující algoritmus [2]:

Tento algoritmus konverguje při vytváření reputačního hodnocení pro každou stranu w a termín τ. [6] Tento algoritmus produkuje pravděpodobnostní formulace hledání témat, která daná webová stránka je autorita.

Reputace stránky w k tématu τ je definována jako pravděpodobnost, že náhodný surfař hledající téma τ a navštíví stránku w. Notace algoritmu navrženém Rafiei a Mendelsonem [6] jsou:

$N\tau$ = celkový počet stran na webu obsahující termín τ

d+x = počet vycházejících linek ze stránky x

p = pravděpodobnost že náhodný surfař vybere stránku rovnoměrně náhodně z množiny stránek obsahujících termín τ

(1 – p) = pravděpodobnost že náhodný surfař následuje vycházející link ze stávající stránky

R = matice, kde řádky korespondují webovým stránkám a sloupce odpovídají každému termínu, který se objeví na webových stránkách. Každý prvek matice R(w, τ) je reputační hodnota webové stránky w odpovídající termínu τ.

Pravděpodobnost, že náhodný surfař navštíví stránku w v každém kroku náhodného kroku je p/N τ jestliže stránka w obsahuje termín τ a je nula v opačném případě. Jestliže stránka x je předchůdce stránky w, potom pravděpodobnost, že surfař navštíví stránku w v k krocích po návštěvě stránky x je

$$\left(\frac{(1-p]}{d_x^+}\right) R^{(k-1)}(x,\tau)$$

Kde $R^{(k-1)}(x,\tau)$ je pravděpodobnost, že surfař navštíví stránku x v kroku (k-1). Algoritmus vypočítává pravděpodobnost iterativně:

Pro každou stranu w a termín τ

Jestliže se τ objeví na stránce w

R(w, τ) = 1/Nr

Jinak R(w, τ) = 0

Jestliže není R konvergováno

Nastav R'(w, τ) = 0 pro každou stránku w a termín τ

Pro každou linku x -> w

$$R'(w,\tau) = R'(w,\tau) + R(x,\tau)/d_x^+$$

Pro každou stránku w a termín τ

$$R(w,\tau) = (1-p)\cdot R'(x,\tau)$$

Jestliže se termín τ

$$R(w,\tau) = R(w,\tau) + p/N_\tau$$

Jiný algoritmus uvádějí Zhang a Dong [7], který je založený na Markovových řetězcích.

3.11.5 Jednoznačné popisy obsahu

Je nezbytné volit jednoznačné popisy obsahu stránek, ať už v meta tagu klíčových slov, či v jiných důležitých elementech www stránky. Jako například nadpisy, titulek stránky, alternativní text obrázku, atd. Vhodně volená slova správného významu, zabrání při vyhledávání dotazů navrácení výsledků s jinou tématikou než měl uživatel při zadávání dotazu na mysli. Jako příklad bych mohl uvést www stránku zabývající se pohlednicemi. Nějaký sběratel pohlednic si napíše www stránky a zvolí popis těchto stránek jako pohledy. A i v klíčových slovech a nadpisech používá slovo pohledy, namísto slova pohlednice. Kdyby správně zvolil slovo pohlednice, nemohlo by se stát, že při zadání slova pohledy do vyhledávače mu kromě jeho stránky vyskočí mnoho stránek s romány o láskyplných pohledech a podobně.

3.11.6 Klíčová slova v nadpisech a popiscích stránky

Neméně významnou částí je zvolení správného popisu, ať už samotného titulku dokumentu, nebo alternativního textu obrázku. Vhodně volená slova mohou usnadnit správné zařazení dokumentu při vracení výsledků ve vyhledávači. Který většinou zjišťuje relevanci odpovědi podle počtu výskytů klíčového slova v textu a délce tohoto textu. V některých vyhledávačích jsou prohledávány jen titulky stránky a klíčová slova, a samotný text již není procházen. Tento postup ukazuje na potřebu správného vyjádření obsahu stránek v několika slovech. Pro správné zařazení co se obsahové části týká. Nemalou roli hrají také slova v tazích označujících nadpisy. Ty jsou také považovány za určité vyjádření obsahu.

4 Způsoby, popis robotů – Google, Seznam

4.1 Robot

Robot je program, který prochází webové servery a webové dokumenty a analyzuje jejich obsah. Pro své vyhledávání mohou roboty používat několik přístupů k vyhledávání. Jedním z přístupů je vyhledávání do hloubky, u kterého se postupuje následujícím způsobem. Nejprve se prohledá startovací dokument a nalezený odkaz na další dokument je vždy zařazen dopředu v seznamu dosud neprohledaných dokumentů. Dalším přístupem je prohledávání do šířky, kde se postupuje podobně jako při prohledávání do hloubky, ovšem rozdíl je v zařazení nových odkazů na konec seznamu. Tímto postupem se nejprve prohledají všechny dokumenty o stejné šířce a poté teprve ostatní. Posledním z nejpoužívanějších postupů vyhledávání je náhodné vyhledávání, kde robot začíná s určitým počtem startovacích dokumentů a náhodně si jeden z nich vybere a ten prohledá. Nalezené odkazy uloží do seznamu, pokud tam ještě nejsou, a pokračuje znovu s náhodným výběrem.

4.1.1 Zatížení serverů

Neblahým faktorem vyhledávacích strojů je také zatěžování webových serverů, ze kterých stahují informace. Toto zatížení se zejména projeví na serverech většího rozsahu, ze kterých se stahuje větší množství informací. Je celkem zřejmé že zatížení serverů je nevítanou událostí. Obzvláště v době kdy tento server potřebuje pracovat na úkonech k nimž byl určen. Proto je nezbytné tomuto střetu zájmů zabránit. Aby kvůli tomuto problému nedošlo k cílenému zákazu indexovacích robotů, nabízí se zde jiné řešení, které se v praxi používá. A tj. časové plánování stahování informací z webových serverů.

4.1.2 Časové plánování

Je nezbytným předpokladem pro řešení problémů jež jsem popsal v minulém bodě. Tímto časovým rozvrhem zabráníme výše popsanému problému zatěžování serverů. Časové plánování vychází z předpokladů, že servery v nočních hodinách bývají minimálně využívány, tím pádem většina vyhledávacích robotů začíná svou práci právě v tuto dobu. Přední světové vyhledávače ovšem nevyužívají výpočetní sílu jen jednoho stroje, nýbrž mnoha strojů, které pracují nezávisle na ostatních a vyhledávají ve svém vlastním poli působnosti. Tyto stroje by bez jakékoliv vzájemné komunikace mohly indexovat najednou stejné webové dokumenty. Proto je nezbytné, aby podléhaly vyššímu vedení, které dohlíží na jejich působnost, čímž je jejich společná centrální jednotka. Pokud by totiž celý webový prostor procházel jen jeden indexovací stroj, tak by jeho práce trvala neskutečně dlouho a tím pádem by nepracoval jen v nočních hodinách, ale nepřetržitě. Čímž by zatížení serverů bylo značné, vezmeme-li v potaz, že počet vyhledávačů jde do tisíců. Pokud by každý z těchto robotů měl zájem o jeden známý, se značným množstvím informací, často se měnící, a proto často indexovaný server. Tento server by nebyl schopen normálně fungovat. A pokud by se tedy předpokládalo, že chceme zabránit zahlcení některých serverů jsme opět u časového plánování, což by pro jednoho indexovacího robota v rámci jedné vyhledávací služby byl značný problém. V omezeném čase působnosti by tento robot naindexoval mnohem méně než v současnosti dokáží indexovací roboti předních vyhledávačů, a tím by se také prodloužila doba opakovaného indexování. Tímto by aktuální informace byly aktualizovány po mnohem delší době.

4.2 Pole působnosti robotů

Robot má jistá omezení, která jsou určena textovým souborem s názvem robots.txt a nebo metatagy name=“robots“. Na těchto místech lze

nastavit jací, nebo zda všichni, roboti nesmí indexovat tyto stránky. Dalším omezením mohou být stránky s autorizovaným přístupem. V neposlední řadě jsou překážkou pro vyhledávací roboty stránky, na které nevede žádný odkaz.

4.3 Seznam

Jedná se o katalogový vyhledávač s možností fulltextového vyhledávání. Pro fulltextové vyhledávání v českém jazyce využívá svých vlastních fulltextových technologií. Avšak pro vyhledávání ve všech jazycích využívá fulltextové technologii google. Nevrátí-li fulltextové vyhledávání seznamu žádný výsledek, automaticky se spouští fulltextové vyhledávání googlu. Vyhledávání ve firemním katalogu je též součástí vyhledávání na seznamu. Pokud je hledané slovo obsaženo v názvu některé z kategorií, seznam vrátí odkaz na tuto kategorii.

4.4 Google

Google je nejpoužívanějším vyhledávačem na světě. Nabízí asi 117 jazykových rozhraní. Má nejobsáhlejší prostor vyhledávání informací. Je to fulltextový vyhledávač. Tj. nevyhledává jen v popiscích www stránek, ale v celém obsahu. Nevyhledává ovšem odpověď na uživatelem zadaný dotaz v okamžiku jeho napsání. Google prochází web průběžně a indexuje si procházené stránky do své databáze. To znamená, že na stránkách vyhledá klíčová slova a ta si uloží do databáze s odkazem na stránky, kde tato klíčová slova nalezl. Google se pyšní největším počtem indexů. Google však není jen fulltextovým vyhledávačem nabízí i katalogové zpracování webu. Robot používaný googlem se jmenuje googlebot.

5 Praktická část – Programování

Tento vyhledávací robot je, oproti běžně používaným vyhledávacím robotům, značně zjednodušen, neprovádí například indexování. Tento robot je konstruován tak, aby předvedl základní dovednosti vyhledávacího robota, aniž by potřeboval databázi obřích rozměrů. Zabývá se především procházením webových stránek, ze kterých vyseparuje odkazy na další stránky a vyhledá na těchto stránkách podle zadání buď emaily, a nebo zadaný výraz. Tento robot běží na vláknech a stahuje až 10 stránek najednou. Přístup k výběru nepřečtených stránek je vybírán prvkem náhody. Nepřečtené stránky jsou získány v podobě odkazů z databáze, kde jsou označeny jako nepřečteny a posléze jsou načteny a zpracovány pomocí odkazů, jenž na ně ukazují. Doba stahování se dá naplánovat pomocí plánovače.

5.1 Načítání stránky

Nedílnou součástí vyhledávacího robota je načítání obsahu stránky. To je v tomto případě prováděno třídou PageDownload. Pomocí instanční proměnné url, získáme jako parametr konstruktoru adresu načítané stránky. Pokud url není prázdnou adresou vyšleme požadavek a načte se obsah stránky v kódování utf8. Pokud se ovšem zjistí použitím metody ParseEncoding jiné kódování, uloží se obsah stránky v tomto kódování. Obsah stránky se načetl do instanční proměnné s názvem text. Jako ukázka kódu je zde metoda, která ve třídě PageDownload zajišťuje stažení stránky, metoda Download.

```
private void Download()
        {
            if (link == null) return;

            try
            {
                Encoding encoding = Encoding.UTF8;
```

```
                request = WebRequest.Create(link.Url);
                request.Timeout = TIMEOUT;

                HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
                Stream stream = response.GetResponseStream();
                StreamReader reader = new StreamReader(stream,
encoding);
                string text = reader.ReadToEnd();
                response.Close();
                reader.Close();

                encoding = ParseEncoding(text);
                if ((encoding == null) ||
(encoding.Equals(Encoding.UTF8)))
                {
                    pageText = text;
                    return;
                }

                request = WebRequest.Create(link.Url);
                request.Timeout = TIMEOUT;

                response =
(HttpWebResponse)request.GetResponse();
                stream = response.GetResponseStream();
                reader = new StreamReader(stream, encoding);
                text = reader.ReadToEnd();
                response.Close();
                reader.Close();

                pageText = text;
            }
            catch
            {

                pageText= string.Empty;
            }

            request = null;
            thread = null;
            Complete = true;
              }
```

5.2 Průchod obsahu stránky

5.2.1 Separování odkazů

Další část se zabývá zpracováním staženého obsahu stránky. Tento úkon má na starost třída LinkSeparator, která jako parametry konstruktoru obsahuje text načtené stránky, odkaz z jaké stránky byl text načten a objekt, který pracuje s databází. Nejprve se vyseparují odkazy pomocí metod Parse a PreparateLinks. Pokud tyto odkazy neobsahují mailto a nebo koncovky jako jpg, dtp , txt, zip, JPG, rar, png a podobné. A nejsou-li již obsaženy v databázi, uloží se odkaz do databáze. Nyní ukázka zdrojového kódu metody PreparateLinks.

```
public void PreparateLinks(DownloadSubject subject)
        {
            string[] tiles = pageText.Split('<');
            for (int i = 0; i < tiles.Length; i++)
            {
                if ((tiles[i].StartsWith("a ")) ||
(tiles[i].StartsWith("A ")))
                {
                    string link = Parse(tiles[i]);
                    if (link != null)
                    {
                        link = FullLink(link);

                        if (subject == DownloadSubject.Email)
                        {
                            if
(!dbsEngine.ContainsEmailLink(link))
                            {
                                if (link.Contains(".jpg") ||
link.Contains(".dtp") || link.Contains(".txt") ||
link.Contains("mailto") || link.Contains(".zip") ||
link.Contains(".JPG") || link.Contains(".rar") ||
link.Contains(".jpeg") || link.Contains(".png") ||
link.Contains(".avi") || link.Contains(".ppt"))
                                { }
                                else
                                {

dbsEngine.SaveEmailLink(link, false);
                                }
                            }
                        }
                        else
                        {
```

```
                                if
(!dbsEngine.ContainsTermLink(link))
                                {
                                    if (link.Contains(".jpg") ||
link.Contains(".dtp") || link.Contains(".txt") ||
link.Contains("mailto") || link.Contains(".zip") ||
link.Contains(".JPG") || link.Contains(".rar") ||
link.Contains(".jpeg") || link.Contains(".png") ||
link.Contains(".avi") || link.Contains(".ppt"))
                                    { }
                                    else
                                    {

dbsEngine.SaveTermLink(link, false);
                                    }
                                }
                            }
                        }
                    }
                }

        }
```

5.2.2 Čistá doména

Neméně potřebné je zjištění takzvané čisté domény. To je url adresa bez koncovky. Například u adresy http://www.seznam.cz/robots.txt je adresa http://www.seznam.cz čistou doménou. K tomuto zjištění slouží metoda Url, která odřízne konec url adresy, pokud obsahuje nějaké známé koncovky jako htm, php, xhtml, phtml, asp, a další. A pokud url adresa jednu z těchto koncovek neobsahuje, odstraní se pouze koncové lomítko z této url adresy, pokud je obsaženo, a url je považována za čistou.

```
  public void Url()
        {
            string url = link.Url;

            string end = url.Remove(0, url.LastIndexOf('/') +
1);
            if (end.Contains(".htm") || end.Contains(".php") ||
end.Contains(".xhtml") || end.Contains(".phtml") ||
end.Contains(".asp") || end.Contains(".xml") ||
end.Contains(".html") || end.Contains(".HTM") ||
end.Contains(".HTML") || end.Contains(".jpg") ||
end.Contains("php") || end.Contains("fcgi"))
            {
```

```
                this.url = url.Remove(url.LastIndexOf('/'));
            }
            else
            {
                if (url.EndsWith("/")) url =
url.Remove(url.LastIndexOf('/'));
                this.url = url;
            }

          }
```

5.2.3 Neúplné odkazy

S proměnou čistá doména pracuje metoda FullLink, která zajišťuje zpracování odkazů, které mají podobu neúplného odkazu. Například začínající na „./“, „../“ a nebo „/.“ Pokud odkaz začíná na „./“, přidá se tento odkaz k čisté doméně kromě tečky na začátku. Začíná-li odkaz na „../“, znamená to, že z čisté domény ještě odebereme text od konce k poslednímu lomítku, a k takto upravené čisté doméně teprve přidáme odkaz jenž začínal „../“, ovšem opět bez teček. Zde je ukázka metody FullLink.

```
private string FullLink(string link)
        {
            if (link.StartsWith("http"))
            {
                return link;
            }
            if (link.StartsWith("./"))
            {
                return url + link.Remove(0, 1);
            }
            if (link.StartsWith("../"))
            {
                string url2 = url.Remove(url.LastIndexOf("/"));
                return url2 + link.Remove(0, 2);
            }
            if (link.StartsWith("/"))
            {
                string url2 = url;
                if (url.Contains("http"))
                {
                    url2 = url.Remove(0, url.IndexOf("/") + 2);
                }
                if (url2.Contains("/"))
```

```
                {
                    url2 = url2.Remove(url2.IndexOf("/"));
                }

                url2 = "http://" + url2;

                return url2 + link;
            }

            return url + "/" + link;
        }
```

5.2.4 Separování emailů

Po vyseparování odkazů na další stránky se ještě provádí separování emailů z načteného textu. Vzhled emailové adresy má relativně volná pravidla. Proto je mnohdy obtížné odchytit všechny emaily. Kvůli tomu jsem použil dva postupy, jejichž spojení považuji za velmi účinné. Jedná se o regulární výraz a test na koncovku emailu, zda spadá do domény tld (top level domain). Regulární výraz pro hledání emailů, který jsem převzal z [11] vypadá následovně:

```
[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,4}
```

Tento výraz znamená:

[a-zA-Z0-9._-] – může obsahovat alfanumerické znaky, nebo tečku,podtržítko či pomlčku.

+ -znamená opakování vzoru nejméně jednou.

{2,4} – znamená pevný počet opakování v tomto rozmezí.

Pokud textový řetězec vyhovuje této šabloně, je vystaven ještě testu zda končí na jednu 262 známých tld. Pokud vyhovuje, je tento řetězec považován za email, a pokud již není uložen v databázi, uloží se.

5.3 Databáze

Pro ukládání výsledků vyhledávání, jakožto nalezených nových odkazů či emailů, nebo pro uložení tld domén, které jsou nutné pro ověřování koncovek u emailů, jsem si navrhl tabulky v databázi.

Obr. 4: Diagram tabulek z databáze Crawler.

5.3.1 Funkčnost databáze

Tato databáze s názvem crawler obsahuje 5 tabulek „emaily“, „nalezeno“, „odkazy_emaily“, „odkazy_nalezeno“ a „tld“. Tabulka emaily je tvořena ze sloupců „id“, „email“ a „odkaz“, kde id je číslo identifikující nalezený výsledek, email je řetězec znaků představující nalezený email a odkaz je id číslo z tabulky odkazy_emaily reprezentující odkaz, na kterém byl email nalezen. Tabulka odkazy_emaily se skládá ze sloupců „id“, „odkaz“ a „precteno“. Id opět představuje identifikační číslo výsledku, odkaz představuje řetězec znaků, který byl nalezen a precteno představuje hodnotu true a nebo false. Pokud byl již odkaz použit k prohledávání je nastaveno na hodnotu true, pokud ještě nebyl použit, pak je false. Stejně je tomu i u tabulky odkazy_nalezeno. Další tabulkou je nalezeno, tato tabulka obsahuje sloupce „id“, „dotaz“ a „odkaz“. Což je analogie tabulky emaily, akorát zde je místo

emailu dotaz reprezentován řetězcem znaků. Poslední tabulkou je tabulka tld tato tabulka je jedno účelová, je určena jen ke čtení a porovnávání obsahu s předkládanými výsledky hledání emailů. Její struktura zahrnuje sloupce „id“ a „tld“, kde id je opět identifikační číslo domény a tld je řetězec znaků, který reprezentuje doménu. Pro vkládání do tabulek email, nalezeno a odkaz jsou napsány procedury InsEmail, InsNalezeno, InsOdkaz_nalezeno a InsOdkaz_emaily. Každá z těchto procedur si zjistí poslední vložené id a vloží nový řádek do tabulky představovaný id o jedna vyšším než bylo poslední a parametry metody. Zde je ukázka jedné z procedur.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[InsOdkaz_nalezeno]
    @newId int = 0 OUTPUT ,
      @odkaz varchar(MAX),
      @precteno bit
AS
BEGIN

      SET NOCOUNT ON;

      SET IDENTITY_INSERT [dbo].[odkazy_nalezeno] OFF

      INSERT INTO [dbo].[odkazy_nalezeno](odkaz, precteno)
VALUES (@odkaz, @precteno)

      SET @newId = SCOPE_IDENTITY()
END
```

5.3.2 Práce s databází

Pro komunikaci s databází je zde třída DatabaseDownloader, která nejprve pomocí metody NastavPripojeni, nastaví připojení k databázi. Dále tato třída obsahuje metody pro načítání, ukládání, mazání, update a vkládání

nových hodnot do tabulek v databázi.Metody načítání používají následující příkaz: select * from „název tabulky“. Vkládání je řešeno přes procedury popsané v předchozím odstavci. Mazání je řešeno například takto „delete from odkazy_emaily where id=…“

Update je řešen obdobně, jen místo delete přes příkaz update. Nakonec ukládání je řešeno přes vkládání do databáze a zároveň se ukládá do kolekce v rámci programu. Tato třída dále obsahuje metody pro zjišťování, zda je již v databázi odkaz nebo email uložen, či nikoliv. K tomu slouží právě zmiňovaná kolekce, do které se na počátku načetly data z databáze. A nyní se tato kolekce prochází prvek po prvku a porovnává se s parametrem metody zda se neshodují. Pokud ano, vrátí metoda hodnotu true, v opačném případě je výsledek false a nic nebrání uložení do databáze. Zde je ukázka metody UpdateEmail.

```
private void UpdateEmail(int id, string email, int link)
        {
            SqlCommand comm = new SqlCommand("update emaily set
email='" + email + "', odkaz='" + link + "'where id='" + id +
"'", conn);
            SqlDataReader reader = null;
            try
            {
                conn.Open();
                reader = comm.ExecuteReader();

            }
            catch
            {
                //MessageBox.Show("Při práci s databází došlo k
chybě!");
                System.Console.WriteLine("Při práci s databází
došlo k chybě!");
            }
            finally
            {
                if (reader != null) reader.Close();
```

```
                if (conn != null) conn.Close();
            }
              }
```

5.4 Plánování stahování

Jak jsem již výše uvedl, plánování, kdy se bude stahovat má své opodstatnění. Zde má uživatel možnost nastavit několik různých časů, kdy se bude provádět vyhledávání. Pokud se budou časy vzájemně křížit, bude mu oznámeno, že existují časy navzájem souběžné a budou spojeny v jeden časový interval. O chod plánovače se stará třída Scheduler, zde ukázka:

```
namespace Downloader
{
    public class Scheduler
    {
        private List<Event> events;

        public Scheduler()
        {
            events = new List<Event>();
        }

        public int Count
        {
            get { return events.Count; }
        }

        public Event this[int index]
              {

      ...
```

Aby měl uživatel možnost uchovávat své nastavené časy a při příštím spuštění programu je třeba upravit, je zde metoda Save. Zde je její ukázka:

```
public void Save(string file)
        {
            string head = "<?xml version=\"1.0\"
encoding=\"windows-1250\"?>";
            head += "<scheduler></scheduler>";

            XmlDocument schedulerXml = new XmlDocument();
            schedulerXml.LoadXml(head);

            for (int i = 0; i < events.Count; i++)
            {

schedulerXml.DocumentElement.AppendChild(events[i].Save(schedul
erXml));
            }

            schedulerXml.Save(file);
        }
```

Jak je patrné nastavení plánovače se ukládá do xml souboru. Odkud se později i načítá, když editujeme nastavení plánovače. Zde je ukázka kódu kde je určeno kam se má ukládat tento soubor:

```
public partial class MainForm : Form
    {
        private const string CONFIG_FILE = "config.xml";
        private static string CONFIG_DIR =
Application.LocalUserAppDataPath;
        private const string SCHEDULER_FILE = "scheduler.xml";
        private static string SCHEDULER_DIR =
Application.LocalUserAppDataPath;

        private Engine engine;
        private EngineState state;

        private Configuration config;
        private Scheduler scheduler;

        private Color col;

        public MainForm()
              {
```

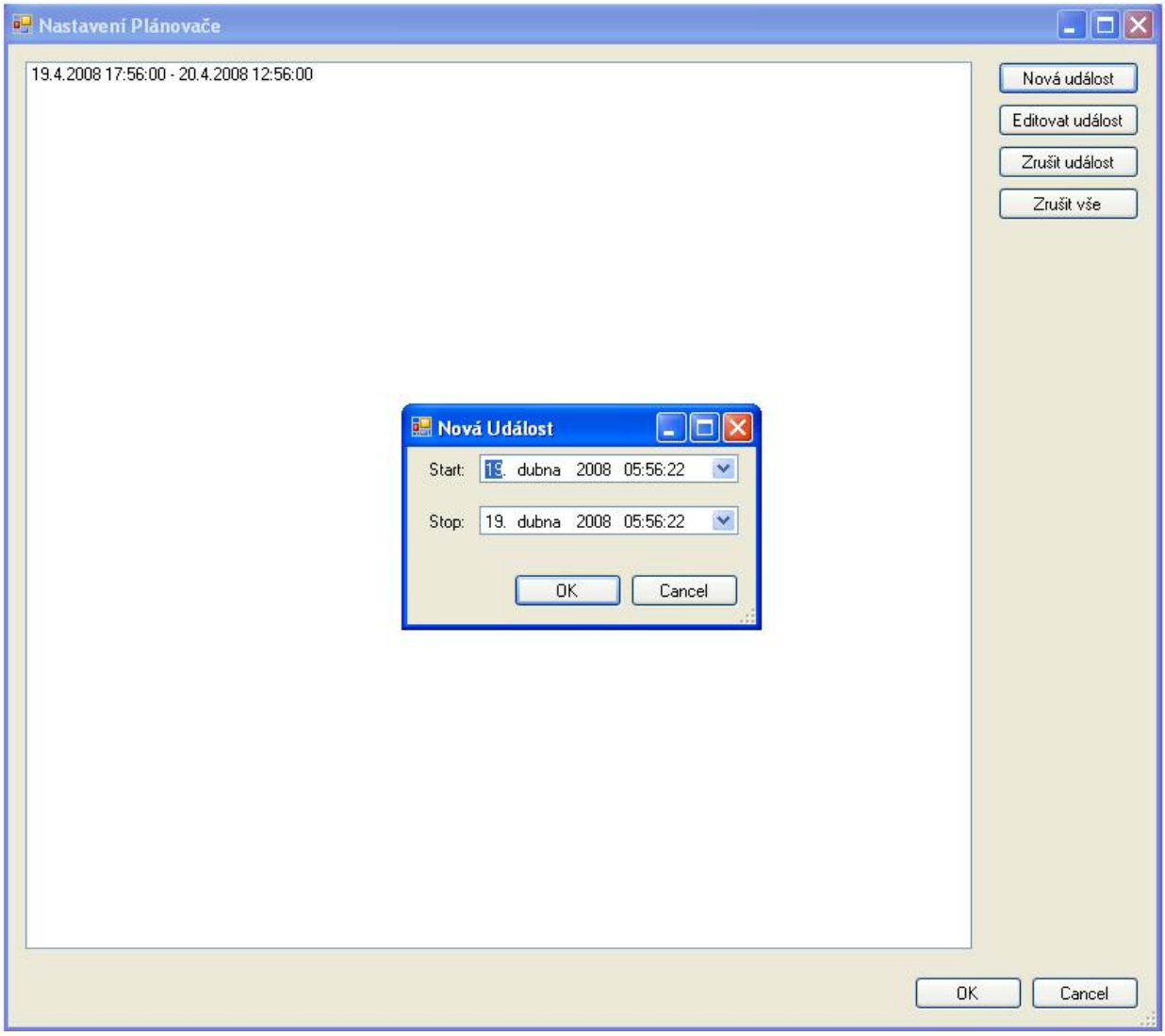
...

a

```
   private void MainForm_FormClosing(object sender,
FormClosingEventArgs e)
        {
            engine.Dispose();


            try
            {
                config.Save(CONFIG_DIR + "\\" + CONFIG_FILE);
            }
            catch { }
            try
            {
                scheduler.Save(SCHEDULER_DIR + "\\" +
SCHEDULER_FILE);
            }
            catch { }
              }
```

...



Obr. 5: Nastavení plánovače stahování a vytvoření nové události.

5.5 Omezení působnosti robota

Vyhledávací roboty během svého prohledávání webových stránek, mohou narazit na řadu překážek.

5.5.1 Neautorizovaný přístup

WWW stránky určené jen pro registrované uživatele, neumožní robotu, aby přečetl odkazy na těchto stránkách a tím mu zabrání ve zpracování dat.

Takto heslem zabezpečené stránky jsou pro robota, velkou překážkou,stránky označí jako nečitelné a pokračuje na jiné odkazy.

5.5.2 Zakázaný přístup

Pokud si tvůrce stránek nepřeje, aby jeho stránky procházely roboty, má kromě heslem chráněného přístupu ještě minimálně dvě možnosti. První z možností je nahrát do kořenového adresáře soubor robots.txt kde je možné zakázat, konkrétním, nebo všem robotům přístup, k určitým částem webu. Nebo druhá možnost je použití <meta name="robots" content="noindex", nofollow>

5.5.3 Špatně navržené stránky

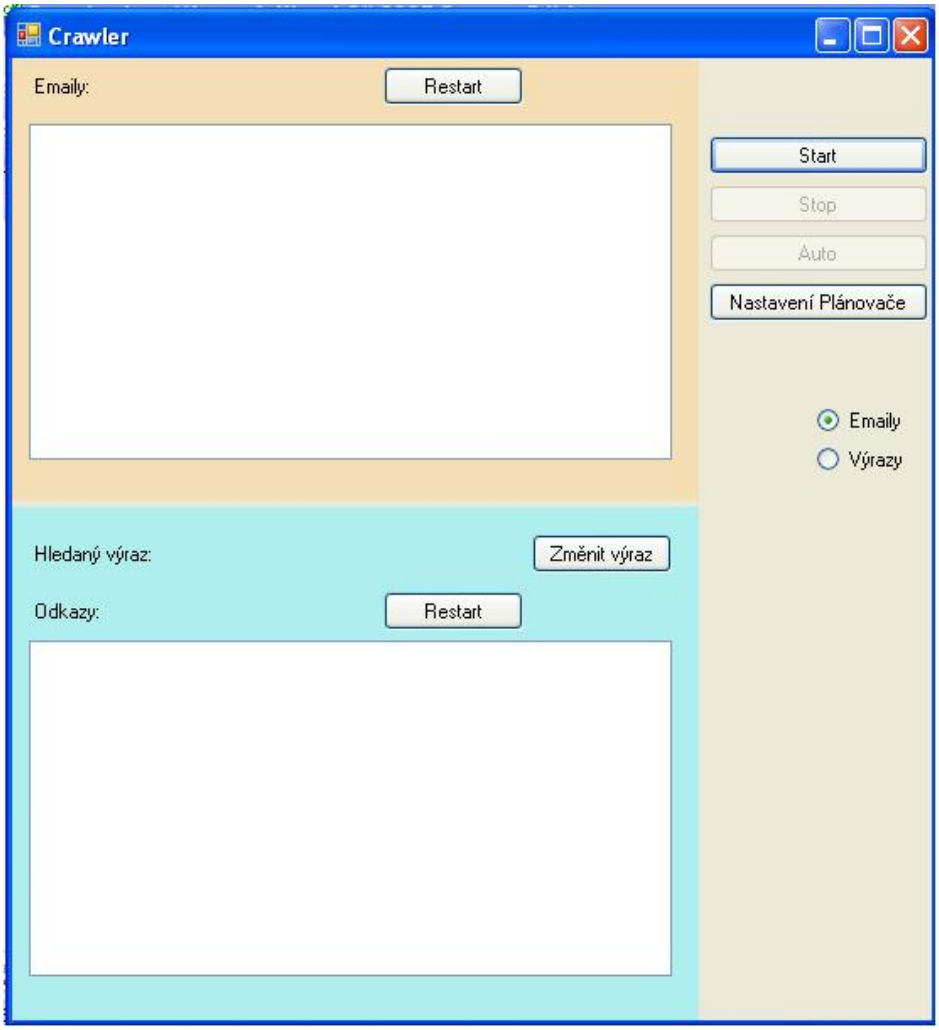
Jednou z hlavních chyb při tvorbě stránek je použití rámců, tím může dojít ke zmatení robota. Pokud načte jen část rámců, kde nebudou žádné odkazy, může se stát, že neprojde tyto stránky celé jak by bylo příhodné. Některé vyhledávače jako například google se takovýmito stránkami v některých případech ani nezabývají.

5.5.4 Neviditelné webové stránky

Jedná se o webové stránky, na které nevedou žádné odkazy. Podle statistik je těchto webových stránek více než těch veřejně indexovatelných. Toto je z hlediska touhy po informacích docela závažný problém. Přední vyhledávače sice nabízejí formuláře pro vyplnění údajů, pro tvůrce webových stránek, kvůli většímu zmapovanému prostoru. Avšak ne každý tvůrce stránek, tento formulář vyplní. Ať už se jedná o neochotu zveřejnění či neznalost.

5.6 Uživatelské ovládání

Hlavní uživatelský formulář je rozdělen na několik částí. V první části se nalézá vyhledávání emailů. Pokud je zatrženo políčko emaily a uživatel klikne na tlačítko start, začne vyhledávání emailů a jejich výpis do listboxu emaily. Toto vyhledávání má dva režimy. Prvním režimem je, že uživatel sám stiskne tlačítko start a tím odstartuje vyhledávání. Druhý režim je autorežim, kde má uživatel možnost v nastavení plánovače nastavit datum a čas, kdy bude stahování probíhat. Tento druh vyhledávání se vyvolá stisknutím tlačítka auto. Druhou částí je vyhledávání textového výrazu, který uživatel zadá po kliknutí na tlačítko změnit výraz. Zde napíše svůj výraz a stiskne tlačítko OK. A zadaný výraz se zobrazí. Pokud si uživatel přeje tento výraz vyhledávat, musí mít zaškrtnuté políčko výrazy. A po té si stačí již jen vybrat, zda chce vyhledávat ihned, a nebo plánovaně. Jak tomu bylo také u emailů. Jak u vyhledávání emailů nebo výrazů, je ještě možnost restart. Při stisknutí tohoto tlačítka dojde k vymazání nalezených emailů a odkazů, kde byly emaily nalezeny, a nebo odkazů, kde byly nalezeny hledané výrazy. Tímto se vymažou tabulky v databázi a uživatel má možnost začít znovu vyhledávat od začátku. Jakmile je spuštěno vyhledávání ať již tlačítkem start a nebo tlačítkem auto, jsou ostatní tlačítka vypnuty. Tím je zabráněno nevhodné manipulaci. Aktivní tlačítka jsou jen stop a nastavení plánovače. Uživatel má během stahování možnost měnit čas stahování. Ještě je nutné podotknout, že pokud je plánovač prázdný, tj. že neobsahuje žádný nastavený čas stahování, je tlačítko auto neaktivní.



Obr. 6: Hlavní okno crawleru.

6 Závěr

Při psaní této práce jsem musel čerpat nejen z česky psaných materiálů, ale bylo zapotřebí přečíst i některé texty psané v angličtině. Což je důkazem toho, že u nás v České republice není tato problematika tolik probírána jako v zahraničí. O psaní webových stránek je na našem trhu značné množství knih, ale o vyhledávačích většinou vychází jen knihy „jak vyhledávat“ ve vyhledávači a ne jak si napsat vlastní vyhledávač. V této práci jsem popsal zjednodušenou variantu běžných vyhledávacích strojů. Tento vyhledávač je ovšem jen ukázkový, jak by se takový vyhledávač mohl tvořit. Proto ani nereaguje na běžná omezení jako je robots.txt a metatagy, které zakazují průzkum webové stránky vyhledávacímu robotu. O těchto omezeních píši v teoretické části, ale již není náplní části praktické. Praktická část se především zabývá procházením stránek a vyhledávání emailů, nebo zadaného textu. Vyhledávání se dá také naplánovat. Zabývat se tímto crawlerem bylo poměrně zajímavé, při nejmenším si budu více vážit profesionálních vyhledávačů. Při tvorbě tohoto vyhledávacího robota jsem musel řešit řadu problémů, jednou z nich bylo jak správně vyseparovat emaily z textu. Dalším problémem by bylo indexování, které jsem musel pro jeho náročnost vynechat a věnovat se tím vyhledávacímu stroji, jenž vyhledává za běhu, a nemá předem připravenou databázi s výsledky. Přesto si myslím, že tento vyhledávací robot, i přes jistá zjednodušení oproti profesionálním, může sloužit ke specializovanému a personifikovanému vyhledávání. Po vhodných úpravách by mohl dále sloužit i jako příklad ve výuce, nebo k praktickému použití vyhledávání informací na internetu.

Reference

[1] SKLENÁK, V., a kol. *Data, informace, znalosti a Internet*. 1. vyd. Praha : C.H.Beck, 2001. ISBN 80-7179-409-0. s. 507.

[2] DEO, N., Gusta, P. *Graph-Theoretic Web Algorithms: An Overview, Proceedings of the International Workshop on Innovative Internet Computing*. , 2001. ISBN 3-540-42275-7. s. 91 – 102.

[3] MARKOV, Z., Larose, D.T. *Data Mining the Web: Uncovering patterns in web content, structure and usage*, John Wiley, New Jersey, 2007. ISBN 978-0-471-66655-4.

[4] KONCHADY, M., *Text mining application programming*. Media, Boston, 2004. ISBN 1-58450-460-9.

[5] KLEINBERG, J., *Authoritative sources in a hyperlinked environment*. *Journal of the ACM*, 46(5): 604-632, Sept. 1999.

[6] RAFIEI, D., MENDELZON, A., *What is this page known for? computing web page reputations*. In *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, The Netherlands, May 15-19, 2000.

[7] ZHANG, D., DONG, Y., *An efficient algorithm to rank web resources*. In *Proceedings of the Ninth World Wide Web Conference*, Amsterdam, The Netherlands, May 15-19, 2000.

[8] DEAN, J., HENZINGER, M., *Finding related pages in the world wide web*. In *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Canada, May 11-14, 1999.

[9] BHARAT, K., HENZINGER, M., *Improved algorithms for topic distillation in a hyperlinked environment*. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 104-111, Aug. 24-28, 1998.

[10] *Cesnet* [online]. 2008 [cit. 2008-04-24]. Dostupný z WWW: <http://www.cesnet.cz/provoz/technika.html>.

[11] *Regulární výrazy* [online]. 2008 [cit. 2008-04-24]. Dostupný z WWW: <http://www.regularnivyrazy.info/email.html>.

[12] *Seznam TLD domén* [online]. 2008 [cit. 2008-04-24]. Dostupný z WWW: <http://www.lupa.cz/texty/seznam-tld-domen/>.

Seznam příloh – Obsah CD

- Text bakalářské práce
  - BakalarskaPrace.pdf
- Program Downloader
- Záložní soubor databáze
  - Crawler.bak