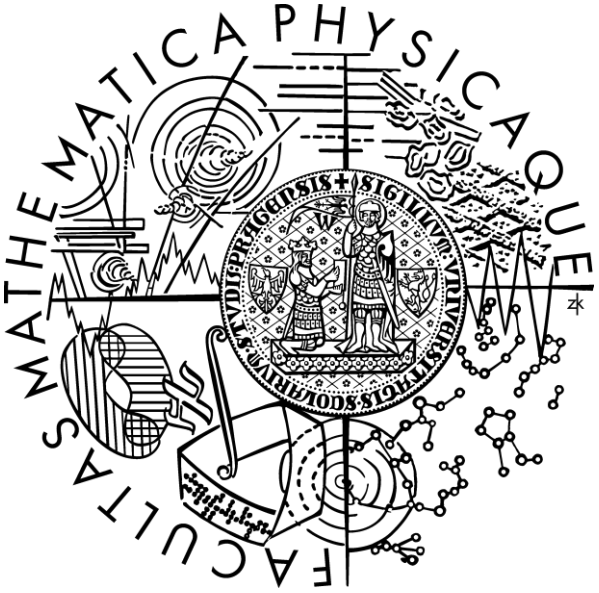
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jan Sochna

SYSTÉM PRO SBĚR XML DAT A METADAT Z INTERNETU

Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. David Bednárek, Ph.D.
Studijní program: Informatika, Softwarové systémy

Rád bych poděkoval především RNDr. Davidu Bednárkovi, PhD. za vedení této diplomové práce a za poskytnuté cenné rady v úvodu práce a při jejím dokončování. Také bych chtěl poděkovat rodině a přátelům za podporu při psaní.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 10. 3. 2010 Jan Sochna

Obsah

Název práce: Systém pro sběr XML dat a metadat z Internetu
Autor: Jan Sochna
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. David Bednárek, Ph.D.
E-mail vedoucího: bednarek@ksi.mff.cuni.cz
Abstrakt: Diplomová práce je zaměřena na návrh a implementaci systému pro sběr veřejně dostupných dokumentů z rodiny XML na Internetu. Záměrem je zautomatizovat a zjednodušit proces sběru dat a dosáhnout stažení kompletních struktur dokumentů z rodiny XML.
Na začátku práce byla provedena analýza čtyř systémů pro sběr dokumentů z Internetu, aby jeden z nich mohl být vybrán jako základ pro řešení diplomové práce. Jako nejvhodnější se ukázal open source webový crawler Apache Nutch. Nově byly navrženy a implementovány úpravy tohoto crawleru tak, aby byl efektivní při sběru XML dokumentů.
Pro zpracování stažených dokumentů byla využita aplikace Analyzer, která byla na základě testu na reálných datech upravena tak, aby zpracování těchto dat umožnila.
Hlavním přínosem diplomové práce je reálně využitelný systém pro sběr dokumentů z rodiny XML z Internetu. Díky rozšíření a úpravám crawleru Apache Nutch se podařilo podstatně eliminovat stahování a ukládání zbytečných dokumentů a zlepšit skladbu stažených dokumentů ve prospěch XML dat.
Klíčová slova: XML, Internet, reálné dokumenty, crawler

Title: Collecting XML data and meta-data from the Internet
Author: Jan Sochna
Department: Department of Software Engineering
Supervisor: RNDr. David Bednárek, Ph.D.
Supervisor's e-mail address: bednarek@ksi.mff.cuni.cz
Abstract: The Diploma Thesis is targeted to design and implement the system for collecting XML-family data from the Internet. The aim of the task is to automate the data collection process and download full structures of XML documents.
A comparison of four existing data collection systems took place at the beginning to choose one of the systems as a base of the solution. The open source web crawler Apache Nutch was identified as the most suitable. Then necessary extensions and modifications of the crawler were designed and implemented in order to make the crawler efficient in downloading XML-family documents.
Downloaded XML-family data were analyzed and evaluated using the Analyzer application, which was enhanced within this Diploma Thesis in order to process the data.
The main outcome of Diploma Thesis is an exploitable system collecting the XML-family documents from the Internet. Implemented modification and extensions of the system lead to elimination of „useless“ documents download, improving the ratio targeted XML-family documents.
Keywords: XML, Internet, real documents, crawler

1. Úvod

1.1 Motivace

XML získalo od doby své standardizace velkou popularitu a neustále se zvyšuje i jeho uplatnění při výměně informací. Jazyk XML je velmi obecný a nabízí mnoho jazykových konstruktů. Četnost využití těchto konstruktů je předmětem řady analýz reálných dat (např. [1, 13]).

Výsledky těchto analýz lze využít při vývoji nástrojů pro zpracování XML dokumentů. Vývoj je možné zaměřit na optimalizaci nejčastěji se vyskytujících konfigurací dokumentů.

Pro relevantnost výsledků analýzy je třeba získat reprezentativní vzorek používaných dat. Ten se získává většinou na základě poloautomatického sběru XML dat, kdy část dat je automaticky stažena pomocí programu, který prochází internetové stránky a stahuje obsah (crawler, též označován jako robot). Další část dat je stažena ze známých úložišť XML dokumentů a některé dokumenty jsou dohledány ručně (zvláště schémata dokumentů).

Dokumenty XML, DTD, XML schémata, XSLT programy, XQuery dotazy a další dokumenty založené na XML, jsou v této práci souhrnně označovány jako XML data.

1.2 Cíl a struktura práce

Cílem této práce je vývoj automatického nástroje pro sběr XML dat. Práce se soustředí na zlepšení procesu sběru volně dostupných dokumentů z rodiny XML z Internetu. Snahou je zautomatizovat a zjednodušit proces sběru dat a dosáhnout stažení kompletních struktur těchto dokumentů.

Tato práce navazuje na projekt Analyzer [11] vytvořený na MFF UK, na kterém se autor práce podílel. Aplikace z tohoto projektu analyzuje dokumenty v kolekcích (například reálné XML dokumenty) a nabízí i jednoduchý sběr dokumentů. Cílem je zajistit sběr XML dat pro analýzu v tomto nástroji s možností využití posbíraných dat i v jiných nástrojích. Součástí práce je proto tvorba rozhraní na Analyzer a v nezbytné míře i jeho úpravy, které odstraňují nedostatky bránící jeho bezproblémovému využití.

Zadání diplomové práce počítalo z praktických důvodů s tím, že základem řešení bude nějaký již existující nástroj pro sběr dat z Internetu, který bude upraven a rozšířen tak, aby co nejlépe splňoval požadavky pro sběr XML dat.

Realizace zahrnuje následující kroky:

- analýzu vlastností několika existujících systémů pro sběr dat z Internetu
- identifikaci předností a hlavních nedostatků těchto systémů z hlediska sběru XML dokumentů
- volbu systému, který se stane základem řešení diplomové práce

- návrh úprav a rozšíření zvoleného systému v souladu s požadavky zadání diplomové práce a jejich implementace
- formulaci námětů k dalšímu rozvoji vypracovaného řešení a analytického nástroje

Řešení navržené v této práci se musí vyrovnat s vlastnostmi a omezeními Internetu (existence nekorektních dat, permanentní proměnlivost obsahu, omezení nastavená ze strany poskytovatelů apod.).

Výsledkem je systém složený ze dvou částí, které tvoří ucelené řešení pro provádění analýz reálných XML dat.

První část sbírá data z Internetu. Jejím základem je upravený existující crawler, který řídí sběr dat, spravuje databázi adres ke stažení a umožňuje definovat intenzitu sběru dat. Do tohoto systému je zapojena detekce XML dat, vyhledávání odkazů na další XML data a prioritizace sběru XML dat před ostatním obsahem, jehož stahování a ukládání je naopak omezeno.

Druhou část tvoří aplikace Analyzer, která po provedených úpravách umožňuje zpracovat stažená data formou analýzy pro statistické účely nebo je uspořádat pro následné použití mimo Analyzer.

Po úvodní kapitole následuje kapitola 2, která obsahuje rozbor různých aspektů sběru dat z Internetu. Zaměřuje se na problémy, kterým crawler čelí a na možnosti jejich řešení. Kapitola 3 popisuje výběr architektury výsledného řešení. V kapitole 4 je popis celkové architektury a implementace jednotlivých částí řešení. V závěru práce je výsledné řešení porovnáno s výchozím stavem a známými řešeními a jsou prezentovány výsledky provedených testů.

2. Sběr a zpracování dokumentů

2.1 Vlastnosti Internetu

Dynamika

Internet se neustále vyvíjí, a proto ani jeho procházení pomocí crawleru není nikdy stejné. Mění se dostupnost serverů, mění se obsah stránek a mění se i nalezené odkazy. Proto je složité porovnávat strategie sběru dat. Změna úspěšnosti je vždy zkreslena chybou, kterou navíc nelze rozumně kvantifikovat.

Chybovost

Ne všechny dokumenty nalezené na Internetu mají z hlediska syntaxe korektní obsah. Část těchto chyb je způsobena například nekompletním stažením dokumentů nebo se jedná již o chyby ve zdrojových datech.

Některé tyto chyby lze následně odstranit. Například knihovna HTML Tidy dokáže vybrané chyby v HTML opravit. Podobně lze nalézt i chybné XML dokumenty a je třeba se s chybami vyrovnat. Opravou XML dokumentů se zabývá mj. projekt Analyzer.

Zkreslující vlivy

Část Internetu zůstává pro roboty neviditelná (skrytá). Standardní vyhledávače nejsou schopny tento obsah indexovat. To je dáno buď neexistencí odkazů na dokumenty v této části, nebo dynamickou povahou stránek. Z těch nelze informace získat buď bez položení vhodného dotazu, nebo kvůli omezení přístupu ke stránkám. Stahované dokumenty se tedy budou omezovat pouze na dostupnou část Internetu.

Dále procházení Internetu komplikují nekonečné struktury stránek *(Spider traps*), které způsobují plýtvání zdroji robota a nevedou ke zjišťovaným informacím [22]. Nezáleží na tom, zda vznikly úmyslně s cílem narušit práci robotů nebo neúmyslně technikou tvorby webu (například kalendář událostí). Těmto stránkám by se měl webový crawler vyhnout.

2.2 Sběr dat

Objem dat na Internetu je velký a zdroje, které jsou k dispozici pro jeho procházení, jsou limitované. Zejména se jedná o:

- čas, který je sběru dat vyhrazen;
- kapacitu úložiště, kam jsou stahovány dokumenty a kde se vytváří podpůrné struktury a dočasné soubory;
- síťové spojení, které má omezenou datovou propustnost. Sem patří i omezení dané infrastrukturou, pomocí které je připojení k Internetu realizované.

Tato omezení vedou k potřebě optimalizovat procházení internetového obsahu a výběr stahovaného a ukládaného obsahu [17].

2.2.1 Optimalizace procházení Internetu

Dokumentů z rodiny XML je na Internetu malé množství v porovnání s ostatními formáty. Největší zastoupení má formát HTML, který podle provedených testů (viz Obr. 6 na straně 30) tvoří 95% dokumentů. Ve zbývajících 5% stažených dat tvoří XML dokumenty cca 1/5. Z toho vyplývá, že zastoupení XML dokumentů činí cca 1%. Je tedy podstatné, jakou cestou budou žádoucí dokumenty identifikovány a staženy, resp. jak bude naloženo s dalším prohledaným obsahem.

Skóre

Webové crawlery používají k výběru relevantních stránek systém založený na hodnocení stránek. Každé je přiřazeno skóre, které vyjadřuje její důležitost a také vhodnost ke stažení. Vhodným uspořádáním stahovaných adres lze získat důležité stránky rychleji [17]. Obdobným způsobem upravíme sběr dokumentů z rodiny XML.

Problémem u skóre je jeho výpočet. Je třeba definovat jednotlivá kritéria výpočtu a přiřadit jim váhu. Příkladem sledovaných kritérií je např. počet odkazů směřujících na stránku, kvalita zdrojů těchto odkazů, počet odkazů na samotné stránce a výsledek analýzy stránky.

Algoritmy pro výpočet skóre lze rozdělit podle způsobu výpočtu na dávkové (offline) a průběžné (online).

Dávkový výpočet skóre stránek využívá znalosti všech doposud stažených stránek a odkazů mezi nimi. Příkladem tohoto algoritmu je PageRank. S rostoucím počtem dokumentů je tento výpočet náročný na prostor a paměť. Existují také vylepšení původního algoritmu, která snižují jeho výpočetní náročnost a umožňují výpočet paralelizovat (např. [18]).

Průběžný výpočet se snaží určit co nejpřesnější skóre hned po stažení dokumentu pouze na základě části informací o všech stažených stránkách a toto skóre postupně zpřesňuje při dalším stahování. Vstupními parametry jsou parsovaný[1] obsah dokumentu, z něj vycházející odkazy a část odkazů směřujících na tento dokument. Příkladem takovéhoto algoritmu je *On-Line Page Importance Computation* (OPIC) [19].

Při hledání XML dokumentů neplatí, že by se XML dokumenty vyskytovaly převážně na stránkách, které mají nejvyšší skóre při použití výše uvedených algoritmů. Skóre totiž není nijak svázané s nálezem XML dokumentů. Můžeme definovat novou metriku, pomocí které lze přiřazovat skóre adresám URL, stránkám a dokumentům podle toho, jak je pravděpodobné, že jsou to XML dokumenty nebo z nich na XML dokumenty vedou odkazy. Tato metrika musí dát vyšší skóre:

[1] Parsování dokumentu zahrnuje analýzu textu, která vede k nalezení specifických informací (například odkazů)

- XML dokumentům označeným při parsování,
- dokumentům odkazovaným z označených XML dokumentů,
- dokumentům, jejichž adresa obsahuje známou příponu XML dokumentů.

2.2.2 Filtrace

Do systému vstupuje velké množství dokumentů a ne všechny dokáže systém zpracovat. Adresy takových dokumentů je potom vhodné odfiltrovat, aby se snížila zátěž cílových serverů a neplýtvaly se systémové zdroje aplikace.

Před samotnou filtrací adres se provádí jejich **normalizace**. Cílem tohoto procesu je, aby každá adresa v systému měla pokud možno jen jednu reprezentaci, a tím se zamezilo duplicitnímu stažení stejného dokumentu. Normalizace je popsána ve standardu Uniform Resource Identifier [12].

Odfiltrovat je možné dokumenty, které nejsou z rodiny XML, a zároveň v nich neexistují odkazy na tyto dokumenty. Je tedy možné filtrovat například multimédia (obrázky, hudbu a videa), *richtext* dokumenty (DOC, PDF, RTF), spustitelné soubory a další binární formáty. Problémem je rozeznat tyto dokumenty ještě před jejich samotným stažením, protože jinak nedojde k úspoře na síťové infrastruktuře ani na cílových serverech. Přesto i filtrace po stažení dokumentů pomáhá a redukuje prostorovou náročnost sběru dat.

Dokumenty je možné filtrovat na základě rozpoznané **dobře známé přípony** dokumentu v adrese. Jedná se o heuristický postup. Ve skutečnosti přípona stoprocentně neurčuje typ a obsah dokumentu.

Dále je možné filtrovat některé **neplatné odkazy**, které obsahují například volání funkcí javascriptu a tím snížit počet dotazů, které by server vyhodnotil jako chybné požadavky.

HTML dokumenty není možné filtrovat před stahováním, neboť právě v nich se nachází většina odkazů na další dokumenty. Tyto dokumenty je ale možné po nalezení odkazů odstranit (neuložit).

Další možností šetření síťových zdrojů a úložného prostoru je **omezení velikosti stahovaných dokumentů**. To lze řešit dvěma přístupy. Buď lze větší dokumenty nestahovat (podle velikosti deklarované v hlavičce odpovědi) nebo je stahovat částečně, tj. po stažení nastaveného objemu stahování přerušit nebo přímo vyžádat pouze omezenou velikost obsahu (jak dovoluje specifikace HTTP 1.1)[1].

2.2.3 Identifikace vazeb mezi XML dokumenty

Chceme-li stahovat kompletní struktury XML dokumentů, je třeba detekovat vazby mezi dokumenty. Pouze jedno existující řešení (Xyleme, popsané v kapitole 3.1.1) vyhledává omezeným způsobem odkazy v XML dokumentech (pouze DTD) [28].

Odkazy mezi dokumenty můžeme rozdělit na ty, které vedou k definici struktury dokumentu nebo jeho stávající definici rozšiřují (označme je *silné odkazy*),

[1] HTTP – Hyper Text Transfer Protocol

a odkazy, které vedou na zdroje, které nepatří do rodiny XML (označme je *slabé odkazy*).

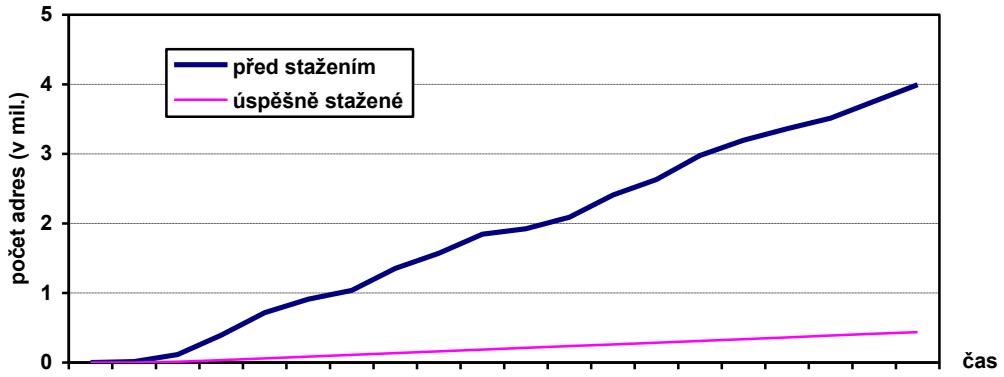
Silné odkazy by měly být stahovány přednostně, ale zároveň nikoliv opakovaně. Slabé odkazy jsou k volitelnému následování a mělo by se s nimi zacházet podobně jako s ostatními nalezenými odkazy v HTML dokumentech.

V následujícím textu se budeme soustředit na silné odkazy, které jsou důležité pro sběr kompletních struktur XML dokumentů. Po prostudování doporučení a specifikací formátů blízkých XML, lze v těchto formátech nalézt níže uvedené konstrukty reprezentující odkazy na jiné XML dokumenty [23–26].

- XML document – *schemaLocation, noNamespaceSchemaLocation, external DTD*
- XML Schema – *import, include, override*
- XSL Transformation – *import, include*
- XML Style Sheet – Processing instruction XML
- RelaxNG – *include, externalRef*

2.2.4 Omezení databáze adres

S rostoucím počtem stažených stránek roste také databáze adres, kterou crawler eviduje (viz Obr. 1). V rámci testování (viz. kapitola 5.2) na 0,44 mil. stažených stránek připadlo 4 mil. nestažených stránek. To odpovídá průměrným cca 10 odkazům objeveným na jedné stránce nebo dokumentu. Databáze adres se tedy rozrůstá podstatně rychleji, než jak přibývají stažené dokumenty. S rostoucí databází adres se prodlužují veškeré operace s touto databází.



Obr. 1 – Vývoj počtu adres v databázi v průběhu sběru

Kromě dříve uvedené filtrace lze pro omezení rozsahu databáze adres využít řadu dalších přístupů, které jsou popsané níže.

Ignorování odkazů v rámci serveru

Pro procházení se uvažují pouze odkazy, které směřují na jiné servery. Pomocí této filtrace lze rychle získat databázi kvalitních zdrojů dat, která ale nevyjadřuje zajímavost těchto dat (vztah ke XML).

Omezení počtu adres pro server v databázi

Každý server může mít v databázi pouze omezený počet adres dokumentů. Vložení nové adresy nad tento limit již není možné. Tuto filtraci je třeba kombinovat s tříděním adres podle jejich skóre, aby nebyly jako první vyřazeny adresy, které potenciálně vedou na XML dokumenty.

Vhodným doplněním je úprava, která v případě nálezu XML dokumentu zvýší kvótu domény, na které byl dokument nalezen, a zavedení absolutního omezení počtu adres pro doménu.

Dohromady tak lze redukovat zátěž serveru, případně i zkreslení výběru, způsobené velkým množstvím XML dokumentů z jediného zdroje. Současně lze tím omezit vliv *spider traps*.

2.2.5 Přetížení lokálních serverů

Web crawler prohledávající Internet do šířky generuje velké množství požadavků na DNS server[1]. Vzhledem k malé velikosti DNS cache v operačním systému, případně její absenci, se proto velmi často sběr dat provozuje na stroji, na kterém současně běží lokální DNS server. Ten vyřizuje požadavky crawleru a udržuje dlouhodobou cache výsledků.

Při paralelním sběru dokumentů pomocí více crawlerů, nakonfigurovaných pro využití v clusteru, může být DNS server nejužším místem v celé architektuře a intenzita požadavků může dokonce vyústit v odmítnutí služby (Denial of Service – DoS) na DNS serveru.

Konfigurace, která vede k vysoké rychlosti sběru dat, by proto měla být konzultována se správcem sítě, aby se předešlo přetížení serverů a narušení provozu sítě.

2.3 Zpracování dat

Pokud již máme k dispozici data, je třeba vyřešit, jak je uspořádat, aby je bylo možné dále analyzovat. V určité míře tyto problémy řešil projekt Analyzer, na který tato práce navazuje.

Jeden ze záměrů této práce je ukázat, jak se dokáže Analyzer vypořádat s daty z heterogenního zdroje, jakým Internet je. Práce má ukázat jeho případná slabá místa a navrhnout jejich řešení.

[1] Server zajišťující překlad symbolického názvu (doménového jména) serveru na jeho IP adresu.

2.3.1 Indexace

Potřebujeme-li v existující kolekci dat vyhledávat informace, nabízí se přístupy využívající fulltextové vyhledávání nebo využívající mapovaní do relační databáze či nativní XML úložiště spojené s indexací.

Indexaci je možné využít k vyhledávání podobných či souvisejících dokumentů. Pokud ke každému dokumentu budou evidovány použité jmenné prostory (*namespaces*), bude potom možné seskupit dokumenty, které mají kvůli použití stejného schématu podobnou strukturu. Příkladem ručně vytvořené databáze, která indexuje využitá schémata, je XML Datasets [21].

Další možností indexace je vypustit elementy nebo je nahradit pouze jejich názvem a zachovat jejich obsah. Takto vytvořený text, zbavený značek, by mohl sloužit pro fulltextové vyhledání.

2.3.2 Správa kolekcí

Práci s kolekcemi dokumentů řeší projekt Analyzer. Umožňuje třídit dokumenty na základě analýzy a na základě sady parametrizovatelných filtrů. Analýzy a kolekce jsou programovatelné. Stávající možnosti rozřazování je tedy možné dále rozšířit pomocí nových pluginů.

Takto uspořádaný systém umožňuje analyzovat data dvoustupňově. Primární analýza a filtrace se provádí již během sběru dat. Při zpracování v Analyzeru lze dojít k přesnějším výsledkům a filtrovat i dokumenty, které nesprávně prošly první filtrací.

Aby bylo možné dokumenty v kolekcích využít při dalším zpracování, musí Analyzer umožnit export dokumentů v kolekcích. Tato funkcionalita v původní verzi Analyzeru chybí a je třeba ji pro potřeby této diplomové práce doplnit.

2.4 Ostatní aspekty prohledávání a sběru dat z Internetu

2.4.1 Etika

Při použití crawleru dochází ke konfliktu mezi provozovateli webových serverů a těmi, kdo touto cestou studují Internet. Výzkum potřebuje v omezeném čase získat co nejvíce hledaných dat. Provozovatelům webových serverů to způsobuje náklady ve formě využitých zdrojů (síťové prostředky a výpočetní výkon procesoru).

Společenský přínos a další etické aspekty procházení Internetu podrobně rozebírá práce Web Crawling Ethics Revisited [3], která například tvrdí, že akademický výzkum měřící stav dat na Internetu působí střední zátěž pro servery a má pouze nepřímý společenský přínos. Komerční vyhledávače (př. Google, Seznam) oproti akademickému výzkumu mají přímý společenský přínos, a to jak formou zpřístupnění informací uživatelům, tak směrováním uživatelů k provozovatelům webových stránek.

Možností regulace automatického procházení webových stránek se zabývá doporučení The Robots Exclusion Protocol [2] (*robots.txt*). To umožňuje

provozovatelům webových serverů nastavit, které části webu mohou roboti procházet.

Řešení navržené v této práci respektuje uvedené doporučení. V opačném případě by totiž hrozilo, že stroj či doména, ze které sběr dat probíhá, bude zablokována.

2.4.2 Copyright

Webový crawler stahuje dokumenty a ukládá si jejich kopie bez svolení majitele. Tím může porušit copyright a odpovídající právní předpisy na ochranu autorských práv. Problém může vyvstat například při zveřejnění stažených dat.

Podle studie Web Crawling Ethics Revisited [3] zatím funguje systém volitelného vyloučení (opt-out policy) problémového obsahu. Provozovatelé internetových stránek mohou pomocí pravidel v *robots.txt* vyloučit chráněnou část obsahu a ostatní dotčení mohou požádat o vyloučení problematických dokumentů.

3. Návrh řešení

3.1 Sběr dat

V této kapitole je popsán postup a výsledek výběru programového základu pro řešení sběru dokumentů z rodiny XML z Internetu.

Na začátku práce byly vytipovány celkem čtyři systémy pro sběr dat: Xyleme (Larbin), Egothor, Apache Nutch a Bixo. Ty byly podrobeny analýze a na základě jejich vlastností a perspektiv z nich byl následně vybrán základ pro sběr dat.

Zvažována byla i možnost využití webových služeb.

3.1.1 Existující řešení

Xyleme a Larbin

Xyleme je komerční projekt komplexního úložiště XML dat [4, 28]. Projekt vznikl v roce 2000. Řeší sběr dat z Internetu a ze zdrojů zadaných zákazníky. Dokumenty spravuje v úložišti a snaží se je udržet aktuální. To zajišťuje pomocí opakovaného stahování crawlerem a systému vynucených aktualizací (*push update*). Kromě sběru a uložení dat umožňuje také provádění dotazů nad dokumenty. Podle dostupné dokumentace spravuje pouze dobře formované XML dokumenty a DTD.

Robot z tohoto projektu se jmenuje Xyro (Xyleme robot) a není volně dostupný. Správci projektu pouze zpřístupnili část dat pro výzkumný účel [13].

Nekomerční verze robota je dostupná pod názvem Larbin [5]. Jedná se však pouze o základní web crawler bez jakéhokoliv rozšíření směrem ke stahování a analýze XML dat. Poslední zveřejněná verze tohoto robota je z roku 2003.

Vzhledem k tomu, že Xyleme není volně dostupný, nebyl uvažován jako základ pro sběr dat.

Egothor

Egothor [7] je webový crawler původně vyvíjený na Matematicko-fyzikální fakultě Univerzity Karlovy (MFF UK), který nabízí kontinuální stahování, indexování a možnost vkládat nové adresy ke stažení bez nutnosti přerušení sběru. Zároveň se soustředí na efektivní indexaci a vyhledávání v indexovaných datech.

Významnou vadou poslední uvolněné verze systému Egothor je to, že je zastaralá. Verze 1.3 byla vydána v roce 2005 a nebyla již dále aktivně vyvíjena. Nemá kompletní dokumentaci API a má omezenou dostupnou literaturu. Nová hlavní verze má kompletně nahradit původní verzi 1.3, ale zatím není jisté, zda nebo kdy bude uvolněna.

Egothor odkazy vyhledává pouze v HTML dokumentech, ačkoliv indexaci podporuje i pro další typy souborů. Obsah dokumentů nad pevně zadaný limit není ukládán. Data se ukládají do samostatných komprimovaných souborů. To vede k úložišti s tisíci malými soubory. Manipulace s takto rozsáhlým úložištěm

není jednoduchá. V nové verzi by měl Egothor využívat pro ukládání dokumentů projekt Docserver [8], který optimalizuje úložný prostor.

Apache Nutch

Open source webový crawler Apache Nutch [9] vznikl před rokem 2003. Kombinuje v sobě robota pro sběr dokumentů a indexaci s vyhledáváním, kterou zajišťuje knihovna Apache Lucene. Celé řešení je distribuovatelné díky knihovně Apache Hadoop (ta vznikla právě kvůli projektu Nutch). Od svého vzniku se projekt dopracoval k vydání verze 1.0 v březnu 2009. Vývoj dalších verzí pokračuje a vydání verze 1.1 se očekává v první polovině roku 2010.

Standardní systém je rozšiřitelný pomocí soustavy pluginů a rozhraní (*extension points*). Z pohledu této práce jsou zajímavá rozšíření pro filtraci adres (URLFilter), hodnocení adres a dokumentů (ScoringFilter), parsování dokumentů (Parser) a pro indexaci (IndexingFilter).

Data se ukládají v indexovaných sekvenčních souborech. Odděleně jsou ukládány informace o nalezených odkazech, výsledcích parsování a samotný obsah dokumentů. Dokumenty větší než nastavený limit jsou zkráceny. Toto úložiště může být distribuované.

Bixo

Bixo je autory charakterizován jako nástroj pro získávání dat z webu, zaměřený na tok dat [10].

Bixo, stejně jako předchozí systémy, obsahuje také vlastní web crawler. Internet neprochází do šířky, jako je to obvyklé u robotů pro velké vyhledávací systémy, ale do hloubky pro zvolené domény. Jedná se o *topical crawler*. Zaměřuje se na extrakci a další zpracování dat obsažených ve stažených stránkách.

Nad staženými daty neposkytuje žádnou formu vyhledávání, a tedy neprovádí ani indexaci. Podle propozicí [10] se dokáže vyrovnat i s nástrahami Internetu ve formě *spider traps*. Konkrétní způsob jejich řešení již v uvedeném materiálu zmíněn není.

Google

Google je jeden z největších vyhledávačů. Během indexování Internetu narazí na miliardy dokumentů, mezi kterými objeví i velké množství dokumentů z rodiny XML, jejichž adresy by bylo užitečné získat. K tomu by ovšem bylo nutné najít vhodně formovaný dotaz, který by vracel dokumenty z rodiny XML a zároveň vracel co nejmenší počet *false positives* (dokumentů mimo rodinu XML).

Do analýzy je Google začleněn jako doplněk a alternativa k předchozím řešením. S vhodným API a licenčními podmínkami by mohl být cenným zdrojem informací při sběru XML dat.

Lze zkonstruovat dotaz pro vyhledávání, který vrací některé dokumenty z rodiny XML. Adresy takto vrácených dokumentů by byly vhodné pro vložení do databáze adres crawleru, protože by umožnily okamžité zvýšení výtěžnosti sběru určitého typu dokumentů. S dostatečným počtem různých dotazů by tak bylo možné získat širokou paletu dokumentů.

Bohužel Google ve svých podmínkách pro používání služby [16] nedovoluje automatické dotazování. Lze jej ale využít pro ruční sběr dat.

To je způsob vhodný pro nalezení méně frekventovaných typů souborů, jako jsou například dotazy XQuery. Některé je možné získat například dotazem: „*filetype:xq AND doc*“. Úspěšně byly rovněž vyzkoušeny i dotazy: „*targetNamespace filetype:xsd*“, „*entity filetype:dtd*“ pro získání XML schémat a DTD.

3.1.2 Shrnutí existujících řešení

Egothor, **Nutch** a **Larbin** se orientují na HTML a obsah čitelný pro člověka. Zaměřují se na fulltextovou indexaci a efektivní vyhledávání. Do tohoto konceptu tak nezapadají XML dokumenty, které mají spíše význam pro automatické zpracování.

Bixo se zaměřuje na konkrétní analýzu již během provádění sběru dat. Tj. nejedná se o sběr dat pro pozdější použití, ale spíše o sběr dat a řešení konkrétního problému. Proto se nejeví jako vhodný nástroj pro vyhledávání dokumentů z rodiny XML.

Xyleme ani **Google** nelze využít kvůli uzavřenosti a licenčním podmínkám.

3.1.3 Volba základu sběru dat

Vyloučíme-li nedostupný komerční nástroj Xyleme, zastaralý Larbin a Google, který je z hlediska automatického sběru dat nepoužitelný, je možné uvažovat pro základ sběru dat buď systém Egothor nebo Apache Nutch.

Egothor má několik nedostatků, které jej prakticky diskvalifikují z možného budoucího využití jako základu řešení sběru dat pro tuto práci. Jsou to zejména:

- neúplná dokumentace,
- výhled, že současná verze bude kompletně přeprogramovaná, a tedy úpravy provedené na této verzi budou pro příští verzi nepoužitelné.

Apache Nutch nabízí dobré zázemí vyzrálého open source projektu. API tohoto robota je stabilizované, a mělo by být podporováno alespoň do změny hlavní verze. Vývoj nad tímto systémem má tedy perspektivu do budoucna.

S ohledem na výše uvedené srovnání vychází hodnocení lépe pro aktuální a dostupný systém Nutch, a proto byl tento systém zvolen jako základ pro sběr dat v této diplomové práci.

3.2 Platforma a architektura

Řešení je postaveno na platformě Java, což je dáno návazností na projekt Analyzer a výběrem systému Nutch jako základu sběru dat. Systém Nutch má podporu pro běh na *NIX systémech a z tohoto důvodu je práce vyvíjena na systému Linux.

Celý systém se skládá ze dvou částí. První část tvoří crawler Apache Nutch, zvolený v předchozí kapitole, který umožňuje procházet Internet do šířky a vyhledávat i stahovat dokumenty z rodiny XML. Systém je v rámci této práce

postupně rozšiřován pomocí pluginů a podle potřeby je funkce systému dále rozšířena či doplněna. Současně je optimalizována konfigurace systému pro sběr XML dat. Druhou částí je aplikace Analyzer, která nalezené dokumenty třídí do kolekcí a umožňuje jejich zpracování. Práce vzájemně propojuje obě aplikace a tím umožňuje dokumenty stažené z Internetu analyzovat.

Spojení mezi sběrem dat a aplikací Analyzer bylo možné řešit dvěma způsoby. Bylo možné buď využít rozhraní pro komponentu *Crawler*, vytvořené v aplikaci Analyzer a sběr dat provádět přes grafické rozhraní, nebo vyvinout sběr dat nezávisle na Analyzeru a pro předávání dat využít export a import.

Jelikož grafické rozhraní omezuje možnosti pro vzdálené spouštění sběru dat a nepodporuje skriptování, je aplikace pro sběr dat vyvíjena samostatně jako konzolová. Stažené dokumenty jsou pak importovány do aplikace Analyzer.

4. Popis implementace

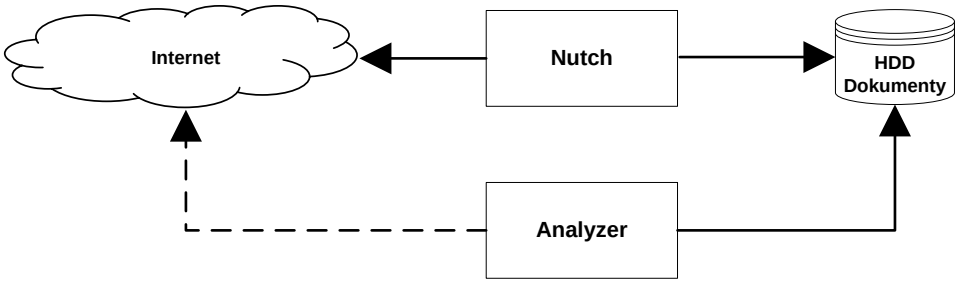
4.1 Architektura systému

Systém využívá funkčnosti dvou hlavních aplikací (viz Obr. 2).

Jádrem sběru dat je upravený a rozšířený Apache Nutch. Pomocí něj je možné stahovat data z Internetu. Stahovaná data jsou organizována do segmentů. Segment vzniká generováním seznamu adres pro stahování a reprezentuje jeden cyklus při sběru dokumentů z Internetu. Po generování adres následuje stahování dokumentů z těchto připravených adres. Obsah vybraných stažených dokumentů (první filtrace) a dalších metadat se ukládá do stejného segmentu jako generované adresy.

Pro zpracování jsou data ze segmentů vložena do aplikace Analyzer. Ten obsahuje rozhraní pro import dokumentů, které slouží pro zadávání vstupních dat. Původně byl Analyzer schopen importovat data pouze ze souborového systému, kde každý importovaný soubor musel být uložen samostatně. Pro potřebu diplomové práce bylo toto rozhraní rozšířeno pro import dat ze segmentů se staženými daty.

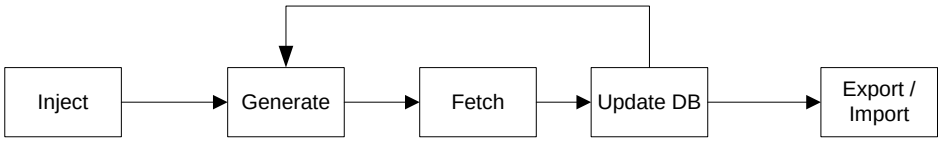
Analyzer umožňuje importovaná data analyzovat a na základě výsledků je třídit do kolekcí. Na začátku analýzy jsou přesněji detekovány dokumenty z rodiny XML (druhá filtrace) a pouze tyto dokumenty jsou dále analyzovány. Dokumenty uspořádané do kolekcí je možné z aplikace dále exportovat pro potřebu analýzy i mimo Analyzer.



Obr. 2 – Schéma architektury

Crawler

Proces stahování názorně ukazuje Obr. 3. Proces stahování zahrnuje: vložení úvodních adres do databáze (inject) a neomezený počet cyklů: generování nejlepších adres pro stahování (generate), stažení těchto adres (fetch), aktualizace databáze (updatedb). Následně je možné stažená data zpracovat.

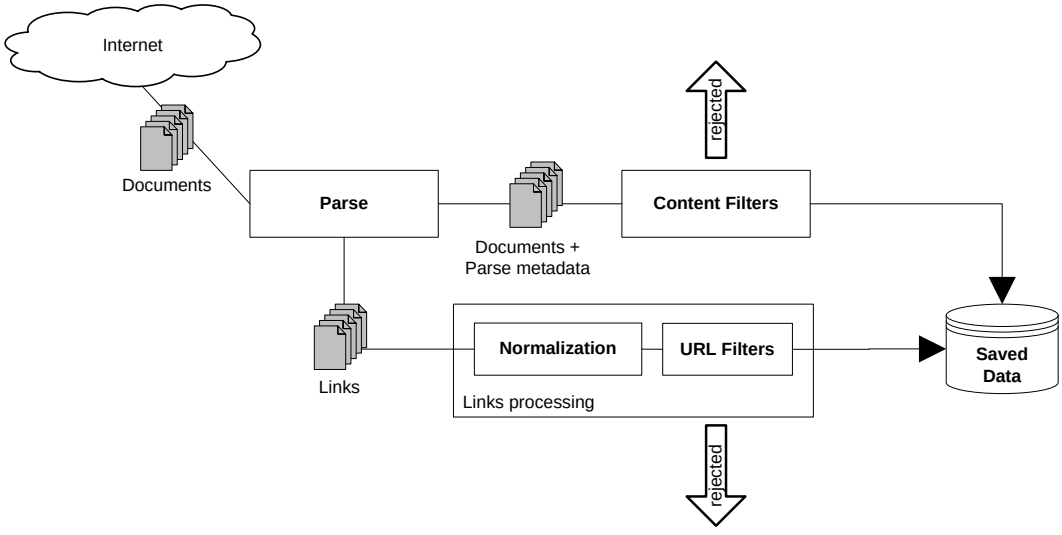


Obr. 3 – Schéma procesu sběru dat[1]

O výběr adres se stará Nutch komponenta *Generator*. Generované adresy jsou ovlivněné algoritmem použitým pro výpočet skóre, který využívá heuristiky k ovlivnění pořadí adres (např. známé přípony dostávají lepší pořadí). Výběr adres upravuje také plánovač (*scheduler*), který ovlivňuje, jak často jsou adresy stahovány.

Samotné stahování adres zajišťuje Nutch komponenta *Fetcher*. Adresy ke stažení jsou načteny do fronty, odkud je jednotlivá vlákna vybírají a stahují. Fronta respektuje nastavenou maximální intenzitu požadavků na server. Informace o stažení je ihned uložena. Stažené dokumenty jsou parsovány a tehdy se také hledají odkazy v dokumentech. Tyto informace jsou následně uloženy. Před ukládáním prochází dokumenty filtrací. Na závěr celého stahování *Fetcher* všechny uložené informace roztřídí. Dochází například také k filtraci objevených odkazů na základě URL filtrů. Zpracování dokumentů při sběru ukazuje Obr. 4.

Po stažení je třeba aktualizovat databázi adres podle informací ze staženého segmentu. Tento proces řídí Nutch komponenta *CrawlDb*. Zapisuje do databáze poslední pokus o stažení, jeho výsledek a nově nalezené odkazy. K jednotlivým záznamům se počítá nové skóre na základě výsledku parsování a nalezených odkazů.



Obr. 4 – Schéma zpracování dokumentů při sběru

[1] Anglické termíny použité v obrázcích jsou převzaty z terminologie používané v Apache Nutch.

Analyzer

Pro analýzu a třídění dat slouží aplikace Analyzer. V rámci této práce se do ní data vkládají importem stažených segmentů.

V aplikaci Analyzer je možné využít různé pluginy, které poskytují analytické funkce pro výpočet a vizualizační funkce pro prezentaci výsledků. Analýza se konfiguruje pomocí výběru jednotlivých analytických metod (*Analysis*) a jejich uspořádáním do *Assembly*. Sem patří analytické metody *Detector*, *Corrector*, *Tracer* a *Analyzer*.

Detector slouží k rozpoznání typu dokumentu. Podle tohoto typu se spouští další analýzy podle *Assembly*. *Corrector* slouží k opravě chybných dokumentů, *Tracer* vyhledává v dokumentech odkazy a nakonec *Analyzer* dokumenty analyzuje a ukládá výsledky této analýzy.

Na základě výsledků analýzy probíhá rozřazení dokumentů do kolekcí (*Collection*). Jedna skupina kolekcí se nazývá *Cluster* a za rozřazování dokumentů do kolekcí v jednom *Cluster*u je zodpovědný *Collector*.

S kolekcemi je možné dále pracovat, zejména generovat statistiky nebo exportovat dokumenty z kolekce.

Podrobnější popis aplikace je dostupný v dokumentaci k aplikaci [6].

4.2 Crawler

V této kapitole jsou rozebrány podrobnosti implementace crawleru. Zejména pak úpravy tohoto systému stahování oproti vydané verzi Apache Nutch. Zdrojové kódy byly získány na stránkách projektu [9].

4.2.1 Filtrace adres

Každá adresa vstupující do systému nejprve prochází normalizací. Pozdější porovnávání adres probíhá textově a tak je třeba reprezentaci sjednotit co nejdříve.

Normalizované adresy následně podléhají filtraci. Nutch obsahuje předpřipravené pluginy pro filtraci URL adres, které vstupují do systému. Jejich přehled je spolu s typem filtrace uveden v následující tabulce.

DomainRegexpFilter	filtrace vybraných domén a subdomén (whitelist)
PrefixURLFilter	filtrace podle použitého schématu URL (whitelist)
SuffixURLFilter	filtrace podle zakončení URL omezuje stahované typy souborů (blacklist)
RegexpURLFilter	filtrace na základě regulárních výrazů (blacklist+whitelist)

Tabulka 1 – Přehled pluginů v systému Nutch pro filtraci URL adres

Pro potřebu sběru reálných XML dat jsou využity všechny filtry kromě filtru domén, který by přinesl zúžení záběru analýzy. Konfigurace byla upravena tak, aby nebyly stahovány dokumenty, které nemají přínos ke sběru XML dat.

Prefix filtr je využit pro filtraci adres, které je systém schopen stáhnout. Jedná se o adresy s protokolem HTTP a FTP[1]. Díky tomu vyřadí URL adresy se schématem *mailto* nebo *javascript*.

Suffix filtr je použit pro vyloučení adres dobře známých typů multimédií (AVI, MPG, MP3), spustitelných programů (EXE), *richtext* dokumentů (DOC, PDF, RTF) a dalších.

Filtr využívající **regulárních výrazů** umožňuje vyhnout se některým formám *spider traps* a dále filtruje adresy na základě klíčových slov uvnitř adresy. Například přidaný zákaz výskytu závorek v adrese zamezil vstupu adresám, které obsahují volání funkce javascriptu. Tím se předchází stahování dokumentů, které by skončilo chybou.

4.2.2 Filtrace dokumentů

Statistika stažených dat ukazuje, jaké množství dat je nakonec zajímavé resp. zcela nezajímavé v porovnání se staženým objemem. Jak již bylo uvedeno, rozhodující část dokumentů (cca 95%) tvoří HTML stránky. Tato data jsou zapotřebí jen pro vyhledání odkazů na nové stránky, proto je není třeba ukládat.

Vyřazením těchto nepotřebných dokumentů se kromě celkového objemu výsledných dat také výrazně sníží objem dat udržovaný v dočasných souborech během stahování. Dočasné soubory navíc nejsou komprimované, a tak se na nich plně projeví úspora zahozených dat. Díky tomu je možné analyzovat s omezeným diskovým prostorem větší část internetového obsahu.

Nutch obsahuje funkcionalitu pro omezení velikosti jednotlivých stahovaných dokumentů. Velké dokumenty zkracuje podle nastaveného limitu. Tento postup je vhodný, protože XML dat je na Internetu jen malé množství a pro získání odkazů není třeba stahovat celé HTML stránky. Díky tomu lze se stejnou kapacitou sítě stáhnout větší množství dokumentů. Bohužel XML data jsou také krácena a je vhodné je stáhnout znovu s alternativní konfigurací crawleru (s využitím funkce popsané v kapitole 4.3.2). Nicméně opakované stažení se tyká méně než 1/10 dokumentů z původního počtu stažených (opětovně nejsou stahovány HTML dokumenty a další nevhodné formáty).

Pomocí parametru konfigurace „fetcher.content.store“ lze nastavit, zda se má vůbec nějaký obsah dokumentů ukládat. Toto nastavení ale není dostatečné. Proto byl přidán další mechanismus, pomocí kterého lze ukládání ovlivnit. Význam původního parametru konfigurace však zůstal zachován.

Optimalizovat filtraci dokumentů je možné dvěma způsoby, popsanými dále s jejich výhodami a nevýhodami. Jejich vizualizaci zachycuje Obr. 5 na straně 23.

Whitelist

Jako první se pro řešení problému s velkým objemem stažených a uložených dat nabízí filtrace na základě pozitivní identifikace hledaného vzoru. Filtraci lze

[1] FTP – File Transfer Protocol

provádět na základě typu staženého dokumentu (MIME typ[1]) nebo na základě výsledku parsování.

Při úspěšné detekci je k výsledku připojena značka. Přítomnost značky je následně testována před ukládáním. V případě její absence nebude obsah dokumentu uložen a zachová se pouze informace o stažení dokumentu a odkazech vedoucích z tohoto dokumentu.

Konkrétně je do metadat (*ParseMetadata*) pozitivně identifikovaného dokumentu vložen záznam „XML_BASED“ s hodnotou „true“.

Pokud je filtrace prováděna zjednodušeně, jako v našem případě (s cílem omezit výpočetní náročnost), dochází k nerozeznání části vyhovujících dokumentů a ty jsou zbytečně odfiltrovány.

V testu se nedostatečně konfigurovaný whitelist projevil uložením úzké množiny dokumentů, zatímco využitelných dokumentů bylo mnohem více.

Výhodou tohoto řešení je minimalizace prostoru potřebného pro uložení dokumentů. Jeho nevýhodou je, že i malá odchylka může způsobit zamítnutí dokumentu, ačkoliv by do zpracované kolekce měl patřit.

Jako základní problém použití whitelistu se jeví, že v době psaní filtru nejsou zpravidla známy všechny typy vyhledávaných dokumentů.

Konfigurace je možná pomocí položky nastavení:

Název	Hodnota	Popis
fetcher.store.content.interesting	{true | false}	zda je whitelist povolený

Ve výsledném řešení této práce je použití techniky whitelist v konfiguraci deaktivováno a používá se následující řešení.

Blacklist

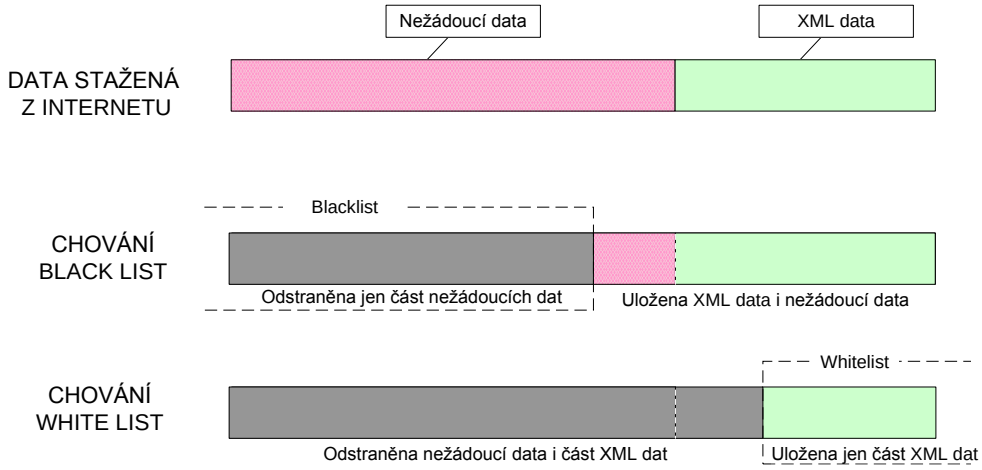
Blacklist byl zaveden hlavně pro omezení velkého objemu, který zabírají HTML a XHTML dokumenty. Implementované řešení testuje MIME typ ukládaného dokumentu proti zadanému regulárnímu výrazu a v případě shody zabrání uložení dokumentu.

V konfiguraci se definují tyto položky:

Název	Hodnota	Popis
fetcher.store.content.skip.unsuitable	{true | false}	zda je filtr povolený
fetcher.store.content.unsuitable.pattern	regulární výraz	regulární výraz pro zakázané MIME typy

[1] MIME typ – identifikátor formátu dokumentu [14]

Chování whitelist a blacklist přibližuje následující obrázek.



Obr. 5 – Porovnání metod blacklist a whitelist na stažených datech

4.2.3 Moduly parseru

Parsování je řízené rozpoznaným MIME typem dokumentu. Typ je rozpoznáván pomocí externí knihovny Tika parser [15]. Originální systém Nutch má jednoduchou funkčnost pro asociaci parseru s typem zpracovatelných dokumentů. Každý plugin na parsování lze svázat s jediným MIME typem nebo lze parser definovat jako výchozí. Výchozí parser je použit automaticky, když specifické parsery (parser určený pro shodný typ dokumentu) nevrátí žádný výsledek nebo žádný specifický parser není znám. Pokud vstup neodpovídá dovednosti výchozího parseru, musí vrátit prázdný výsledek, aby mohl být spuštěn další parser v pořadí.

Pokud by na tento systém bylo napojeno parsování různých typů XML dokumentů, muselo by být zajištěno, aby MIME typ všech těchto dokumentů byl stejný nebo napsat parser, který bude akceptovat všechny typy dokumentů.

Pokus o sjednocení detekce typu dokumentu selhal, když Tika parser nevhodně detekoval část XHTML jen jako jednoduchý XML dokument. Kvůli nevhodné detekci by bylo třeba, aby stejný kód, který zajišťuje parsování HTML, byl napsán i pro parsování XML dokumentů. Proto je raději využit vzdáleně hlášený MIME typ a pouze pokud není uveden, využije se detekce pomocí Tika.

Všechny XML dokumenty nepoužívají jediný MIME typ „application/xml“. Místo toho jsou dokumenty založené na XML označeny většinou typem s příponou „+xml“ (např. „application/xslt+xml“).

Aby bylo možné asociovat plugin pro parsování s více MIME typy, byl Nutch rozšířen o možnost zadat příslušnost pluginu k MIME typu pomocí regulárního výrazu.

Hledání parseru pro konkrétní typ dokumentu potom probíhá v následujícím pořadí, a to do nalezení prvního, který nevrátí chybový výsledek:

1. MIME typ dokumentu a pluginu se přesně shodují
2. MIME typ dokumentu odpovídá regulárnímu výrazu akceptovaných typů pluginu
3. plugin nedefinuje regulární výraz a deklaruje akceptaci všech MIME typů
4. dokument nelze parsovat

4.2.4 Sběr odkazů z XML dokumentů

Pro analýzu XML dokumentů a hledání odkazů v systému Nutch byl naprogramován plugin *parse-xml*. Každý dokument, který je detekován jako XML (MIME typ dokumentu obsahuje „xml“), je analyzován pomocí tohoto pluginu. Plugin využívá možnost akceptovat různé MIME typy pomocí regulárního výrazu (viz 4.2.3)

Pro analýzu XML dokumentů je použit existující parser Xerces [20]. Při volbě typu XML parseru přicházely v úvahu možnosti Document Object Model (DOM) a Simple API for XML (SAX). DOM vyžaduje znalost celého dokumentu, který načítá do paměti, a není tolerantní k chybám. Oproti tomu SAX umožňuje postupné zpracování dostupné části dokumentu a umožňuje tedy vyrovnat se s neúplnými dokumenty. Proto je použit SAX a plugin tím získává základ odolnosti proti chybám.

Při analýze jsou v dokumentech vyhledávány XML *namespaces*, na základě kterých se rozlišuje, v jakém formátu se zde mohou vyskytovat odkazy na další dokumenty. Pro různé typy dokumentů se používají specifické vyhledávací strategie.

Plugin vyhledává v dokumentech z rodiny XML následující typy odkazů:

- XML document – *external DTD, schemaLocation, noNamespaceSchemaLocation*
- XML Schema – *import, include, override*
- XSL Transformation – *import, include*
- XML Style Sheet – Processing instruction XML (*href* odkazy)
- RelaxNG – *include, externalRef*

Strategie se mohou kombinovat, tzn. že techniky pro obecný XML dokument se použijí i v případě schémat nebo transformačních programů.

Při hledání odkazů v dokumentu se porovnávají jména elementů a jejich *namespace* s očekávanými názvy dle specifikace nebo doporučení. V případě shody je podle definice extrahován odkaz, který je uložen do seznamu odkazů dokumentu. Na závěr je tento seznam vložen do výsledku parsování dokumentu.

Atributy *schemaLocation* a *noNamespaceSchemaLocation* se mohou vyskytovat dle specifikace [23] u libovolného elementu, proto jsou po detekci namespace „`http://www.w3.org/2001/XMLSchema-instance`“ prohledávány všechny elementy v dokumentu. Tento mechanismus pokrývá i odkazy, které mohou být v dokumentech XML Schema a XSL Transformation.

Odkazy v RelaxNG dokumentech se mohou vyskytovat pouze ve vybraných elementech a ty se hledají podle jména.

Protože velké množství dokumentů na Internetu je nevalidních či dokonce špatně formovaných, je potřeba, aby tím způsobená chyba měla minimální dopad na přínos celé analýzy dokumentu. Pokud tedy parser nalezl odkazy na další dokumenty, které může předat k dalšímu zpracování, tak případnou výjimku pouze vytiskne, zastaví další zpracování dokumentu a vrátí doposud nalezené odkazy.

4.2.5 Skóre

Pro alternativní výpočet skóre bylo implementováno rozhraní *ScoringFilter*. Pravidla upravující skóre jsou nastavena níže uvedeným způsobem.

Adresám, které přibývají do databáze, je přiřazeno výchozí skóre 0. Pro ručně vložené adresy (inject) je toto skóre zvýšeno na hodnotu 1, aby byly staženy přednostně. Pokud vkládaná adresa končí známou příponou XML dokumentu (xml, dtd, xsl, xslt, xsd, xq, rdf, owl, rng), zvyšuje se přiřazené skóre o 0,5.

Skóre je dále zvýšeno i u adres odkazů, které byly nalezeny v XML dokumentech, a to o 0,5.

Při generování jsou vybírány adresy s nejvyšším skóre, které ještě nebyly staženy. To při uvedeném hodnocení vybírá potenciální XML data, protože jejich skóre je vyšší než u ostatních obyčejných adres nalezených v HTML dokumentech.

Dokument, který má známou příponu, je stahován přednostně, ale nemusí patřit do rodiny XML, protože přípona může být uvedena chybně. Skóre takto stažených dokumentů je sníženo o 0,5. Tím se zruší zvýšení skóre, ke kterému došlo při vložení dokumentu do databáze.

Alternativně bylo možné zvyšovat skóre při generování adres a tím předejít permanentnímu zvýšení skóre u dokumentů, které po stažení nejsou XML. To by ale zvýšilo výpočetní náročnost fáze generování, která navíc probíhá opakovaně a týká se výrazně vyššího počtu adres, než kolik jich je staženo. Proto je použito řešení popsané výše.

4.3 Analyzer

První verze Analyzeru byla vydána k úspěšné obhajobě projektu v červnu 2009. Od té doby se podařilo autorovi této práce a části původního týmu nalézt a opravit řadu chyb, zlepšit modularitu aplikace a připravit tak prostor pro další vylepšování.

V této kapitole jsou popsané funkce Analyzeru vytvořené pro potřebu této diplomové práce.

4.3.1 Import dokumentů

Analyzer původně umožňoval import dat pouze ze souborového systému. Toto omezení bylo později zobecněno na jakýkoliv registrovaný objekt typu *Storage*[1].

[1] Úložiště dokumentů v aplikaci Analyzer, které umožňuje podle identifikátoru náhodný přístup k obsahu dokumentů.

Pro potřebu této práce bylo přidáno další rozhraní pro import dat. Nově lze vložit i dokumenty, ke kterým není náhodný přístup, například načítat data ze sítě.

Modul *Nutch import* pro Analyzer vznikl, aby umožnil importovat data ze stažených segmentů. Data jsou předána v proudu a Analyzer si je znovu sám ukládá.

Je možné zrealizovat i alternativní zpřístupnění stažených dat. Celý segment by byl využit jako zdroj dat a k dokumentům v něm by se přistupovalo přímo pomocí indexu. Díky tomu by bylo možné odstranit jedno kopírování.

4.3.2 Export kolekcí

Analyzer původně neumožňoval export vytvořených kolekcí dokumentů. Tato funkcionalita byla rovněž doplněna až v rámci této práce.

Každý dokument v exportované kolekci je uložen do vybraného adresáře a informace o jeho adrese a novém názvu souboru je přidána do souboru s metadaty (pojmenovaný „_INDEX_METADATA“) ve stejném adresáři. Jedná se o exportované Java *Properties*, kde klíčem je název exportovaného souboru a hodnotou je adresa, ze které byl dokument stažen. Jednotlivé záznamy jsou uloženy v řádcích a mají formát:

```
<název souboru>=<adresa URL>
```

Řešení by bylo možné upravit i tak, aby byly dokumenty exportovány do společného balíku, se kterým by se snáze manipulovalo. Bylo by také možno exportovat místo dokumentů pouze jejich adresy. Tím by vznikl seznam vhodný pro následné opakované stažení.

4.3.3 Zobrazení domén a adres

Původní Analyzer nepracoval efektivně s doménami a adresami. Prakticky se jednalo o implementaci spoléhající na vyhledávání průchodem pole. Analyzer byl testován pouze na omezeném vzorku dat z několika málo domén, proto tento nedostatek nebyl při vývoji objeven.

Problém vyvstal při importu cca 5 tisíc dokumentů posbíraných pomocí modifikované verze Nutch, která vznikla v rámci této práce. V průběhu tohoto pokusu Analyzer přestal reagovat.

Proto byl systém evidence upraven na implementaci pomocí hashovací tabulky. Díky provedené změně Analyzer nyní tento objem zvládne. Testováno bylo vložení a zpracování cca 60 tisíc dokumentů, které proběhlo bez jakéhokoliv problému s odezvou aplikace.

Otevřeným problémem nadále zůstává zobrazení domén v aplikaci, neboť jejich velké množství, v případě sběru dat z Internetu, spolu s jejich řazením je pro uživatele velmi nepřehledné. To však není předmětem této diplomové práce.

4.3.4 Aktualizace změn

Původní Analyzer při každé změně zobrazené položky generoval požadavek na překreslení. Při náročných operacích nebyly tyto požadavky zpracovány hned a v aplikaci se hromadily, což vedlo k jejímu „zamrzání“.

Implementací systému pozdržených aktualizací, kdy se opakované požadavky na překreslení téhož prvku seskupují, se zredukovalo množství událostí, které systém musí zpracovat, a tím se zlepšila reakční doba aplikace. Díky obecnému návrhu bylo následně možné odložené aktualizace využít i pro další typy objektů.

5. Testy a srovnání s jinými řešeními

5.1 Srovnání systémů

V rámci této práce se podařilo implementovat systém, který je schopen rozpoznat různé druhy XML dokumentů. Na základě heuristik preferuje tyto dokumenty při stahování a je schopen vyhledat v nich odkazy na další dokumenty z rodiny XML. Díky využití znalosti formátu odkazů, je schopen detekovat i odkazy relativní. Tato detekce odkazů je dostupná i pro dokumenty, které nejsou staženy celé.

Oproti tomu systém Egothor ani originální systém Nutch neumí detekovat žádné odkazy z XML dokumentů. Ačkoliv oba tyto systémy dokážou objevit odkazy na některé XML dokumenty při analýze stažených HTML stránek, tak není jejich stažení nijak prioritizováno.

Xyleme se liší od dříve uvedených systémů detekcí odkazů na DTD dokumenty. Podrobnosti o způsobu zpracování ale nejsou dostupné. Tento systém přikládá velkou váhu aktuálnosti dokumentů. Naproti tomu řešení v této diplomové práci se snaží získat hlavně co nejširší vzorek dat.

Zavedením filtrace ukládaných dokumentů se podařilo snížit objem uložených dokumentů o 98% a tím ušetřit místo na pevném disku. Filtraci HTML dokumentů používá i projekt Xyleme, který ukládá pouze XML a DTD dokumenty. Systém sběru dat z této práce vyřazuje dokumenty, které jsou rozpoznány jako nevyhovující. Takto je kromě XML dat ukládán ještě další obsah, který by mohl být pro analýzu zajímavý. Ten tvoří cca 32% uložených dat.

V rámci této práce nebyla řešena indexace obsahu stažených dokumentů. Tuto funkci nabízí pro XML pouze Xyleme.

Tím, že Xyleme poskytuje operace nad staženými daty, vyžaduje dobře formované (*well formed)* dokumenty. Pro analýzu reálných dat je však vyloučení neplatných dat zkreslující. Neplatná data je možné zkusit opravit nebo na nich samotné techniky oprav zkoumat.

5.2 Testování

Testování probíhalo nejčastěji na vybraném stroji v počítačové laboratoři MFF UK na Malé Straně. Protože sběr dat probíhal v akademických prostorách, bylo třeba zvlášť obezřetně volit zátěž serverů. Ta byla omezena na maximálně 500 adres pro jeden server při každém cyklu stahování. Sběr dat nebyl testován na maximální rychlost sběru, ale na obecnou schopnost vyhledávání a stahování dokumentů z rodiny XML.

Testování probíhalo průběžně s vývojem aplikace. Postupně rostl objem stahovaných dat a podařilo se takto získat velké množství dokumentů, které lze charakterizovat takto: celkem bylo stahováno 716 tisíc dokumentů, z toho se jich nepodařilo stáhnout 69 tisíc. Staženo bylo cca 20 GB, z toho uloženo pouze 800 MB. Dokumenty byly stahovány v 16 samostatných cyklech obsahujících generování adres, stahování dokumentů a aktualizaci databáze. Po posledním

cyklu obsahovala databáze 6 milionů adres. Ve stažených datech bylo identifikováno 48 262 XML dokumentů.

Výsledná verze systému pro stahování byla použita pro následující test. Použit byl náhodný výběr volně dostupných adres podle návodu [27] čítající cca 800 adres, kterými byla v úvodu naplněna databáze adres (seed). Sběr probíhal v průběhu 3 dnů v cyklu 19 stahování po maximálně 30 tis. adresách.

Celkem bylo při testu stahováno téměř 511 tis. dokumentů, z nichž jen necelých 438 tis. bylo staženo úspěšně. Ze stažených dokumentů tvoří XML data 12 tis., tedy 3,62% [1].

Úvodní vygenerované adresy a skript zajišťující sběr dat jsou také k dispozici na přiloženém DVD. Součástí jsou i statistiky vygenerované z tohoto sběru.

Systém na rozpoznávání odkazů v XML dokumentech byl navržen obecně a se širokou paletou rozpoznávaných forem odkazů. V rámci testů nebyly některé typy dokumentů na Internetu nalezeny, proto ani příslušné formy odkazů nebyly ověřeny. Sem patří např. odkazy v RelaxNG dokumentech, které se nepodařilo ve vzorku objevit. V minimálním počtu byly nalezeny odkazy na schémata XML pomocí zpracování atributu *schemaLocation* a *noNamespaceSchemaLocation*, neboť velká většina XML dokumentů tento atribut postrádá.

Mezi uloženými dokumenty stále zabírá jistou část XHTML. Je proto nutné při zpracování dat rozhodnout, zda i tato data mají být využita a případně zavést jejich filtraci.

[1] Při pokusu o opakování testu se jeho výsledky nemusí shodovat vzhledem k dynamice Internetu zmíněné v kapitole 2.1.

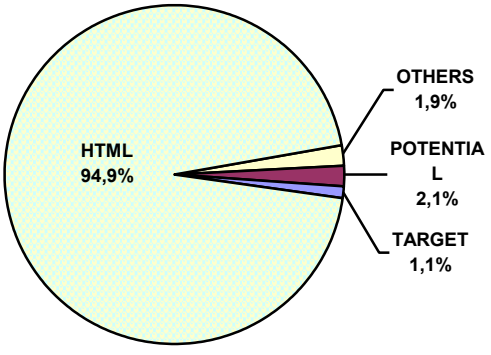
6. Závěr

Na základě analýzy tato práce rozšiřuje vybraný webový crawler Apache Nutch. Práce ukazuje hlavní nedostatky, které bránily v jeho použití pro sběr XML dat a navrhuje jejich možné řešení. Vybrané hlavní nedostatky pak implementace upravuje a přetváří tak původní systém v nástroj, který je možné využít pro cílený a vcelku efektivní sběr XML dat. Díky propojení s aplikací Analyzer vzniká systém, který je schopný zajistit data pro statistické zpracování, analyzovat je a přehledně prezentovat výsledky.

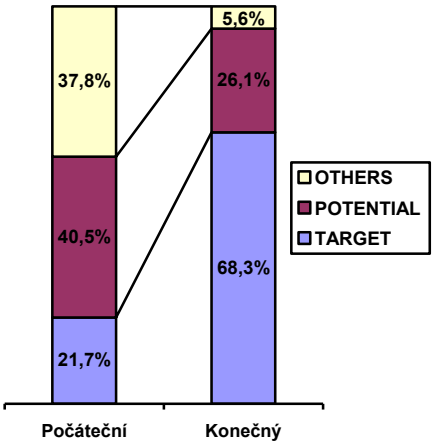
Do budoucna se počítá s využitím řešení pro analýzu reálných dat v prostředí aplikace Analyzer (například pro rozšíření existující analýzy reálných XML dat [1]). Pro jiné využití dat je možné data exportovat. Mezi další využití, která se nabízejí, patří například testování úložišť XML dokumentů nebo vyhledávání chybných XML dokumentů a testování algoritmů pro jejich opravu a zpracování.

Funkčnost sběru XML dokumentů a chování aplikace při relativně velkém objemu dat (cca 20 GB) byly ověřeny provedením několika testovacích sběrů dat. Rozsah testů byl limitován, protože testování probíhalo ve školní laboratoři a bylo třeba omezit intenzitu odchozích požadavků, aby se předešlo přetížení sítě.

Výsledky prezentují následující obrázky. Obr. 6 ukazuje rozložení typů stažených dokumentů, které poskytoval zvolený crawler na začátku. Kategorie TARGET obsahuje dokumenty z rodiny XML, kategorie POTENTIAL obsahuje textové dokumenty, které mohou být v aplikaci Analyzer správně detekovány (dokumenty typu text/plain), a poslední kategorie OTHERS obsahuje dokumenty, u kterých lze vyloučit souvislost s XML dokumenty (sem patří multimédia, binární formáty a zbytek formátů). Do kategorie HTML spadalo 95% stažených dokumentů (98% staženého objemu). Analýzou zbývajících 5% dokumentů, získáme rozložení, které ukazuje počáteční stav na Obr. 7. Zjištěné rozložení dokumentů včetně jejich klasifikace je k dispozici na přiloženém DVD ve formě tabulky MS Excel.



Obr. 6 – Původní rozložení dokumentů podle typu ve staženém vzorku



Obr. 7 – Změna složení staženého obsahu díky této práci

Díky implementaci kroků uvedených v této práci se podařilo zlepšit poměr sebraných dat, jak ukazuje konečný stav na témže obrázku. Vedle toho, že

konečné řešení není zatíženo nadměrným obsahem HTML, se podařilo zvýšit účinnost sběru XML dokumentů a zároveň snížit podíl špatného obsahu v ukládaných datech.

Na základě testování se podařilo identifikovat řadu námětů pro další vývoj. Značná část byla implementována a projevila se zlepšením vlastností crawleru. Některé z nich však nebylo možné realizovat, protože svým zaměřením a rozsahem byly nad rámec této práce. Zůstaly proto jako podněty pro další rozvoj. Jejich seznam je uveden níže.

Osobním přínosem autora práce je, že úspěšně upravil výchozí nástroje, které si na základě analýzy vybral na začátku práce, a propojil dvě aplikace, které se vzájemně doplňují a umožňují opakovaně analyzovat reálná data na Internetu. Autor dále ověřil využitelnost aplikace Analyzer pro práci se staženými dokumenty, a navrhl řešení identifikovaných problémů a podstatnou část jich též implementoval, čímž se zvýšily užitné vlastnosti výsledného systému.

Možná vylepšení

Jako nejdůležitější se jeví omezení nadměrného růstu databáze adres, které lze realizovat pomocí omezení počtu adres v systému pro domény a subdomény, jak je popsáno na konci kapitoly 2.2.4.

Dále by bylo vhodné zavést detekci a stahování XML dat v komprimovaných archivech. V současné chvíli je stažení všech archivů blokováno filtrováním. Podmíněné stahování odkazovaných archivů na základě analýzy stránky (vyhledávání klíčových slov) by umožnilo stažení některých komprimovaných XML dokumentů a přitom stále filtrovalo ostatní archivy.

Další rozšíření či úpravy nejsou již tak naléhavé. Patří mezi ně například implementace heuristického dohledávání schémat a dokumentů, které nejsou dostupné na očekávané adrese, do aplikace Analyzeru nebo rozšíření vyhledávání odkazů o další formáty XML dokumentů.

Literatura

[1] Mlýnková I., Toman K., Pokorný J. (2006): Statistical Analysis of Real XML Data Collections, MFF UK

[2] Koster M. (2010): Robots Exclusion Protocol http://www.robotstxt.org

[3] Thelwall M., Stuart D. (2006): Web crawling ethics revisited: Cost, privacy, and denial of service. *J. Am. Soc. Inf. Sci. Technol.* 57, 13, 1771–1779

[4] Xyleme – A dynamic warehouse for XML data of the Web (2010), Xyleme Inc. http://xyleme.com

[5] Ailleret S. (2009): Larbin – Multi-purpose web crawler http://larbin.sourceforge.net/index-eng.html

[6] Schejbal J. a kol. (2009): Analyzer documentation, MFF UK http://urtax.ms.mff.cuni.cz/anaxml/doc.html

[7] Galamboš L. a kol. (2010): Egothor – Java search engine http://www.egothor.org/

[8] Hudák R. a kol. (2009): Docserver – Universal document repository, MFF UK https://gforge.cythres.cz/gf/project/docserver/

[9] Apache Nutch (2010), Apache Software Foundation http://lucene.apache.org/nutch/

[10] Krugler K., Groschupf S. (2009): Bixo – web crawler toolkit introduction http://bixo.101tec.com/wp-content/uploads/2009/05/bixo-intro.pdf

[11] Schejbal J. a kol. (2009): Analyzer, MFF UK http://urtax.ms.mff.cuni.cz/anaxml/

[12] Berners-Lee T. a kol. (2005): Uniform Resource Identifier (URI) Syntax, The Internet Society http://labs.apache.org/webarch/uri/rfc/rfc3986.html

[13] Mignet L., Barbosa D., Veltri P. (2003): The XML web: a first study. In *Proceedings of the 12th international Conference on World Wide Web* (Budapest), ACM, 500–510

[14] Registrované MIME typy (2010), The Internet Corporation for Assigned Names and Numbers http://www.iana.org/assignments/media-types/

[15] Tika parser (2010), Apache Software Foundation http://lucene.apache.org/tika/

[16] Google Terms of Service (2010), Google Inc.
http://www.google.com/accounts/TOS

[17] Cho J., Garcia-Molina H., Page L. (1998): Efficient crawling through URL ordering. In *Proceedings of the 7th international Conference on World Wide Web* (Brisbane, Australia), ACM, 161–172.

[18] Wicks J. R., Greenwald A. (2007): More efficient parallel computation of pagerank. In *Proceedings of the 30th Annual international ACM SIGIR Conference on Research and Development in information Retrieval* (Amsterdam), ACM, 861–862

[19] Abiteboul S., Preda M., Cobena G. (2003): Adaptive on-line page importance computation. In *Proceedings of the 12th international Conference on World Wide Web* (Budapest), ACM, 280–290

[20] Xerces Java parser (2010), Apache Software Foundation
http://xerces.apache.org/xerces2-j/

[21] Carmel J. (2010): XML Datasets
http://www.xmldatasets.net/cgi-bin/dt.pl

[22] Manning Ch. D., Raghavan P., Schütze H. (2008): Introduction to Information Retrieval, Cambridge University Press

[23] Thompson H. S. a kol. (2004): XML Schema Part 1 – Structures Second Edition, World Wide Web Consortium
http://www.w3.org/TR/xmlschema-0/

[24] Clark J. (1999): XSL Transformations, World Wide Web Consortium
http://www.w3.org/TR/xslt

[25] Clark J., Makoto M. (2001): RELAX NG Specification, OASIS
http://relaxng.org/spec-20011203.html

[26] Clark J. (1999): Associating Style Sheets with XML documents, World Wide Web Consortium
http://www.w3.org/TR/xml-stylesheet/

[27] Nutch tutorial (2009), Apache Software Foundation
http://lucene.apache.org/nutch/tutorial8.html

[28] Xyleme L. (2001): Xyleme: A Dynamic Warehouse for XML Data of the Web. In *Proceedings of the international Database Engineering & Applications Symposium*. IEEE Computer Society, 3–7

Dodatky

A.Instalační příručka

Pro instalaci a použití programu je třeba, aby na stroji byly nainstalovány následující nástroje:

```
JDK 6
Ant
```

Pro následné používání aplikace Analyzer je třeba grafické rozhraní systému Windows nebo X server na systému typu Linux.

Umístění souborů

Podrobný popis umístění souborů je na přiloženém DVD rozepsán v souboru README v kořenovém adresáři.

/src/analyzer – zdrojové kódy a knihovny k systému Analyzer
/src/nutch – zdrojové kódy a knihovny modifikované verze Apache Nutch
/live/analyzer – spustitelná verze aplikace Analyzer, která nevyžaduje kompilaci
/live/nutch – skript a konfigurace pro zjednodušený sběr XML dat

Kompilace

Pro kompilaci je nutné zkopírovat adresář se zdrojovými kódy (/src) z DVD do adresáře, kde bude možné zapisovat soubory.

Nutch
Přenesený adresář /src/nutch budeme dále nazývat NUTCH_HOME. V tomto adresáři voláním příkazu „ant“ dojde ke kompilaci crawleru a všech jeho rozšíření. Po úspěšném dokončení je možné spustit sběr dat.

Analyzer

Aplikaci Analyzer není třeba kompilovat. Její spustitelná verze je připravena na přiloženém DVD.

Pokud by bylo potřeba ji kompilovat, je nutné mít navíc nainstalované vývojové prostředí Netbeans IDE s rozšířením pro úpravu Javy. Ve vývojovém prostředí je potřeba otevřít projekt umístěný ve zkopírovaném adresáři /src/analyzer/analyzer a ten nechat zkompilovat (volbou *Build ZIPs*). V podadresáři „dist“ tak vznikne analyzer.zip, který obsahuje archivovanou podobu aplikace.

Konfigurace

Konfigurace pro sběr dat se nachází v adresáři **nutch/conf**, konkrétně v souborech **nutch-default.xml** a **nutch-site.xml**. První ze souborů obsahuje výchozí

nastavení proměnných. Pro samotnou konfiguraci je doporučeno zkopírovat příslušnou část konfiguračního souboru z části default do souboru **nutch-site.xml** a tu následně upravit.

Před spuštěním je nutné nastavit identifikaci crawleru. To lze provést v souboru **nutch-site.xml** nastavením proměnných:

Název	Popis
http.agent.email	e-mailová adresa, na které lze kontaktovat provozovatele crawleru
http.agent.url	adresa, na které jsou podrobnosti o důvodu procházení Internetu

Další část konfigurace ovlivňuje rychlost procházení Internetu:

Název	Popis
fetcher.threads.fetch	počet vláken, které stahují data z Internetu
fetcher.server.delay	minimální prodleva mezi jednotlivými požadavky na jeden server
generate.max.per.host	maximální počet adres z jednoho serveru, které jsou vygenerovány pro jeden segment

Pro využití nové konfigurace je třeba aplikaci znovu zkompilovat („ant“).

B. Uživatelská příručka

V uživatelské příručce jsou popsány operace, potřebné ke zprovoznění sběru dat a způsob jejich vložení do aplikace Analyzer.

NUTCH_HOME bude v následujícím textu označovat adresář, do kterého byl instalován modifikovaný Apache Nutch z této práce.

Spuštění sběru

1. Pro zahájení sběru dat z Internetu je potřeba mít seznam úvodních adres, ze kterých prohledávání začne. Pokud je tento seznam k dispozici, je možné pokročit k dalšímu bodu. Pokud nemáte vhodný zdroj úvodních adres, lze úvodní adresy pro tento účel vygenerovat z volně dostupné databáze [27], kterou je nutné stáhnout a rozbalit. Na operačních systémech *NIX to odpovídá následujícím příkazům:

```
wget http://rdf.dmoz.org/rdf/content.rdf.u8.gz
gunzip content.rdf.u8.gz
```

Nyní je třeba ze souboru **content.rdf.u8** vytvořit náhodný výběr:

```
mkdir inject_dir
$NUTCH_HOME/bin/nutch org.apache.nutch.tools.DmozParser
  content.rdf.u8 -subset 5000 > inject_dir/dmoz
```

2. Vybrané adresy je třeba vložit do databáze následujícím příkazem:

```
$NUTCH_HOME/bin/nutch inject crawldb inject_dir
```

Databáze nyní obsahuje úvodní seznam adres.

3. Spuštění sběru dat

```
$NUTCH_HOME/bin/nutch generate crawldb segments -topN $TOP
```

Takto vznikne nový segment v adresáři **segments**. Označme cestu k němu ONE_SEGMENT. Následující příkaz spustí stahovaní nového segmentu.

```
$NUTCH_HOME/bin/nutch fetch $ONE_SEGMENT –threads $THREADS
```

Nyní jsou adresy v segmentu staženy. V následujícím kroku je třeba aktualizovat databázi podle nově stažených dat.

```
$NUTCH_HOME/bin/nutch updatedb crawldb $ONE_SEGMENT
```

4. Statistika stažených dat (volitelně)
Ze stažených dat lze vygenerovat orientační tabulku, která obsahuje popis stažených dokumentů.

```
$NUTCH_HOME/bin/nutch statmime stat_output segments/2*
```

Pokud je třeba získat dat více, je možné pokračovat znovu od bodu 3, jinak lze začít dalším bodem.

5. Import dat do aplikace Analyzer (volitelně)
Import dat je popsán v uživatelské příručce k aplikaci Analyzer [6]. Rozdílem je, že jako zdroj importu je třeba zvolit „Nutch segment“ a nikoliv „Filesystem storage.“ Dále již aplikace vyžaduje vybrání segmentu, ze kterého se mají data importovat. Zde je třeba zvolit jediný stažený segment (tj. adresář označený v předchozím textu ONE_SEGMENT). V dalším kroku je třeba vybrat, kam mají být dokumenty uloženy (lze využít výchozí možnost). Po potvrzení se zahájí import dat.

Jednoduchý sběr dat

Pro zjednodušený sběr dat lze využít skript na DVD (/live/nutch/download.sh), ke kterému je připravena i sada adres na naplnění databáze (získaná podle 1. kroku v předchozím popisu). Ve skriptu je třeba upravit hodnotu proměnné NUTCH_HOME tak, aby odpovídala adresáři, kam byl Nutch nainstalován.

Modifikací dalších vlastností uvnitř skriptu lze nastavit například počet cyklů stahování nebo maximální počet dokumentů v jednom segmentu.