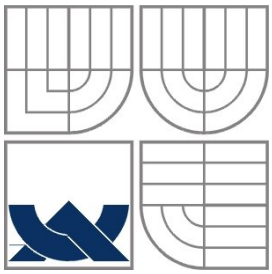
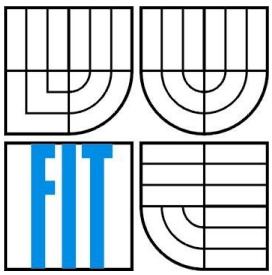


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

GENERÁTOR VĚDECKÝCH WEBOVÝCH PORTÁLŮ

SCIENTIFIC WEB PORTAL GENERATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MILOŠ KUNDRÁT

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2008

Abstrakt

Generátor vědeckých webových portálů. Celistvý projekt, jehož součástí je tato diplomová práce, se skládá z uživatelského prostředí GUI, komunikace procesů, komunikace programu s uživatelem a ze skriptu pro automatické rozpoznávání vyjmenované entity metodou extrakce sémantické informace ze značkovaného textu. Poslední jmenovaná část je hlavní náplní mé diplomové práce. Cílem diplomové práce je skript (prototyp), který na základě vstupního XML souboru se jmény vědeckých pracovníků vrací výstupní XML soubor s URL adresami na jejich domovské stránky a stránky s výpisem publikací. Značná část mé práce bude věnována shromáždění kvalitních testovacích dat a nakonec důkladné statistické analýze výsledného chování skriptu, který jsem naprogramoval. Tato část nám podá ucelený obraz o procentuální úspěšnosti a rychlosti navrženého skriptu. Skript bude připraven pro začlenění do společného projektu.

Klíčová slova

Python, XML, HTML, UrlLib, lXML, lXML eTree a xml.dom.minidom knihovna, Internet, URL, Web, Google.com, Yahoo.com, Yahoo Search API, Google API.

Abstract

Scientific web portal generátor. The entire project, whose part this dissertation is, consists of the users interface GUI, the processes‘ communication, the software‘s communication with a user and of the script for automatic parsering the selected entity by a method of extraction the semantic information from the marked text. The last-named part is a main content of my dissertation. The dissertation‘s objective is a script (prototype), which returns on the basis of an input XML file with names of research workers, an output XML file with URL home pages‘ addresses and pages‘addresses with a list of publications. Great deal of my dissertation will be devoted to the first-rate test datas assembly and in the end to the profound static analysis of the resulting script‘s behaviour, which I have programed. This part will survey us about the percentage fruitfulness and about the rate of the designed script. The script will be prepared for the integration into the common project.

Keywords

Python, XML, HTML, UrlLib, lXML, lXML eTree a xml.dom.minidom Library, Internet, URL, Web, Google.com, Yahoo.com, Yahoo Search API, Google API.

Citace

Kundrát Miloš: Generátor vědeckých webových portálů. Brno, 2008, diplomová práce, FIT VUT v Brně.

Generátor vědeckých webových portálů

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením doc. RNDr. Pavla Smrže, Ph.D.
Další informace mi poskytl Marek Schmidt.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....

Jméno Příjmení

V Brně dne 19. května 2008

Poděkování

Tímto bych chtěl poděkovat panu doc. RNDr. Pavlu Smržovi, Ph.D. za jeho odbornou pomoc, kterou mi při řešení práce ochotně a vždy poskytl, a zároveň za nemalou odbornou pomoc panu Marku Schmidtovi.

Obsah

Úvod

Každý z nás, kdo používá internet, se každodenně potkává s webovým rozhraním, které mu usnadňuje práci, když potřebuje vědět něco, co nezná – vyhledává informace. Pravděpodobnost nálezu námi požadované informace přes specializované webové portály je v dnešní době již velmi vysoká. Avšak v okamžiku, kdy požadujeme větší množství informací, narážíme na úhlavního nepřítele člověka – čas. Představme si, že chceme vyhledat k více než stovce měst jejich nejvýznamnější hotely. Určitě si dokážete spočítat množství času, které bychom strávili, kdybychom jednotlivé hotely vyhledávali zadáváním názvů měst na stránkách vyhledávačů[1] či portálů[2].

My stojíme před velmi podobným problémem, a to vyhledáním URL[3] adres domovských stránek akademických nebo též vědeckých pracovníků a posléze na těchto domovských stránkách URL adresy odkazující na stránku obsahující seznam jimi vydaných publikací. V situaci, kdybychom chtěli tuto operaci provést ručně (přímo přes webový portál vyhledávače), zabrala by nám tato procedura řádově hodiny času. Naším cílem je navrhnout skript, který by tuto práci odvedl za nás. Pokusím se nalézt takové řešení problému, které bude časově nejméně náročné a zároveň bude splňovat nejvyšší pravděpodobnostní hodnoty úspěšnosti nálezu požadované stránky, spolehlivosti a funkčnosti.

Řešení diplomové práce je možné rozdělit na **4 nejdůležitější etapy vývoje**:

- **vytvoření testovacích dat**
  (ruční získání URL adres domovských stránek a URL adres stránek obsahující publikace vědeckých pracovníku pomocí vyhledávače Google.com)
- **vývoj testovacího skriptu**
  (získání statistických dat pro závěrečnou analýzu dosažených výsledků)
- **implementace samostatného skriptu pro budoucí použití**
  (návrh skriptu automaticky rozpoznávající vyjmenované entity metodou extrakce sémantické informace ze značkovaného textu s využitím rozhraní Yahoo API a posléze Google API)
- **analýza získaných výsledku a zamyšlení se nad možným zdokonalením**

[1] Vyhledávačem je označována stránka, která nabízí službu k vyhledávání zadaných výrazů skrze internet pomocí vyhledávajícího skriptu. Mezi nejznámější vyhledávače patří Google.com, Yahoo.com a další.

[2] Portál je webová stránka, která využívá k vyhledávání služby spřátelených vyhledávačů. Mezi portály na české scéně patří Seznam.cz, Centrum.cz a další.

[3] URL (Unique Resource Locator) je řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu. (Uniform Resource Locators (URL). [dostupné 29. 12. 2007]. [HTML dokument]. [http://tools.ietf.org/html/rfc1738].)

(statistická analýza získaných výsledků a vyhodnocení funkčnosti a rychlosti navrženého skriptu)

Ze závěrů, které učiníme na základě vyhodnocení statistických výsledků, dále budeme přemýšlet nad možným vylepšením a hledat příčiny daných výsledků.

Na projektu samotném bylo pracováno již v rámci semestrálního projektu, kdy byla vytvořena testovací sada domovských stránek autorů, která se postupem času stala neaktuální a byla vytvořena nová. Zároveň byl vytvořen jednoduchý prototyp testovací aplikace, který byl v této diplomové práci zdokonalen a důkladně analyzován z hlediska funkčnosti a úspěšnosti. Na základě této analýzy vychází konečná verze skriptu, která splňuje požadavky pro budoucí použití ve spojení z prací mých kolegů.

Kapitola 1

Technologie a webová rozhraní použitá při vývoji skriptu aplikace

1.1 Python

Python je moderní programovací jazyk vyvinutý Guido van Rssumem. Používají jej stovky tisíc programátorů, a jejichž počet se každým rokem zdvojnásobuje.

Python přitahuje nové uživatele svou širokou paletou možností. Umožňuje vytvářet aplikace mnohem rychleji než při programování v tradičních jazycích, jako jsou C, C++ nebo Java. Jedná se o na platformě nezávislý jazyk, který běží stejně na Windows, unixových operačních systémech, OS/2 i na systémech počítačů Macintosh. Lze jej používat pro psaní malých aplikací nebo skriptů stejně jako pro vývoj velkých softwarových projektů Poskytuje přístup k velmi výkonnému a uživatelsky jednoduchému grafickému uživatelskému rozhraní. Navíc je jeho vývoj otevřený a je zdarma.

Python je moderní jazyk vyšší úrovně s mnoha vlastnostmi:

- Automatická správa paměti.
- Dynamické psaní.
- Jednoduchá konzistentní syntaxe a sémantika.
- Použitelný pro všechny hlavní platformy (Windows, Unix, Apple apod.)
- Vysoce modulární.
- Vhodný pro psaní skriptů i vývoj rozsáhlých projektů.
- Dostatečně rychlý a snadno rozšiřitelný o moduly C/C++ pro části vyžadující vysokou rychlost výpočtů.
- Snadný přístup k vývojovým nástrojům pro tvorbu grafických uživatelských rozhraní.
- Zabudované mechanismy a funkce pro trvalé (persistentní) ukládání objektů, vyspělá asociativní pole, rozšiřitelná syntaxe tříd, univerzální srovnávací funkce a tak dále.
- Výkonné knihovny pro numerické výpočty, manipulace s obrázky a uživatelská rozhraní, webové skripty a další.
- Podpora široké a aktivní komunity uživatelů.
- Lze ho integrovat s řadou jiných programovacích jazyků, což umožňuje využít síly obou a vyhnout se tak jejich slabinám.

Zdroj: *Harms, D., MacDonald, K.: Začínáme programovat v jazyce Python. 1, Brno. Computer Press 2006, s. 3, 14.*

Syntaxe a jeho dynamické typy dohromady s interpretováním kódu dělají jazyk výborným nástrojem pro rychlý vývoj aplikací (Rapid Application Development, RAD). Zdrojové kódy interpretu a standardních knihoven jsou volně ke stažení z domovské stránky Pythonu (http://www.python.org/) a je možné je dále volně modifikovat a distribuovat.

1.2 XML a HTML

XML – Extensible Markup Language. Výskyt zkratky je stále častější. Skrývá se pod ní technologie určená k organizaci údajů v textových zápisech, kterým mohou porozumět nejen lidé, ale také stroje. Za krátkou dobu existence W3C[4] pro XML se tato technologie šíří rychlostí virů. Zasáhla nejen internetové prohlížeče, databázové stroje, ale také programovací a skriptovací jazyky.

XML je podmnožinou SGML[5]. Jde o jakousi zjednodušenou verzi, která obsahuje z SGML to nejdůležitější. Můžeme si tedy definovat vlastní značky (*tagy*) a pak pomocí nich vytvářet dokumenty a zpřístupňovat je ostatním pomocí webu.

Jazyk HTML[6], který se již delší dobu ve velkém měřítku používá pro tvorbu www stránek má rysy, které brání využití efektivnějších metod prohledávání. V HTML se jednotlivé části textu označují značkami, které textu přiřazují většinou čistě prezentační význam. Text lze označit jako nadpis, buňku tabulky nebo úsek, který bude zobrazen zvýrazněně. Tím, že XML umožňuje definovat vlastní sadu značek, můžeme těmito značkami mnohem výstižněji označit jednotlivé části textu. Text lze tedy označit např. jako název básně, jméno autora, kurz akcie, rodné číslo atd. -- záleží pouze na tom, jaké značky definujeme v DTD[7]. Takto označkovaný a strukturovaný text má mnohem vyšší informační hodnotu než stránka zapsaná v HTML.

---

[4] W3C = World Wide Web Consorcium; mezinárodní konsorcium jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Cílem konsorcia je „Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic které zajistí dlouhodobý růst Webu“.
(W3C, World Wide Web Consorcium). [dostupné 28. 12. 2007, překlad]. [HTML dokument]. [http:// www.w3.org/].)

[5] SGML je standardní jazyk určený k formálnímu popisu struktury dokumentů. Je definován na základě normy ISO 8879 vydané roku 1986.

[6] HTML neboli Hypertext Markup Language vzniká v roce 1991 a řadí se dnes k nejrozšířenějším značkovým jazykům. Internetový prohlížeč má za úlohu vyhodnotit a zpracovat text v závorkách (= značkách). Text mimo tyto závorky je zobrazen na obrazovce.

[7] DTD (Document Type Definition, česky Definice typu dokumentu) je jazyk pro popis struktury XML případně SGML dokumentu. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy. DTD tak například vymezuje jazyky HTML a XHTML.

1.2.1 lXML a lXML.eTree knihovna

Knihovna lXML slouží v jazyce Python k vázání knihoven libxml2 a libxslt. Její jedinečnost spočívá v tom, že kombinuje rychlost a uvádí úplnost těchto knihoven s jednoduchostí nativního Python API. Knihovna lXML.eTree dále nabízí jednoduchou práci se strukturovaným textem, resp. zajišťuje výstup strukturovaného textu.

1.2.2 UrlLib a XML.dom.minidom knihovna

Knihovna *UrlLib* nám umožňuje komunikovat se servery (posílat a přijímat data). Já konkrétně využívám nejvíce z této knihovny funkci při komunikaci s Yahoo Search API.

Knihovna *XML.dom.minidom* je užitečná pro práci se soubory ve formátu XML: Obsahuje funkce, s kterými lze dokument XML procházet, editovat atd.

Kapitola 2

Manuální shromáždění URL adres vědeckých pracovníků pro statistickou analýzu

2.1 Domovské stránky (homepage)

Pro účely důkladné analýzy vytvořeného skriptu a k myšlence plně funkčního skriptu a plnohodnotných závěrů nad nejdůležitějšími vlastnostmi (procentuální úspěšností, časovou náročností atd.), jsem shromáždil seznam osob a jejich URL adresy domovských stránek. Ke zkoumání jsem vybral skupinu vědeckých pracovníků – konkrétně skupinu akademických pracovníků sdružujících se ve skupině zabývající se zpracováním přirozeného jazyka – People in NLP[8] :

http://nats-www.informatik.uni-hamburg.de/~siemonse/whoiswho-alphabet.en.html

Na stránkách této vědecké skupiny jsem nalezl seznam členů včetně odkazu na jejich domovské stránky. Při ověřování existence daných odkazů jsem bohužel zjistil, že seznam domovských stránek je téměř nepoužitelný, poněvadž informace jsou staršího data. Většina odkazu již byla nefunkčních, příp. akademický pracovník, který měl domovskou stránku na univerzitě, kde pracoval, po opuštění této pozice již na této univerzitě většinou svou domovskou stránku zrušil, anebo mu byla zrušena automaticky. Využil jsem tedy pouze jmenného seznamu, který čítal něco málo přes 200 jmen, a vyhledal jsem URL adresy domovských stránek ručně přes vyhledávač Google.com (právě při této činnosti jsem si vyzkoušel časovou náročnost ručního vyhledávání). Některé domovské stránky jsem nedokázal nalézt vůbec. Konečný seznam, který se mi podařilo ověřit, obsahoval 200 funkčních URL adres na domovskou stránku, a byl použit pro vývoj skriptu pro semestrální projekt.

Při vývoji a namátkové analýze výsledků skriptu, který je cílem této diplomové práce, jsem postupně docházel k závěru, že se statistika po delší době zásadně mění a některé stránky, které byly vyhodnoceny jako dostupné, se staly nedostupnými. Tato změna v časovém rozmezí listopad 2007 – duben 2008 byla natolik znatelná, že nebylo možné tato data nadále pro účely testování skriptu využít, a proto byla shromážděna nová testovací data, a to z následujících webových stránek:

http://aclweb.org/aclwiki/index.php?title=People

Pro statistické účely jsem opět ručně ze zmiňovaných, nových stránek s aktuálními údaji shromáždil nové domovské stránky k potenciálně vyhledávaným vědeckým pracovníkům (dále jen „autor“) a zároveň shromáždil URL adresy stránek s výpisy publikací. Pro snadné statistické výpočty

[8] NLP = Neuro-Linguistic Programming; Název pochází z disciplín, které ovlivnily počátky rozvoje tohoto vědeckého odvětví. Prvopočátky začaly bádáním nad vztahy mezi neurologií a lingvistikou, a pozorováním zákonitostmi jejich chování. (NLP What Does the Name Mean?. [dostupné 26. 12. 2007]. [HTML dokument]. [http://www.nlpschedule.com/w_neuro_linguistic_programming_definition.html].)

jsem shromáždil 100 plně funkčních a dostupných URL adres domovských stránek a ke každé z nich jednu URL adresa s odkazem na stránku s publikacemi. Dohromady tedy testovací data obsahuji 100 URL adres domovských stránek a 100 URL adres s výpisem publikaci.

2.2 Stránky sdružující publikace

Pro analýzu skriptu, jenž byl součástí semestrálním projektu, jsem dále shromáždil seznam URL adres odkazujících na stránky těchto 200 autorů, které sdružují publikace k příslušnému autorovi. Tento postup byl totožný s postupem předchozím, avšak již neexistuje žádný takový seznam, který by sdružoval zmiňované odkazy na publikace, tudíž jsem musel ručně projít všechny domovské stránky a odkaz manuálně vyhledat. Z předchozí zkušenosti je více než zřejmé, že se seznam testovacích dat opět zúží, poněvadž stránky s publikacemi některých autorů nemusí existovat, anebo se mi je nepodaří nalézt.

Pro potřeby ověření funkčnosti skriptu k semestrálnímu projektu byla vytvořena pouze testovací sada obsahující již zmiňovaných 200 URL adres domovských stránek autorů. Avšak jak bylo řečeno výše, tato testovací data se stala neaktuální a byla vytvořena nová, čítající 100 URL adres domovských stránek. Zároveň s těchto nových URL adres byly získány URL adresy na stránky obsahující výpis publikací, které daný autor publikoval, příp. se k nim hlásí. URL adresy na stránky s publikacemi pro účely testování funkčnosti a statistickou analýzu skriptu byly opět získávány ručně, což mi opět ukázalo, jak časově náročné je „separování“ nějaké informace s textu ručně.

Konečná podoba testovacích dat pro skript čítá 100 URL adres na stránky s výpisem publikací. Je zřejmé, že na domovské stránce se mohou vyskytovat odkazy na publikace dva a i více. Já jsem však do testovacích dat zařadil pouze jednu URL adresu – subjektivně nejdůležitější. Skript, který však bude naprogramován, získá ze stránky všechny adresy, které by se mohly nějakým způsobem týkat publikací autora a potenciálně na ně odkazovat. Výsledkem úspěšnosti bude, zda se URL adresa uložená v testovacích datech shoduje s některou z adres získaných z domovské stránky autora.

Kapitola 3

Vývoj vyhledávacího skriptu

3.1 Vyhledávací API rozhraní

Řada vyhledávačů nabízí možnost využití jejich vyhledávací skriptu, konkrétně „*cestu*“ a přístupová práva k vyhledávacímu stroji (Search API[9]) pracujícího nad jejich databází shromážděných odkazů pro „*soukromé*“ účely. Mezi vyhledávače, které po registraci zájemce o skript zdrojové kódy k přístupu k vyhledávacímu skriptu poskytnou, patří Google.com, Yahoo.com, MSN.com a další. Skripty se od sebe liší jak svou rychlostí, tak svoji technologií, jakou nad Internetem vyhledávají. Proto můžeme očekávat u vyhledávačů rozdílné výsledky v úspěšnosti nálezu hledaného výrazu. V historii tyto rozdíly byly propastné. Dnes se rozdíly s novými technologiemi a principy vyhledávaní stírají. I přesto otestuji vyhledávací schopnosti dvou vyhledávačů – Yahoo.com a Google.com. K použití vyhledávacích nástrojů od těchto společností slouží aplikační rozhraní, konkrétně Yahoo API a Google API. Ne všechny vyhledávače však svůj skript zpřístupňují. Na scéně českých vyhledávačů snad doposud neexistuje[10] žádný takový, který by svou metodu vyhledávaní (vyhledávací stroj) propůjčil pro účely „veřejnosti“.

„*Za zmínku stojí případ spolupráce Seznam.cz a Atlas.cz, kdy Seznam.cz dodává zákazníkům využívající vyhledávání na stránkách Atlas.cz vlastní vyvinutý software. Do konce roku 2007 nabídne Seznam k volnému použití jednoduchý vyhledávací algoritmus, který si bude moci každý vložit na své internetové stránky.*“

Zdroj: DigiWeb. Ekonom.Ihned.cz. *Soumrak nad giganty.* [dostupné 29. 12. 2007]. [HTML dokument]. 09. 08. 2007. [http://digiweb.ihned.cz/c6-10134700-21775700-i00000_d-soumrak-nad-giganty].

Jak bylo řečeno výše, pro začátek jsem se rozhodl využít služeb, resp. vyhledávacího stroje, společnosti Yahoo.com, a tudíž jsem vybral k implementaci a testování procentuálního úspěchu a časové náročnosti do mého skriptu Yahoo Search API rozhraní (zkr. Yahoo API). Pro získání přístupu ke zdrojovým kódům a možnosti využívat služeb vyhledávacího stroje, byla nutná registrace, která mi zajistila získání aplikačního kódu (*appied = Application ID*). Začlenění cesty k Yahoo Search API do mého skriptu naleznete v příloze 1: Výňatek ze zdrojového kódu „zaslání řetězce na vyhledávající webový portal“. Již v první fázi testování funkčnosti skriptu jsem narazil na omezení, a to v podobě omezeného počtu dotazů odeslaných přes toto API rozhraní. Tato skutečnost mi práce velmi stěžovala, poněvadž v případě blokace bylo třeba více jak 60 minut čekat, než bude funkčnost rozhraní pro

[9] API (Application Programming Interface) = rozhraní pro programování aplikací.

[10] K 28.12.2007.

získaný aplikační ID (klíč) v původním stavu. V konečné fázi vývoje tento problém byl velmi znát, protože provádění samotného skriptu trvalo cca 30-40 min, a když se do toho projevila ještě blokace přístupu, pak za celý den práce jsem byl schopný otestovat skript cca 8x, což je pro programátora při ladění velmi svazující a časově náročné.

Ve skriptu jsem za účelem srovnání rychlosti a procentuální úspěšnosti hledání naimplementoval též skript využívající vyhledávací stroj společnosti Google.com. K použití Google API rozhraní potřebujeme opět přístupový klíč, který již není od 5. prosince roku 2006 poskytován. Bohužel se mi nepodařilo získat ke Google API přístup, a proto jsem byl nucen využít vyhledávací stroj Google.com „napřímo“[11]. Tato technika je časově velmi náročná[12], poněvadž návratovou hodnotou je kompletní zdrojový kód stránky obsahující nejen URL adresy odkazující na stránky týkajících se hledaného řetězce, ale i titulky stránek, popisky a další informace, které pro nás nejsou důležité. Sémantickou extrakcí jsem jednotlivé URL adresy ze zdrojového kódu vyextrahoval.

3.2 Načtení vstupních dat – jmen autorů a jejich URL adres a URL adres s publikacemi

Seznam autorů jsem původně v semestrálním projektu umístil do textové/datového souboru *autori.txt*. Seznam URL adres domovských stránek jsem umístil do textového/datového souboru *home_url_autori.txt.* Nalezené URL adresy s výpisem publikací jsem umístil rovněž do textového-datového souboru *publikace_url_autori.txt.*

Poněvadž však byl skript původně vyvíjen pro zařazení do celistvého projektu, byl po domluvě s kolegy schválen XML formát vstupních i výstupních dat (bylo využito DTD[13] jazyka XML), který otevírá cestu pro jednodušší komunikaci a propojení jednotlivých komponent kolegů v celek, a navíc splňuje podmínky požadované strukturovanosti. Testovací data nadále zůstala v textovém formátu, poněvadž mi slouží pouze k testování funkčnosti a úspěšnosti skriptu.

Načítání vstupního XML souboru probíhá pomocí knihovny *xml.dom.minidom*, která umožňuje jednoduchý přístup k informacím obsaženým ve formátu XML (ukázka 1) v načítaném vstupním

[11] Vložíme hledaný řetězec (jméno autora) přímo do vyhledávací url adresy, která je odeslanýma z formuláře na stránkách Google.com do vyhledávacího stroje.

[12] Návratová hodnota bude obsahovat nejen URL adresy stránek, ale veškeré značky jazyk HTML, titulky, popisy a stručné obsahy naleznutých souvisejících stránek s hledaným řetězcem.

[13] „DTD (document type definiton) je významnou součástí všech jazyků odvozovaných od SGML, tedy i jazyků HTML, XHTML a XML. DTD je jakousi šablonou, která určuje strukturu daného dokumentu, vymezuje povolené prvky (elementy, atributy) a určuje tak standardy, kterými se dokument řídí. (DTD, document type definiton.“ [dostupné 27. 12. 2007]. [HTML dokument]. [http://www.adaptic.cz/znalosti/slovnicek/dtd.htm].)

souboru. Nejprve je načten celý soubor znak po znak do proměnné typu řetězec. Tento řetězec je postoupen funkci *parseString*[14], která je obsažena právě ve zmiňované knihovně *xml.dom.minidom*. Ta nám z řetězce vytvoří stromovou strukturu, kterou následně po patrech postupně procházíme a získáváme potřebné informace (ukázka 2)

```
<?xml version="1.0" encoding="utf-8" ?>
     <result>
          <author>
            <name>Jméno autora 1</name>
            <university>Vědecké pracoviště 1</university>
          </author>
          <author>
            <name> Jméno autora 2</name>
            <university> Vědecké pracoviště 2</university>
          </author>
     </result>
```

Ukázka 1: *Formát vstupního XML souboru*

[14] Funkce parseString propojuje XML analyzátor s tzv. „DOM builder“, který umí přijmout analyzátorem události z nějakého SAX analyzátoru a převést je do tzv. DOM stromu. [dostupné 04. 05. 2008]. [HTML dokument]. [http://docs.python.org/lib/module-xml.dom.minidom.html].)

```
for i in xmldoc.childNodes:
  if i.nodeType == xml.dom.minidom.Node.ELEMENT_NODE:
    for e in i.childNodes:
      if   e.nodeType   ==   xml.dom.minidom.Node.ELEMENT_NODE   and
e.nodeName == 'author':
    for k in e.childNodes:
      if   k.nodeType   ==   xml.dom.minidom.Node.ELEMENT_NODE   and
k.nodeName == 'name':
        name = k.childNodes[0].nodeValue
        autor.append(name)
      if   k.nodeType   ==   xml.dom.minidom.Node.ELEMENT_NODE   and
k.nodeName == 'university':
        university = k.childNodes[0].nodeValue
        univerzita.append(university)
```

Ukázka 2: *Získávání dat z XML dokumentu*

Testovací data jsou načítána standardním způsobem. Skript načítá každý soubor zvlášť znak po znaku a to do proměnné typu *seznam*. Jakmile narazí na znak „*čárka*“, je první prvek (jméno či url adresa) seznamu načten. Pokračujeme čtením dalšího řádku v souboru, tj. dalšího prvku (URL adresy). Princip načítání se opakuje do doby, než se načte symbol konce souboru. Stejný způsob je použit při načítání URL adres s výpisem publikací.

Seznam URL adres domovských stránek (homepage):

[URL adresa domovské stránky 1, URL adresa domovské stránky 2 , ...]

Seznam URL adres s výpisem publikací:

[URL adresa publikaci 1, URL publikaci 2 , ...]

Formát, ve kterém jsou textové soubory obsahující URL adresy domovských stránek a URL adresy publikaci, byly vytvořen pouze pro účel testování.

3.3 Odeslání řetězce do vyhledávacího Yahoo API rozhraní

Mám vytvořenou strukturu pro nejdůležitější část skriptu, a to odeslaní řetězce do vyhledávacího stroje společnosti Yahoo (výňatek ze zdrojového kódu viz Příloha 1), který využívám zprostředkovaně přes API rozhraní, které běží na adrese:

http://search.yahooapis.com/WebSearchService/V1/webSearch.

Jak už bylo řečeno výše, pro umožnění přístupu ke službám Yahoo, byla nutná registrace a vygenerování aplikačního kódu ID.

V mém případě byl vygenerován kód:

CnxZyrrV34Hy7VTmueJPVkj0rCD3ttmoKGwhFQbiO9TRzjDWzjbGFnE0T5FSopVR0Ao5Jg,

který je pro chod skriptu velmi důležitý.

API rozhraní vrací data ve formátu XML (viz Příloha 2), která je nutné pro naše potřeby upravit. K tomu jsem použil knihovny *lXML* a *lXML eTree*, které mi pomohly s vyseparování potřebných údajů a to konkrétně s URL adresou, které je uvozena „značkovacími závorkami" *<Url> </Url>*. Tuto získanou URL adresu pro další potřeby uložím do seznamu na pozici, která bude odpovídat pozici jména autora v seznamu autorů. V kapitole 4 se budeme zabývat analýzou úspěšnosti a rychlosti daného postupu.

3.4 Odeslání řetězce do vyhledávacího stroje Google.com

Pro srovnání statistických výsledku jsem dále naimplementoval vyhledávání pomoci vyhledávacího stroje od společnosti Google.com. Již teď bohužel víme, že v prosinci 2006 bylo vydání aplikačních kódů pro přístup ke Google Search API pozastaveno[15].

Jak jsem se zmínil už v předešlé kapitole, bohužel se mi nepodařilo najít ani nikde zapůjčit pro testovací účely již dříve vydaný aplikační klíč, tudíž jsem byl nucen využit vyhledávání cestou „napřímo". Využil jsem odkazu, který je odesílán z formuláře na stránce portálu Google.com při vyhledávání jakéhokoliv řetězce (ukázka 3).

*http://www.google.com/search?q=**hledany+retezec**&rls=com.microsoft:cs&ie=UTF-8&oe=UTF-8&startIndex=&startPage=1*

Ukázka 3: *Vložení hledaného řetězce „natvrdo"*

Vložil jsem hledaný řetězec (jméno autora) do URL odkazu na tučně označené místo, avšak pokus o získání jakýchkoliv informací o hledaném řetězci, byl bezvýsledný. Google bohužel

[15] „As of December 5, 2006, we are no longer issuing new API keys for the SOAP Search API. Developers with existing SOAP Search API keys will not be affected." (Google SOAP Search API (Beta). [cit. 02. 01. 2008]. [HTML dokument]. [http://code.google.com/apis/soapsearch/].)

Překlad: K 5. prosinci roku 2006 již nebudou dále vydávány nové API kódy pro přístup k vyhledávacímu rozhraní. Vývojáři s existujícím klíčem tímto nebudou nijak ovlivněni.

neumožňuje svévolné používání některých parametru v externích skriptech. Jediným řešením, kterým se dalo toto omezení obejít, bylo nasimulovaní přístupu libovolným prohlížečem webových stránek a společně s dotazem na Google zasílat příslušnou hlavičku, kterou zasílá prohlížeč, když využívá služeb vyhledávače od společnosti Google.com. Rozhodl jsem se nasimulovat přístup prohlížeče Mozilla/4.0 (ukázka 4).

```
txheaders = {
      'User-Agent': 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)',
      'Accept-Language': 'en-us',
      'Keep-Alive': '300',
      'Connection': 'keep-alive',
      'Cache-Control': 'max-age=0'}
```

Ukázka 4: *Simulace přístupu prohlížeče Mozilla/4.0 pomocí hlavičky*

Funkcí *url_file.read()* stáhneme zdrojový kód „manuálně“ získaného odkazu. Tento zdrojový kód však nemůžeme zpracovat pomocí knihoven lXML, poněvadž se nejedná o formát XML, ale musíme vytvořit novou část skriptu, která nám dokáže vyextrahovat URL adresy ze zdrojového kódu stránky (z HTML kódu). Již teď můžeme předpokládat, že časová náročnost bude vyšší než v případě Yahoo Search API. Provedením analýzy procentuální úspěšnosti a časové náročnosti a srovnáním s výsledky získanými analýzou výstupu z vyhledávacího stroje společnosti Yahoo.com toto tvrzení vyvrátíme, anebo naopak potvrdíme. Pokud rozdíl mezi procentuální úspěšností nálezu správné URL adresy domovské stránky akademického pracovníka přes Google.com bude znatelný a hodnotný, a časová náročnost nebude nabývat extremních hodnot, budeme brát tuto variantu návrhu v potaz.

3.4.1 Separace URL adresy z HTML kódu

Po odeslání vyhledávacího odkazu (ukázka 2) je vrácen zdrojový kód ve formátu HTML, který obsahuje 10 URL odkazů týkajících se hledaného řetězce. Mým úkolem bylo vyseparovat nejjednodušším způsobem těchto 10 URL adres.

Ve zdrojovém kódu bylo nutné najít něco, co by se opakovalo s pevnou pravidelností a bylo nejblíže k hledané URL adrese. Po důkladném prostudování HTML kódu jsem usoudil, že nejsnazší a snad i nejrychlejší cestou bude extrakce URL adresy z místa, kde Google uvádí odkaz na podobné stránky. Konkrétně jsem se zaměřil na parametr „*related*“, za kterým hned po znaku dvojtečky následuje požadovaná URL adresa v plném a nám vyhovujícím tvaru (ukázka 5).

```
<a class=fl href="/search?hl=cs&amp; rls=com.microsoft:*&amp;
q=related:www.fit.vutbr.cz/news/studinfo/2008/msg00200.html">
Podobné stránky</a></nobr></div></td>
```

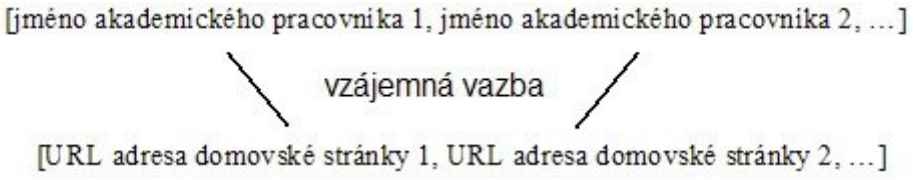
Ukázka 5: *Část zdrojového kódu, z které vyextrahujeme URL odkaz*

Separace do konečné podoby již proběhne pomocí standardních funkcí práce s řetězcem knihovny *string*. Nejprve nalezneme ve zdrojovém kódu řetězec „*related*" a poznamenáme si jeho pozici. Od této pozice zdrojový kód odřízneme a zůstává nám jen zbývající část od poznamenané pozice. V té hledáme první znak dvojtečky, který je bezprostředně před požadovanou URL adresou. Nyní nám zbývá nalézt koncové uvozovky a máme URL adresu označenu z obou stran. Nyní nám již nebrání nic k tomu, abychom URL adresu vyextrahovali a uložili do seznamu.

Stojíme však před dalším problémem a to, že vrácený zdrojový kód obsahuje prvních 10 URL adres a my jich vyžadujeme 100. Řešení je jednoduché. Využijeme cyklu, který při získání 10. URL adresy zvedne počítadlo o deset a zavolá znovu vyhledávací stroj Google.com s vyhledávacím odkazem (ukázka 2), avšak s novou hodnotou u parametru *startPage* (vždy o 10 vetší; parametr značí od kolikáté URL adresy se bude výpis zobrazovat). Na separaci dalších 10 URL ze zdrojového kódu nově vrácené HTML stránky, použijeme stejný postup jako u stránky první.

3.5 Srovnání nalezených URL adres domácích stránek s očekávanými výsledky

Nalezené URL adresy domovských stránek akademických pracovníků jsou uloženy do proměnné typu *seznam*. URL adresa na 1. pozici v seznamu URL adres odpovídá jménu autora na 1. pozici v seznamu jmen akademických autorů (obrázek 1).



Obrázek 1: *Vzájemná vazba mezi seznamy*

Postupně procházím potenciální URL adresy domovských stránek nalezené/vrácené vyhledávacím strojem k hledanému řetězci (akademickému pracovníkovi) a srovnávám s URL adresou uloženou právě ve zmiňovaném seznamu očekávaných výsledků. Před samotným srovnáním ověřuji, zda nalezena URL adresa je v plném tvaru, tzn. má na začátku sama sebe řetězec „*http://*"

nebo řetězec „*https://*“. V případě, že tomu tak není, je tento řetězec před nalezenou URL adresu přidán „*http://*“.

Dále uvažuji případ, že je nalezena URL adresa, která není tzv. v kořenovém tvaru, ale ve tvaru, kdy se odkazuje přímo na nějaký startovací soubor, např. index.html. V tom případě, pokud se shoduje část URL adresy před názvem tohoto souboru, je nalezení považováno za úspěšné (ukázka 6)

```
http://www.domovska_stranka/
http://www.domovska_stranka/index.html
http://www.domovska_stranka/webstart.aspx
```

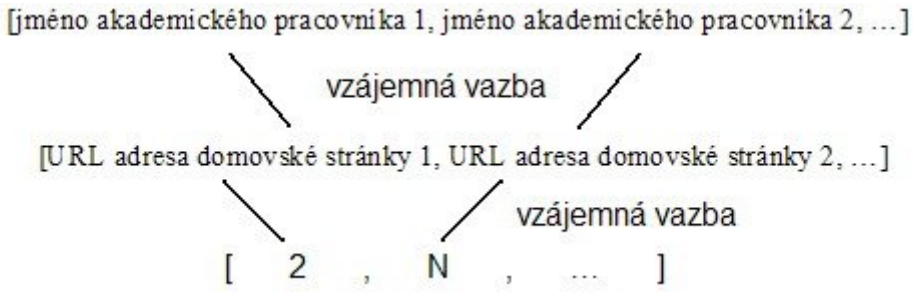
Ukázka 6: *Srovnání dvou URL adres s odlišným startovacím souborem*

Zpočátku předpokládám, že nedosáhneme žádné shody, tzn. URL adresa nalezena nebude, a proto uložíme znak „*N*“ (= nenalezena). V případě shody je znak „*N*“ přepsán pozicí nalezené URL adresy v rámci všech adres vrácených vyhledávacím strojem Yahoo Search API (ukázka7).

```
nalez_homepage.append('N') # předpoklad: nenalezení shodné URL adresy
if url == seznam_homepage[i]: cislo_homepage = cislo_url + 1
                              nalez_homepage[i] = cislo_homepage
                              nalezena_homepage = url
```

Ukázka 7: *Vložení hledaného řetězce „natvrdo“*

Získané informace jsou opět ukládány do proměnné typu *seznam* a vzniká další vzájemná vazba mezi již existujícími seznamy (obrázek 2).



Obrázek 2: *Vzájemná vazba mezi seznamy 2*

Mou snahou je dosáhnout co nejlepších výsledků, a proto jsem naimplementoval další část kódu, která by měla zajistit vyšší úspěšnost nálezu požadované domovské stránky a vylepšit tak pozici, na které vyhledávač tuto stránku vrátí. Tuto změnu k lepším výsledkům by měla přinést skutečnost, že URL stránky se nebudou vyhledávat pouze na základě jména autora, ale byl přidán další upřesňující parametr – vědecké pracoviště autora. V případě, že skript zjistí, že nalezení domovské stránky, která by odpovídala domovské stránce v testovacích datech, je neúspěšné (znak „*N*“), zavolá se znovu vyhledávací stroj, resp. odešle se vyhledávacímu stroji nový požadavek na hledání, avšak již ne pouze se jménem autora, ale i s lokalitou vědeckého pracoviště dané osoby (univerzita, firma atd.). Očekávám zvýšení úspěšnosti a posun nalezených, požadovaných stránek ve vyhledávači na přední pozice. Zda můj předpoklad byl správný, je důkladně rozebráno v kapitole 4.

3.6 Nalezení URL adresy stránky shromažďující publikace

Druhou důležitou částí skriptu je prozkoumání domovské stránky autora a nalezení adresy, která odkazuje na stránku obsahující vydané publikace. V rámci semestrálního projektu jsem vytvořil část skriptu, která zvládá vyextrahovat hledaný řetězec (Příloha 3). Princip separace byl dále v diplomové práci zdokonalován. Postup, který použiji, bude z velké části podobný postupu využitého při sémantické extrakci URL adres ze stránky, kterou nám vrátí vyhledávací stroj Google.com (kapitola 3.4.1). Očekávám mnohem větší časovou náročnost, protože zdrojový kód budu nucen stáhnout pro každou domovskou stránku.

Princip sémantické extrakce URL adres stránek s výpisem publikací

1. Stažení zdrojového kódu domovské stránky
2. Zjištění počtu výskytu hledaného řetězce „publikace“
3. Postupné procházení zdrojového kódu a zaznamenávaní pozice hledaného řetězce
4. Opětovné procházení pro jednotlivé pozice a hledání řetězce „*href*“
5. Nalezení levých uvozovek
6. Nalezení pravých uvozovek
7. Separace potenciálního URL odkazu na stránku s výpisem publikací

Stažení zdrojového kódu domovské stránky

Využijeme následující jednoduchý příkaz (funkce), který však udělá hodně práce:

```
objekt_zdrojovy_kod = urllib.urlopen (url_adresa)
```

Funkce „*urlopen*“ obsažená v knihovně „*urllib*“ stáhne zdrojový kód a je nadále použitelný jako objekt. Z objektu přečteme zdrojový kód a převedeme do formátu řetězce pomocí následující funkce:

```
zdrojovy_kod = url_file.read()
```

Díky této funkci se nám do proměnné „*zdrojovy_kod*“ stáhne kompletní stránka v podobě zdrojového kódu, se kterým budeme dále pracovat. Celý řetězec v této proměnné příkazem:

```
zdrojovy_kod = string.lower(zdrojovy_kod)
```

Zdrojový kód převedeme na malá písmena pro lepší a přesnou práci s řetězcem. Zároveň si do pomocné proměnné uložíme původní obsah zdrojového kódu (nijak neupravený) pro další použití. Nyní máme zdrojový kód připravený pro sémantickou extrakci (ukázka 8) požadovaných URL adres stránek s výpisem publikací.

Zjištění počtu výskytu hledaného řetězce a zaznamenání pozice každého z nich

V druhém kroku k nalezení požadované URL adresy na stránku s výpisem publikací je nalezení řetězce „*publikace*“. Hledaný řetězec nebudeme hledat pouze v češtině (obrázek 3) – to bychom příliš neuspěli - ale budeme vyhledávat anglické slovo *„publications“*, a k němu odpovídající anglická synonyma.

Je více než pravděpodobné, že ne všechny domovské stránky autorů budou v anglickém jazyce (ne všichni autoři pocházejí z anglicky hovořících zemi, anebo mají stránky lokalizované pro celosvětové měřítko), a proto jsme se rozhodli nevyhledávat ve zdrojovém kódu stránky pouze anglické výrazy, ale i výrazy a synonyma v dalších čtyřech nejrozšířenějších evropských jazycích – v němčině, ve francouzštině, v italštině a ve španělštině.

```
publikace = [
     "publikace","knihy",                              #česky
     "publication","books","projects","papers"         #anglicky
     "publikation","buches",                           #německy
     "travail","livres",                               #francouzsky
     "pubblicazioni","libri",                          #italsky
     "publicaciones","libro"                           #španělsky]
```

Obrázek 3: *Překlady slova „publikace“ a jejich synonyma*

Je více než pravděpodobné, že se v kódu bude nacházet určitý hledaný výraz vícekrát, a proto je nejprve provedeno hledání výrazu pro určení počtu výskytu. V okamžiku, kdy je některý z výše

jmenovaných výrazů nalezen, předpokládáme, že jsme nalezli výraz „*publikace*“ v místě jako ve struktuře na obrázku 4, anebo na obrázku 5. Pozici prvního znaku hledaného výrazu si poznamenáme do seznamu, který posléze využijeme při separaci hledaných URL odkazu na stránky s výpisem publikací.

```
<a href="http://www. homepage.cz/odkaz"><b>Publikace</b></a>
```

Obrázek 4: *Místo nálezu výrazu „publikace“*

```
<a href="http://www.homepage.cz/publikace">Výtisky</b></a>
```

Obrázek 5: *Místo nálezu výrazu „publikace“ 2*

Učinili jsme tak první krok k získání správné URL adresy, která by měla odkazovat na stránku s publikacemi. Již nyní víme, že nám může a také s největší pravděpodobností proklouzne stránka, která bude mít odkazy – např. menu - v jiné podobě, než ve kterém jsme schopni získat jeho zdrojový kód. Pokud bude menu vytvořeno např. jako objekt, tzn. nebudeme mít přístup ke zdrojovému kódu, nedokážeme tento odkaz v menu nalézt. Příkladem může být objekt aplikace *Adobe Flash*. My však předpokládáme, že se nám podařilo stáhnout kompletní zdrojový kód domovské webové stránky autor.

Opětovné procházení pro jednotlivé pozice a hledání řetězce „href“, nalezení pravých a levých uvozovek

Načteme zdrojový kód a vezmeme pouze tu část, která se nachází před nalezeným výrazem „*publikace*“ a podrobíme tuto část kódu dalšímu prohledávání. V tomto kroku budeme hledat výraz „*href*“[16]. Máme zjištěno, že se v těsné blízkosti výskytu řetězce „*publikace*“ nachází řetězec „*href*“, což nám dává jistotu, že jsme narazili na potenciálně hledaný URL odkaz.

Po nalezení si opět pozici prvního znaku hledaného výrazu poznamenáme. Od této pozice opět kód ořízneme a zůstává nám část kódu jako v případě obrázku 4. Ořezáním kódu zprava (pozice řetězce „*publikace*“) a zleva (pozice řetězce „*href*“) nyní již máme určené hranice kódu, ve kterém by se měla hledaná URL adresa nacházet (obrázek 6).

```
...href="http://www. homepage.cz/odkaz"><b>...
```

Obrázek 6: *Vyextrahovaná část kódu s hledanou URL adresou*

[16] Atribut HREF (hypertext reference) tagu A jazyka HTML určuje cíl hypertextového odkazu.

Nyní již postačí nalézt pozici prvních a posledních uvozovek a vyextrahovat tíženou URL adresu. Některé z nás může po hlubším zamyšlení napadnout otázka, proč nevyhledáváme uvozovky rovnou a proč se zdržujeme vyhledáváním atributu „*href*“. Představme si část kódu, ve které se nachází nadpis „*Publikace*“ a odkaz na tíženou stránku se nachází jinde, než před námi nalezeným výrazem „*publikace*“ (ukázka 8). V takovém případě bychom vybrali v uvozovkách část kódu, která by s URL adresou neměla nic společného, a to řetězec „*nadpis1*“. Skript však nalezne též druhý výskyt slova publikace a to právě v odkazu *„http://www.homepage.cz/publikace“*, který je dle uvedeného postupu lehce z kódu získatelný.

```
<h4 class="nadpis1">Publikace</h4>
<a href="http://www.homepage.cz/publikace">2007</a>
```

Ukázka 8: *Případ kódu, kde by preventivní nevyhledání výrazu „href“způsobilo chybu*

V úvahu beru i možnost, že se na stránce odkaz na stránku s publikacemi bude nacházet na více místech. Aby v seznamu odkazu nevznikaly duplicitní záznamy, je před vložením kontrolována předešlá existence vkládaného odkazu. V případě, že odkaz v seznamu není, je přidán.

V potaz beru výskyt relativních[17] a absolutních odkazů[18]. Většina odkazů se ve zdrojovém kódu vyskytuje v podobě relativních odkazů (ukázka 9).

```
<a href="publikace.html">Publikace</a>
<a href="http://www.homepage.cz/books.php">Publikace</a>
```

Ukázka 9: *Relativní a absolutní odkazy*

V případě, že je použita relativní adresa (není nalezeno označení protokolu) je označení protokolu doplněno. Do seznamu URL adres na výpisy publikací je tak vždy vložena adresa absolutní (v testovacích datech se též nacházejí pouze adresy absolutní, tj. s označením protokolu).

[17] Relativní odkaz je použit, pokud se odkazujeme na dokument v rámci stránky. Prohlížeč sám doplňuje URL stránky, z které je odkazováno. V případě, že se odkazujeme na dokument nacházející se ve stejném adresáři, stačí napsat jen jméno souboru.

[18] Absolutní odkaz je použit, pokud se odkazujeme na dokument umístěný mimo domovský server. Odkaz tak musí být v plném tvaru a musí začínat označením protokolu (http://)

3.7 Srovnání nalezených URL adres shromažďující publikace s očekávanými výsledky

Po vyhledání URL adres stránek s publikacemi na všech domovských stránkách autorů, které budou uloženy v proměnné typu *seznam*, a budou opět vázány na předchozí seznamy, dojde opět k porovnání tohoto seznamu s očekávanými výsledky (kapitola 4).

Představme si stránku, na které je v menu položka „*Publications*“ (ukázka 10 – zdrojový kód). V takovém případě je skriptem vrácena (nalezena) adresa: *http://www.web.cz/publications.php*, která se shoduje s adresou uvedenou v testovacích datech.

```
<a href=”http://www.web.cz/publications.php”>Publications</a>
```

Ukázka 10: Část zdrojového kódu stránky

A nyní si představme stránku, na které se nevyskytuje v menu pouze položka „*Publications*“, ale navíc i položka „*Books*“ a „*Projects*“ (ukázka 11 – zdrojový kód). Jak bude výsledek, když v testovacích datech je uložena pouze url adresa týkající se položky „*Publications*“?

```
<a href=”http://www.web.cz/publications.php”>Publications</a>
        <a href=”http://www.web.cz/books.php”>Books</a>
    <a href=”http://www.web.cz/projects.php”>Projects</a>
```

Ukázka 11: Část zdrojového kódu stránky

Skript nalezne odkaz týkající se položky „*Publications*“ a uloží ji. Zároveň však nalezne zbývající 2 odkazy, které jsou pro nás též zajímavé a použitelné. Tyto odkazy jsou též uloženy. Analýza později sice hodnotí pouze úspěšnost nálezu odkazu, který se nachází v testovacích datech, avšak uchovává všechny potenciální stránky, na kterých by se výpis s publikacemi mohl nacházet. Výstup skriptu pak vypadá takto:

```
[[jméno autora, domovská stránka],
[http://www.web.cz/publications.php, http://www.web.cz/books.php,
http://www.web.cz/projects.php]]
```

Kapitola 4

Analýza úspěšnosti a časové náročnosti

V kapitole 4 bude analyzovat výsledky, kterých jsem dosáhl testováním skriptu, který jsem vytvořil. Poněvadž budu skript testovat s různými rozhraními a jedná se o prototyp, který je stále modifikovatelný, jsou části kódu, které nejsou používány pro daný test, zakomentovány. V následujících podkapitolách shrnu výsledky, kterých bylo dosaženo již při vývoji prototypu skriptu v semestrálním projektu a budou uvedeny výsledky, které byly pořízeny v rámci diplomové práce. Skript jsem otestoval s Yahoo Search API rozhraním a pro posouzení kladů a záporu jsem skript otestoval i nad Google.com. Jak se dočtete níže, statistický se pro další využití více hodí použití rozhraní Yahoo Search API – jak kvůli lepším výsledkům v nálezu domovské stránky autora, tak v rychlosti a možnosti využívat aplikační rozhraní. Také proto již dále nejsou uváděny výsledky, kterých by bylo dosaženo při použiti Google.com v činnosti s hledáním stránek s výpisem publikaci. Je zřejmě, že pokud nebyly uspokojivé výsledky při hledání domovských stránek, pak je zákonitě jasné, že výsledky nálezu stránky s publikacemi jsou závislé a vychází z úspěšnosti nalezení domovské stránky.

4.1 Zhodnocení úspěšnosti nálezu URL adresy domovské stránky pomocí prototypu skriptu na původních testovacích datech ze semestrální práce

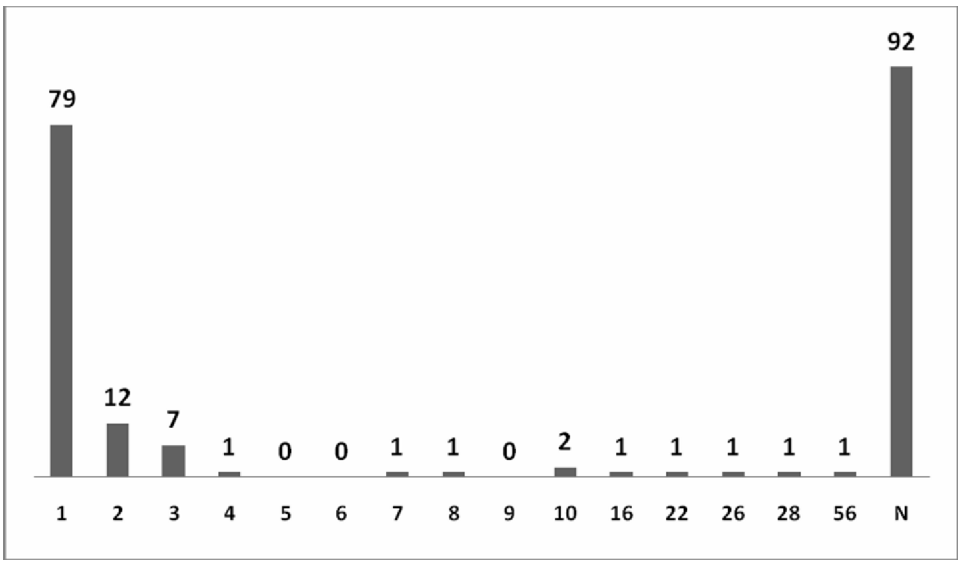
Skript, který jsem navrhl v semestrální práci, jsem otestoval na původních testovacích datech, které obsahovaly 200 jmen autorů. Očekával jsem od Yahoo Search API vrácení URL adres, které by měly odpovídat či být nějakým způsobem blízké hledanému řetězci jména daného autora (ukázka 12). Ke každému jménu nám API rozhraní vrátí 100 URL adres, tj. 100 potenciálních domovských stránek.

```
Domovska stranka autora  steven paul abney nalezena na N. pozici.
Domovska stranka autora  james f allen nalezena na 1. pozici.
     http://www.cs.rochester.edu/u/james/
Domovska stranka autora  hiyan alshawi nalezena na 3. pozici.
     http://research.google.com/pubs/author88.html
Domovska stranka autora  ivana kruijff-korbay nalezena na N. pozici.
Domovska stranka autora  susan armstrong nalezena na 10. pozici.
     http://www.chem.gla.ac.uk/~susana/armstrong.html
Domovska stranka autora  ricardo baeza-yates nalezena na N. pozici.
Domovska stranka autora  michael barlow nalezena na 1. pozici.
     http://www.michaelbarlow.com/
```

Ukázka 12: Část v*ýstupu skriptu informujícího o pořadí hledané URL adresy vzhledem k ostatním vráceným adresám z Yahoo Search API*

Nás však zajímalo nejen pořadí, na kterém se nalézala URL adresa domovské stránky shodná s URL adresou v naší testovací sadě[19] v rámci všech nabídnutých URL adres, ale i časová náročnost této operace při zmiňovaném počtu 200 jmen autorů a celková procentuální úspěšnost navrhnutého skriptu s vybraným API rozhraním.

Jak můžeme vidět (graf 1), shodná URL adresa byla nalezena na 1. pozici 79x z 200 URL domovských stránek. Z ukázky 6 můžeme vyčíst, že 39,5 % URL adres se nám přes Yahoo Search API podařilo nalézt okamžitě na prvním místě. Nalézt URL adresu se nám podařilo celkem 108x, což značí zarážející 54% úspěšnost z 200 URL adres. Domovské stránky v testovací sadě byly nalezeny ručně přes portál Google.com. Je možné, že by byl rozdíl mezi nabízenými URL adresami portálu Google.com a Yahoo Search API rozhraní až tak extrémní? Je možné, že se ve skriptu nachází chyba, anebo je tato 46% neúspěšnost pravdivá a Yahoo Search API skutečně nabízí odlišné URL adresy.



Graf 1: *Statistika použití Yahoo Search API - hledání URL podle jména autora. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

Je logické, že v případě použití vyhledávacího stroje portálu Google.com „napřímo“ bychom získali mnohem lepší výsledky, když víme, že URL adresy v testovací sadě pochází právě z tohoto vyhledávače.

[19] Testovací sada = sada, obsahující seznam všech URL adres domovských stránek vědeckých pracovníků „ručně“ nalezených. O těchto adresách víme, že existují a jsou skutečně domovskými stránkami daných pracovníků.

Časová náročnost tohoto skriptu při odesílání 200 jmen akademických pracovníků, generování 100 URL adres potenciálních domovských stránek ke každému pracovníkovi (tzn. 20 000 URL adres) a celkové vyhodnocení pozic vzhledem k testovací sadě, byla naměřena: 7 min a 12 sec.

4.2 Použití rozhraní Yahoo Search API

V následujících kapitolách provedu analýzu získaných výsledku, provedu srovnání a pokusím se najít příčiny úspěšnosti/neúspěšnosti skriptu, který bude využívat rozhraní Yahoo Search API. Na základě analýzy pak bude doporučeno rozhrání, které se bude nejlépe svou úspěšností, spolehlivostí a rychlostí hodit pro náš skript.

4.2.1 Procentuální úspěšnosti nálezu URL adresy domovské stránky (Yahoo)

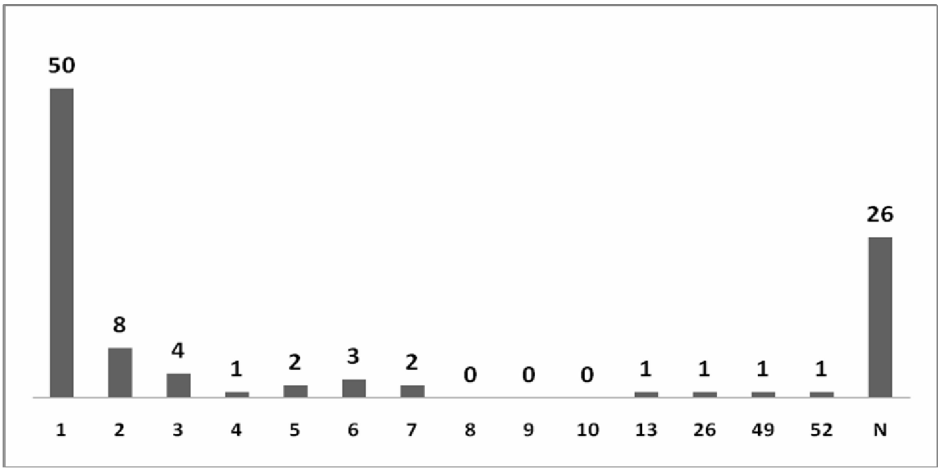
Úspěšnost skriptu byla testována hned na 4 variantách:

- Odeslání dotazu se jménem autora
- Odeslání dotazu se jménem autora a jeho působiště
- Odeslání dotazu se jménem autora v podobě fráze[20]
- Odeslání dotazu se jménem autora a jeho působiště v podobě fráze

Při používání vyhledávačů v běžných prohlížečích jsme zvyklí zadávat frázi do dvojitých uvozovek. V odeslání přes Yahoo API jsem narazil na problém a při použití dvojitých uvozovek API nevrací žádné výsledky. Je tedy nutné používat uvozovky jednoduché.

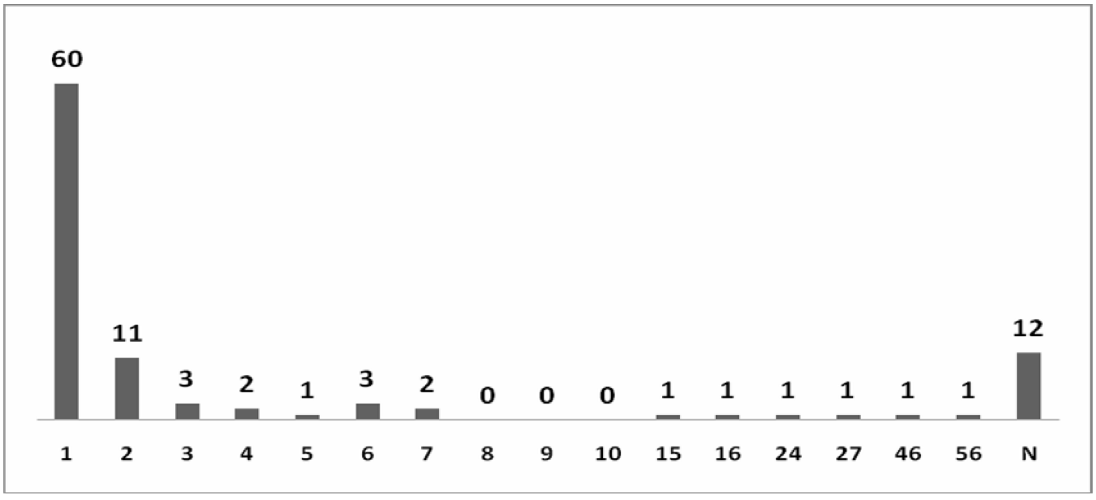
Ve všech případech byla použita aktuální testovací sada (tj. 100 URL adres domovských stránek, které byly zjištěny ze seznamu vědeckých pracovníku – kapitola 2.1).

[20] Vyhledávače umožňují vyhledávat požadovanou informaci jako celek = frázi, zachovávající si pořadí slov, anebo jako jednotlivá slova, která se mohou na stránce vyskytovat v libovolném pořadí a četnosti



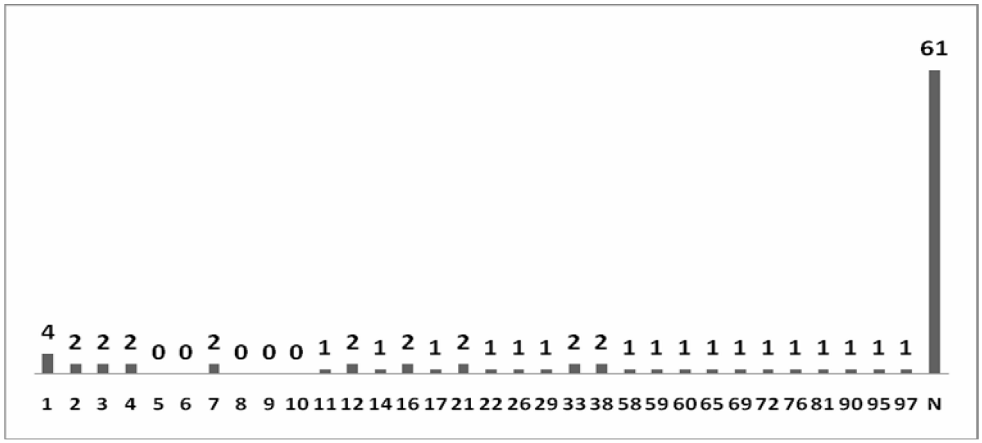
Graf 2: *Statistika použití Yahoo Search API - hledání URL podle jména autora. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

Z grafu 2 můžeme vyčíst, že počet URL adres, které se shodovaly s adresami v testovací sadě oproti statistice ze semestrální práce, přibylo. Celých 50% adres se umístilo na první pozici a rovných 70% z celkového počtu se vešlo v pořadí do první desítky. Neúspěšnost klesla na 26%.



Graf 3 *Statistika použití Yahoo Search API - hledání URL podle jména autora a působiště. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

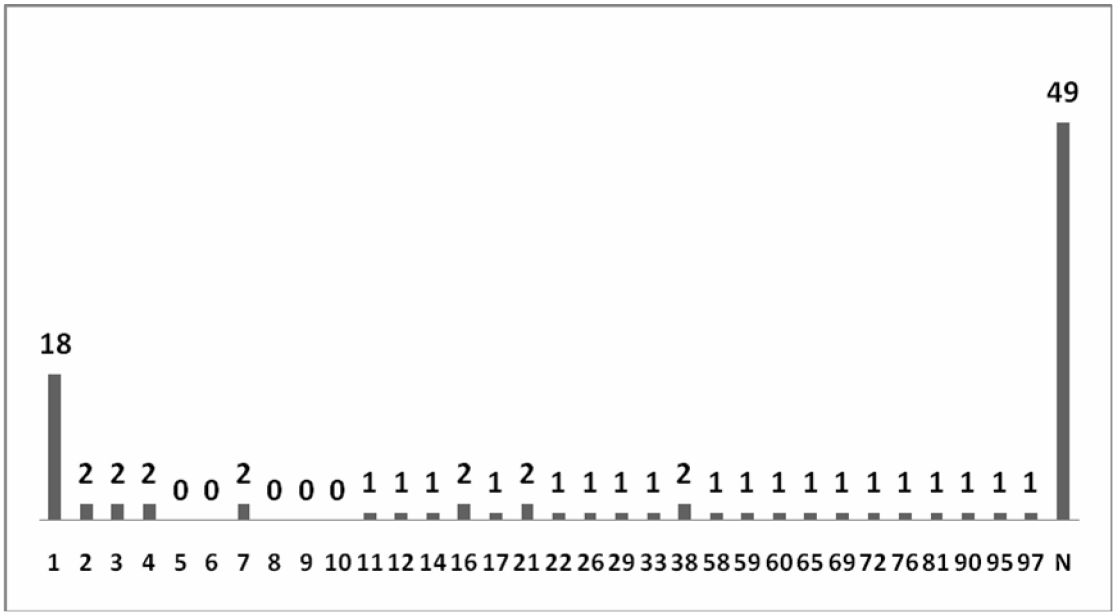
Při použití nejen jména ale i působiště autora se úspěšnost výskytu na 1. místě zvýšila o rovných 10% a neúspěšnost, tj. počet nenalezených URL adres na domovskou stránku autora se snížila z 26 na 12 adres, což odpovídá 12% neúspěšnosti.



Graf 4: *Statistika použití Yahoo Search API - hledání URL podle jména autora v podobě fráze. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

Graf 4 vyjadřuje překvapivou statistiku při hledání jména ve formě fráze. 61% domovských stránek z testovacích dat nebylo vůbec nalezeno a zbývajících 39% je rozmístěno po celém spektru grafu. Pouhé 4% URL adres byly nalezeny na první pozici a pouhých 12% ze 100 URL adres se dostalo do první desítky. Yahoo API vrací v případě fráze absolutně odlišné URL adresy oproti případu, když jméno autora je zasláno jen jako řetězec. Čím může být tato rozdílnost způsobena nedokážu vysvětlit.

Z grafu 5 je zřejmé, že přidání parametru navíc v podobě působiště (univerzity) se úspěšnost nálezu na první pozici zlepšila, ale velmi malým podílem. Celkem 18% URL adres domovských stránek se shodovala s URL adresou v testovacích datech a zároveň se nacházelo na první pozici nalezených stránek. Téměř polovina adres nebyla nalezena vůbec. Do první desítky se dostalo 26% adres, což je velmi špatný výsledek. Zbývající URL adresy jsou rozmístěny opět po celém spektru grafu jako v předchozím případě.



Graf 5: *Statistika použití Yahoo Search API - hledání URL podle jména autora a působiště v podobě fráze. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

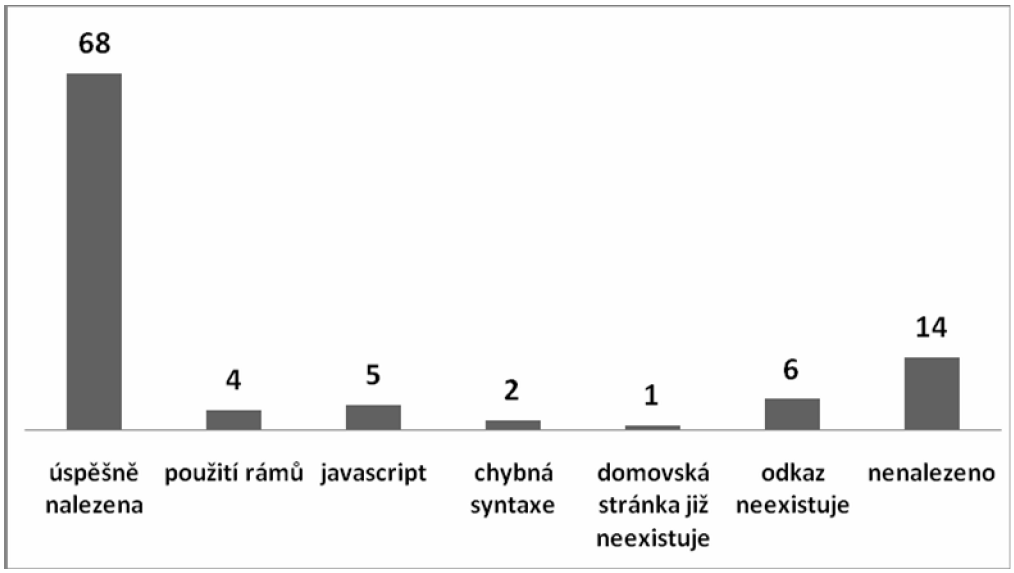
4.2.2 Procentuální úspěšnosti nálezu URL adresy stránky s výpisem publikaci (Yahoo)

Jak jsem se zmínil už v úvodu této kapitoly, pro další vývoj skriptu již bylo využitou pouze Yahoo Search API rozhraní, a to na základě lepších výsledků v poměru úspěšnost/rychlost.

Z grafu 6 můžeme vyčíst, že 68% odkazů se podařilo bez problémů najit. Zbývajících 32% vykázalo neúspěch. Poněvadž je toto číslo relativně vysoké, ručně jsem prošel každou domovskou stránku zvlášť a snažil se najít příčinu, proč se nepodařilo odkaz, který máme uložený v testovacích datech, nalézt. Příčin bylo hned několik (viz graf 6).

Skript prochází zdrojový kód domovské stránky a snaží se nalézt potenciálně správný odkaz (princip viz kapitola 3.6.1). V případě, že má k dispozici nevyhovující zdrojový kód, který neodpovídá domovské stránce, pak je zřejmé, že nemůže odkaz nalézt. Použití rámu[21] je jedním z případů, kdy se tak děje. Obsahem stažené stránky je pouhé rozvržení hlavní stránky a podstrání jsou doplněny až po načtení.

[21] „Rámům se anglicky říká frames a dnes jsou častou součástí webových stránek. V tomto textu se snažím na příkladu znázornit, jak to funguje, a objasnit základní syntaxi. Okno prohlížeče je rozděleno na několik obdélníkových částí (rámů). V každém rámu je (třebaže to nemusí být vidět) samostatná HTML stránka (soubor). Zpravidla je jeden z rámů menu s odkazy, které se po kliknutí realizují v jiném, hlavním okně.“ [dostupné 12. 04. 2008]. [HTML dokument]. [http://www.jakpsatweb.cz/ramy.html].)



Graf 6: *Statistika použití vyhledávače Yahoo.com - hledání odkazu na podstránku s výpisem publikací na domovské stránce. Graf vyjadřuje počet nalezených odkazů ve srovnání s testovacími daty. Symbol N značí URL adresy, které nebyly nenalezeny.*

Dalším případem je použití Javascriptu, kdy je odkaz volán pomocí vložené funkce (ukázka 13), anebo je odkaz součástí menu, které je též vkládáno jako modul.

```
javascript:cont('pub.htm')
```

Ukázka 13: *Časová náročnost skriptu (Yahoo) při hledání domovských stránek*

V několika případech odkaz na domovské stránce neexistoval vůbec a výpis publikací se nacházel přímo na domovské stránce autora. Tyto stránky jako úspěšně vyhodnoceny nebyly a byly zařazeny zvlášť, do kategorie: „odkaz neexistuje“.

Dalším případem je chybná syntaxe HTML kódu, který se tak stává pro analyzátor kódu nepoužitelný (princip viz kapitola 3.6.1).

V jednom případě jsem dokonce narazil na případ, kdy domovská stránka, kterou jsem uvedl do testovacích dat, již neexistuje. Domovské stránky byly všechny jedna po druhé ručně procházeny a byla jejich funkčnost a dostupnost prověřována. Pokud byly nefunkční, anebo nebyly dostupné, nebyly do testovacích dat zařazeny, ač byly součástí oficiálního seznamu domovských stránek autorů.

4.2.3 Časová náročnost skriptu (Yahoo) při hledání domovských stránek

Zaznamenal jsem rychlost skriptu při všech čtyřech činnostech – při hledání jména, jména a univerzity, a při hledání obou případů ve formě fráze. Časovou náročnost bohužel výrazně neovlivníme, poněvadž nejvyšší náročnost na čas připadá odezvě Yahoo Search API rozhraní a na rychlosti s jakou vrací výsledky.

V případě hledání jména bez použití fráze se doba zpracování úkolu pohybuje kolem 3 a půl minuty. Pokud hledáme jméno včetně univerzity, časová náročnost se výrazně zvedne u hledání ve formě fráze a pohybuje se kolem 6 min (ukázka 12). Téměř dvojnásobná hodnota má jednoduché vysvětlení. Yahoo Search Api nevrátí požadovanou domovskou stránku, která by se shodovala se stránkou uloženou v testovacích datech, a proto je ke jménu přidáno působiště a Yahoo Search Api je voláno znovu, avšak už ve tvaru „*jméno působiště*“.

Hledaný výraz	Formát hledání	Časová náročnost výpočtu
Jméno	Bez fráze	3 min 32 sec
Jméno + působiště	Bez fráze	4 min 06 sec
Jméno	Fráze	3 min 22 sec
Jméno + působiště	Fráze	5 min 43 sec

Ukázka 14: *Časová náročnost skriptu (Yahoo) při hledání domovských stránek*

4.2.4 Celková časová náročnost

Při porovnání časové náročnosti a úspěšnost nalezení požadované URL adresy (odpovídající adrese v testovacích datech), docházím k závěru (viz kapitola 4.2.1), že ideálním nastavením bude použití vyhledávače Yahoo Search Api a hledání skrze jméno a působiště. Můžeme argumentovat, že v některých případech má rozhraní Google.com lepší časové požadavky, avšak uvědomme si, že bereme v potaz i výsledky úspěchu nalezení domovské stránky shodující se se stránkou uloženou v testovacích datech.

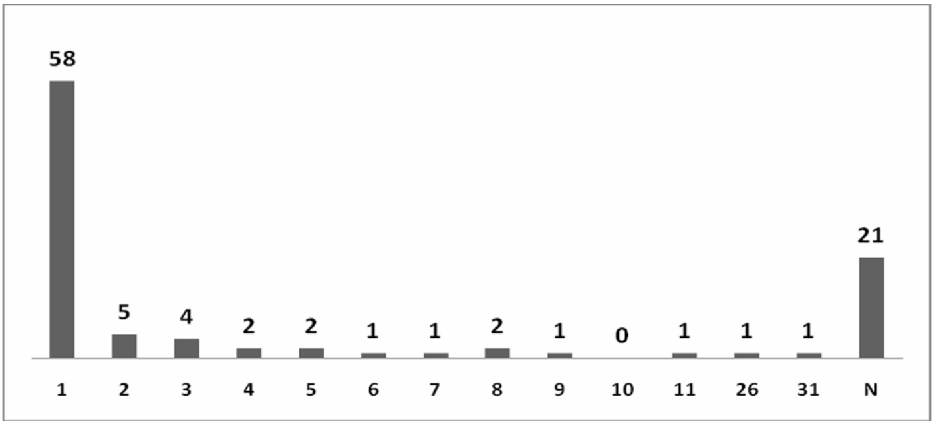
4.3 Použití rozhraní Google.com

4.3.1 Procentuální úspěšnosti nálezu URL adresy domovské stránky (Google)

Úspěšnost skriptu byla opět otestována na 4 variantách:

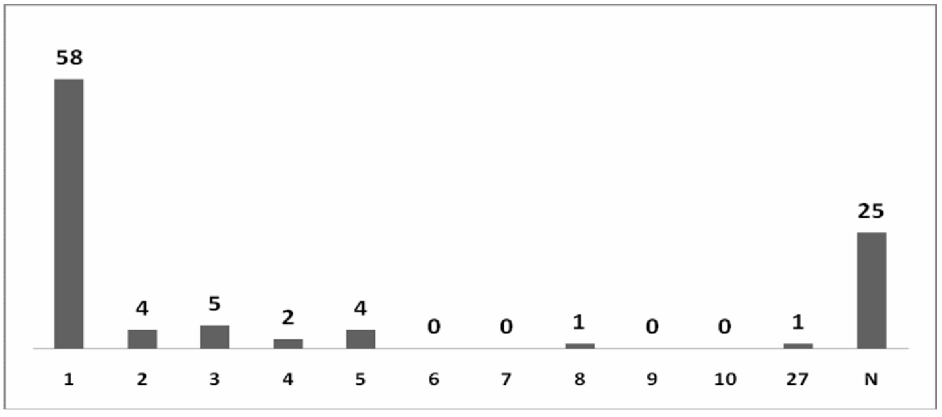
- Odeslání dotazu se jménem autora
- Odeslání dotazu se jménem autora a jeho působiště

- Odeslání dotazu se jménem autora v podobě fráze
- Odeslání dotazu se jménem autora a jeho působiště v podobě fráze



Graf 7: *Statistika použití vyhledávače Google.com - hledání URL podle jména autora. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

Výsledek použití Google.com je velmi příznivý (graf 7). Úspěšnost v první desítce umístění dosahuje 76%. Neúspěšnost je rovna 21%. Všechny nalezené URL adresy se umístily do prvních 30. pozic.



Graf 8: *Statistika použití vyhledávače Google.com - hledání URL podle jména autora a působiště. Graf vyjadřuje počet nalezených adres na určité pozici v rámci všech vrácených URL adres. Symbol N značí URL adresy, které nebyly nenalezeny.*

Při pohledu na graf 8 vidíme, že se veškeré výsledky posunuly do první desítky (74%), avšak počet nenalezených adres se kupodivu zvýšil - je poměrně vysoký – 25%, ač jsem jako u Yahoo

Search API předpokládal, že přidáním parametru navíc (působiště/univerzita) se úspěšnost zlepší. Na 1.pozici se nachází 58% jako v předchozím případě, což je poměrně uspokojivý výsledek.

Při použití fráze je úspěšnost totožná s výsledky, kdy fráze používána není!

4.3.2 Časová náročnost skriptu (Google) při hledání domovských stránek

Hodnoty, kterých dosahuje skript při použití rozhraní Google.com jsou srovnatelné s použitím Yahoo Search API. Poněvadž však skriptu Google nevrací výsledek ve formátu XML, ale v HTML (viz podkapitola 3.4.1), extrahujeme z kódu URL adresy (od rozhraní Yahoo je zasílán XML formát) pomocí sémantických pravidel, což rapidně zvedá časovou náročnost provádění skriptu. Toto zpomalení kompenzuje rychlejší odezva Google.com a tím se rozdíly mezi uvedenými rozhraními snižují (ukázka 15).

Hledaný výraz	Formát hledání	Časová náročnost výpočtu
Jméno	Bez fráze	3 min 24 sec
Jméno + působiště	Bez fráze	4 min 42 sec
Jméno	Fráze	3 min 27 sec
Jméno + působiště	Fráze	3 min 40 sec

Ukázka 15: *Časová náročnost skriptu (Google.com) při hledání domovských stránek*

4.4 Použití prototypu skriptu

Posledními analyzovanými výsledky jsou ty, které nám vrátí skript, který již je použitelný pro začlenění do celistvého projektu.

Skript se skládá z následujících části kódu (v závorce je uvedeno, co daná část vykonává a jaká byla zjištěna její časová náročnost):

- načtení vědeckých pracovníků/autorů a jejich působiště/pracoviště/univerzity
- načtení URL adres domovských stránek autorů z testovací dávky
- načtení URL adres stránek s výpisy publikací autorů z testovací dávky
- odeslání do Yahoo Search API a získání XML souboru s nalezenými adresami
- stažení jednotlivých, nalezených domovských stránek (zdrojový kód)
- vyhledání URL adres odkazujících na stránky s výpisem publikací
- vytvoření výstupního XML dokumentu

Tyto operace budou provedeny vždy, jakmile bude skript začleněn do společného projektu. Časová náročnost tohoto skriptu je 6 až 8 minut. Dle momentální odezvy Yahoo Search API.

Skript lze spustit příkazem z příkazové řádky:

```
python main.py > vystup.xml
```

Závěr

Z analýzy výsledků (kapitola 4.1), kterou jsem provedl už v rámci semestrálního projektu, jsem zjistil, že hodnoty úspěšnosti skriptu vytvořeného pro semestrální práci jsou natolik nevyhovující, že cíl diplomové práce byl jasný – znovu otestovat funkčnost skriptu a nejspíš rozšířit vstupní data o další parametry, které by zúžili a hlavně upřesnili oblast hledaných URL adres. Zabýval jsem se tedy nejprve touto velmi nízkou procentuální úspěšností při hledání domovských stránek vědeckých pracovníků. V situaci, že by se nepodařilo procentuální úspěšnost zvýšit ani přidáním dalšího vyhledávacího parametru (např. vědeckého ústavu, domovské univerzity, případně státu působnosti), byl jsem připraven skript postavit na jiném vyhledávacím stroji.

Poněvadž bylo až podezřelé, že v semestrální práci vychází takové statistiky, vytvořený skript jsem od základu přetvořil. Jednoznačně jsem tak dosáhl nejen vyšší rychlosti získání požadovaných informací, ale v první řadě se statistické výsledky velmi zřetelně zlepšili (viz kapitola 4). I přesto jsem však do skriptu naimplementoval rozšiřující parametr – autorovo působiště (pracoviště, univerzitu). Skript opět důkladně otestoval a zjistil jsem opět nemalé zlepšení. Sice se zvýšila časová náročnost skriptu, ale zvýšení není natolik dramatické, abychom se museli rozhodnout od tohoto rozšíření ustoupit.

Nově jsem nashromáždil aktuální testovací dávku, která čítá 100 jmen a působišť, URL adres domovských stránek a odkazu na stránky s výpisem publikací vědeckých pracovníků (kapitola 2). Původní data využívaná v semestrálním projektu se stala neaktuální – ve smyslu nedostupných a nefunkčních stránek.

Byla vytvořena konečná podoba formátu vstupních a výstupních dat – využil jsem moderního a velmi podporovaného formátu XML, který je pro naše účely nejperspektivnější, poněvadž nám umožňuje strukturovanou skladbu textu – informace. Tato skutečnost byla prokonzultována s ostatními kolegy a vybrána za nevhodnější.

V semestrální práci se nám podařilo v jazyce Python vytvořit poměrně krátký, avšak funkčně hodně bohatý skript, který nám vytvořil dobrý základ pro jeho další vývoj v rámci této diplomové práce. Skript jsem zdokonalil a dosáhl tak příznivějších výsledků.

Skript jsem rozběhl na dvou vybraných vyhledávacích rozhraních – Yahoo Search API a Google Search API (nakonec nepoužito a skript přizpůsoben portálu Google.com). Otestoval jsem hledání pomocí fráze i bez fráze. Na základě provedené analýzy se nejperspektivnějším rozhraním pro náš skript jevilo Yahoo Search API, a proto bylo použito i pro další vývoj. Veškeré výsledky analýz a důvody proč nebylo použito rozhraní Google.com najdete v kapitole 4.

Co se týče časové náročnosti, rychlost, které dosahuje je velmi závislá na odezvě používaného vyhledávacího rozhraní (Yahoo vs. Google). V případě velkého zatížení sítě se pohybuje i kolem 10 min. Obvykla (průměrná) doba trvání však nepřesahuje hranici 5 minut.

Skript byl vyvíjen a testován pod operačním systémem Windows Vista a byla instalována poslední verze jazyka Python (verze 2.5.2). Základní funkčnost byla otestována i pod operačním systémem Linux.

Skript je připraven pro další použití v projektu, který je vyvíjen ve spolupráci s ostatními kolegy. Propojení však nemohlo do termínu dokončení diplomové práce být uskutečněno, poněvadž někteří z kolegů dokončení své práce odložili.

Jediný požadavek, který je kladen na skripty, které by chtěly využit služeb mého skriptu, je striktní dodržení vstupního formátu (struktury) XML dokumentu a přizpůsobily své skripty skutečnosti, že skript opět vrací výstupní soubor opět ve formátu XML.

Celkově tato práce byla pro mne přínosem a naučil jsem se v krátké době mnoho nových věci. Pronikl jsem do tajů rychle se rozvíjejícího se jazyka Python, který byl před diplomovou prací pro mne velkou neznámou a nikdy jsem o něm neslyšel. Jazyk je velmi přátelský a umožňuje vytvářet jednoduché programy dělající složité úkoly.

Literatura

[1] *Uniform Resource Locators (URL).* [dostupné 29. 12. 2007]. [HTML dokument]. [http://tools.ietf.org/html/rfc1738].

[2] Harms, D., MacDonald, K.: *Začínáme programovat v jazyce Python*. 1, Brno: Computer Press 2006. 476 s. ISBN: 80-7226-799-X.

[3] HAVELKA, J. a kolektiv. *Vytváříme WWW stránky a spravujeme moderní web site*. 6. Praha: Computer Press, 2002. 355 s. ISBN 80-7226-748-5

[4] DigiWeb. Ekonom.Ihned.cz. *Soumrak nad giganty*. [dostupné 29. 12. 2007]. [HTML dokument]. 09. 08. 2007. [http://digiweb.ihned.cz/c6-10134700-21775700-i00000_d-soumrak-nad-giganty].

[5] Adaptic. *DTD, document type definiton*. [dostupné 27. 12. 2007]. [HTML dokument]. [http://www.adaptic.cz/znalosti/slovnicek/dtd.htm].

[6] Google.com. *Google SOAP Search API (Beta)*. [dostupné 02. 01. 2008]. [HTML dokument]. 2006. [http://code.google.com/apis/soapsearch/].

[7] *Python Library Reference. Xml.dom.minidom.* [dostupné 04. 05. 2008]. [HTML dokument]. [http://docs.python.org/lib/module-xml.dom.minidom.html].

[8] *Rámy na webu.* [dostupné 12. 04. 2008]. [HTML dokument]. [http://www.jakpsatweb.cz/ramy.html].)

Seznam příloh

Příloha 1

Výňatek ze zdrojového kódu

„zaslání řetězce do vyhledávajícího stroje přes Yahoo API“

```
NS = { 'ns': "urn:yahoo:srch" }

url = 'http://search.yahooapis.com/WebSearchService/V1/webSearch'

appid =

'CnxZyrrV34Hy7VTmueJPVkj0rCD3ttmoKGwhFQbiO9TRzjDWzjbGFnE0T5FSopVR0Ao5Jg'



params = urllib.urlencode({

        'appid': appid,

        'query': hledany_vyraz,

        'results': 100

})



data = urllib.urlopen(url, params).read()

etree = lxml.etree.fromstring(data)
```

…

Příloha 2

Výňatek z kódu ve formátu XML vráceného API rozhraním


```
<?xml version="1.0" encoding="UTF-8"?>

<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:srch" xsi:schemaLocation="urn:yahoo:srch
http://api.search.yahoo.com/WebSearchService/V1/WebSearchResponse.xsd"
type="web" totalResultsAvailable="294000" totalResultsReturned="99"
firstResultPosition="1"
moreSearch="/WebSearchService/V1/webSearch?query=steven+paul+abney&amp;appi
d=CnxZyrrV34Hy7VTmueJPVkj0rCD3ttmoKGwhFQbiO9TRzjDWzjbGFnE0T5FSopVR0Ao5Jg&am
p;region=us">

<Result>

<Title>Steven (Steve) Abney</Title>

<Summary>Steven Paul Abney. He was a grammarian, and could doubtless see
further into the ... Here is my curriculum vitae in brief. ...</Summary>

<Url>http://www.vinartus.com/spa/</Url>

<ClickUrl>http://uk.wrs.yahoo.com/_ylt=A9htfSEoGnxHE_wAah_dmMwF;_ylu=X3oDMT
B2b2gzdDdtBGNvbG8DZQRsA1dTMQRwb3MDMQRzZWMDc3IEdnRpZAM-
/SIG=11ggt8kp2/EXP=1199401896/**http%3A//www.vinartus.com/spa/</ClickUrl><D
isplayUrl>www.vinartus.com/spa/</DisplayUrl><ModificationDate>1171008000</M
odificationDate><MimeType>text/html</MimeType>

<Cache><Url>http://uk.wrs.yahoo.com/_ylt=A9htfSEoGnxHE_wAax_dmMwF;_ylu=X3oD
MTBwZTdwbWtkBGNvbG8DZQRwb3MDMQRzZWMDc3IEdnRpZAM-
/SIG=18koljdm7/EXP=1199401896/**http%3A//216.109.125.130/search/cache%3Fei=
UTF-
8%26query=steven%2Bpaul%2Babney%26results=99%26appid=CnxZyrrV34Hy7VTmueJPVk
j0rCD3ttmoKGwhFQbiO9TRzjDWzjbGFnE0T5FSopVR0Ao5Jg%26u=www.vinartus.com/spa/%
26w=steven%2Bpaul%2Babney%26d=bWfUaLXiP7vX%26icp=1%26.intl=us</Url><Size>21
83</Size></Cache>

</Result>
```


▪▪▪

Příloha 3

Skript extrahující URL adresu stránky s publikacemi ze zdrojového kódu

```
i = 0
url_publikace = []
pocet_url_publications = 0
mnozina_poctu_url_publikaci = []
konec_seznam_url_yahoo = len(seznam_url_yahoo)

while i < konec_seznam_url_yahoo:
  #pripravime si zakladni mnozinu, ktera bude obsahovat [jmeno autor,
homepage, a sem budeme pridavat URL publikaci]
  url_publikace_vnitrni = [[seznam_url_yahoo[i][0],seznam_url_yahoo[i][1]]]
  #nacteneme si zdrojovy kod do objektu
  if seznam_url_yahoo[i][1] == 'nothing':
      url_publikace_vnitrni2.append('nothing')
      url_publikace_vnitrni.append(url_publikace_vnitrni2)
      url_publikace.append(url_publikace_vnitrni)
      i = i + 1
      continue
  objekt_zdrojovy_kod = urllib.urlopen(seznam_url_yahoo[i][1])
  #do promenne nam objekt vrati zdrojovy kod
  zdrojovy_kod = string.lower(objekt_zdrojovy_kod.read())
  zdrojovy_kod_archiv = zdrojovy_kod
  objekt_zdrojovy_kod = urllib.urlopen(seznam_url_yahoo[i][1])
  zdrojovy_kod_nezmensene = objekt_zdrojovy_kod.read()

  publikace = [
  "publikace","knihy", #cesky
  "publication","books","projects","papers" #anglicky
  "publikation","buches", #nemecky
  "travail","livres", #francouzsky
  "pubblicazioni","libri", #italsky
  "publicaciones","libro" #spanelsky
  ]

  #HLEDAME VYRAZ "publikace" a zjistujeme pocty ve zdrojovem kodu
  v = 0
```

```
  delka_publikace = len (publikace)
  pocet_pruchodu = 0
  while v < delka_publikace: #dokud nepouzijeme vsechny hledane vyrazy
slova "publikace"
    #urci pocet vyskytu pro konkretni vyraz slova "publikace"
    pocet_url_publications = string.count(zdrojovy_kod, publikace[v])
    if pocet_url_publications > 0:
      #pridame do mnoziny ve tvaru: [hledany vyraz, pocet vyskytu]

mnozina_poctu_url_publikaci.append([publikace[v],pocet_url_publications])
    v = v + 1



  #ZJISTUJEME POZICE vyrazu "publikace"
  a = 0
  mnozina_pozic_url_publikaci = []
  delka_mnozina_poctu_url_publikaci = len(mnozina_poctu_url_publikaci)
  while a < delka_mnozina_poctu_url_publikaci:
    v = 0
    zdrojovy_kod = zdrojovy_kod_archiv
    mnozina_pozic_url_publikaci_vnitrni = []
    while v < mnozina_poctu_url_publikaci[a][1]:
      pozice_publications = 0

      zdrojovy_kod = string.join(zdrojovy_kod,"")
      #nalezneme ke kazdemu vyrazu "publikace" jeji polohu ve zdrojovem
kodu
      pozice_publications = string.rfind(zdrojovy_kod,
mnozina_poctu_url_publikaci[a][0])
      #pridame do mnoziny ve tvaru: [hledany vyraz, pozice vyrazu]

mnozina_pozic_url_publikaci_vnitrni.append([mnozina_poctu_url_publikaci[a][
0],pozice_publications])
      #odmazeme cast kodu odzadu (od mista vyskytu vyrazu) a zbytek
pouzijeme pro dalsi hledani pozice
      zdrojovy_kod = zdrojovy_kod[:pozice_publications]
      v = v + 1
    mnozina_pozic_url_publikaci.append(mnozina_pozic_url_publikaci_vnitrni)
    a = a + 1
```

```
  z = 0
  y = 0
  pozice_publications = 0
  delka_mnozina_poctu_url_publikaci = len(mnozina_poctu_url_publikaci)
  pocet_pruchodu = 0

  url_publikace_vnitrni2 = []
  while y < delka_mnozina_poctu_url_publikaci:
    pocet_pruchodu = mnozina_poctu_url_publikaci[y][1]
    z = 0
    while z < pocet_pruchodu:
      pozice_publications = mnozina_pozic_url_publikaci[y][z][1]
      if pozice_publications > 0:
        #abychom byli schopni nalezt odkaz k nalezenemu slovu
"publications" musim odstranit cast nepotrebneho retezce, aby prvni
nalezeny tag "href" patril prave tomuto slovu "publications"
        #prevod na seznam, abychom mohli odstranin cast retezce
        zdrojovy_kod_prevod = list(zdrojovy_kod_archiv)
        #odstraneni retezce od daneho mista
        zdrojovy_kod_prevod[pozice_publications:] = []
        #prevod zpet na retezec
        zdrojovy_kod2 = string.join(zdrojovy_kod_prevod,"")
        #nalezeni pozice tagu "href" od konce retezce
        pozice_href = string.rfind(zdrojovy_kod2, "href")
        if pozice_href < 0:
          pozice_href = string.rfind(zdrojovy_kod2, "HREF")
        #od nalezene pozice "href" nalezneme prvni znak uvozovek, ktery
uvozuje pocatek hledaneho odkazu
        cast_kodu = zdrojovy_kod_nezmensene[pozice_href:]
        pozice_poc_uvozovek = string.find(cast_kodu, "\"")
        #nalezneme ukoncujici znak uvozovek - hledame jiz v celem zdrojovem
kodu, protoze pri prvnim orezu mohlo dojit k odriznuti casti hledaneho
odkazu
        pozice_konc_uvozovek = string.find(cast_kodu, "\"",
pozice_poc_uvozovek+1)
        #prevedeme zdrojovy kod na seznam a pozadovany odkaz na
"publications" ziskame vyjmutim casti kodu v rozmezi vyhledanych pozic
uvozovek
```

```
        hledany_odkaz =
cast_kodu[pozice_poc_uvozovek+1:pozice_konc_uvozovek]

        original_url = ''
        hledany_odkaz3 = ''
        existence_url = 0
        hledany_odkaz2 = list(hledany_odkaz)
        #vezmeme z retezce prvnich 7 znaku a zjistime, zda se rovnaji
retezci "http://"
        hledany_odkaz3 = hledany_odkaz2[:7]
        hledany_odkaz4 = hledany_odkaz2[:8]
        hledany_odkaz3 = string.join(hledany_odkaz3,"")
        hledany_odkaz3 = string.find(hledany_odkaz3, "http://")
        hledany_odkaz4 = string.join(hledany_odkaz4,"")
        hledany_odkaz4 = string.find(hledany_odkaz4, "https://")
        #jestli podretezec "http://" nenalezneme, pak musim URL adresu
udelat "rucne", vzor: http:// www.neco.cz/ + publikace.html
        if hledany_odkaz3 < 0 and hledany_odkaz4 < 0:
          # musime se vsak dostat do korenove casti odkazu, tzn. z url typu
"http://www.neco.cz/index.html" musime dostat "http://www.neco.cz/"
          original_url = seznam_url_yahoo[i][1]
          pozice_konce_url = string.rfind(original_url, "/")
          #odstrani vse, co nasleduje za znakem "lomitko"
          original_url = original_url[:pozice_konce_url+1]
          original_url = string.join(original_url,"")
          #nyne jsme ziskali korenovou cast url odkazu, takze ted k tomu
pridame stranku s publikacemi, tzn.  "http://www.neco.cz/" +
"publikace.html"
          hledany_odkaz = list(hledany_odkaz)
          original_url2 = original_url
          #pokud se na zacatku nachazi lomitko, pak jej odstarnime,
ponevadz by se nam pak v URL vyskytovalo 2x za sebou
          #odstraneni prebytecnych tecek z URL (dusledek pouzivani
relativnich a absolutnich adres)
          if hledany_odkaz <> []:
            while hledany_odkaz[0] == '/' or hledany_odkaz[0] == '.':
              del hledany_odkaz[0]
          original_url = original_url2
          hledany_odkaz = string.join(hledany_odkaz,"")
```

```
          hledany_odkaz = original_url + hledany_odkaz
          if hledany_odkaz not in url_publikace_vnitrni2:
            url_publikace_vnitrni2.append(hledany_odkaz)
        #v pripade, ze odkaz je jiz odkazem ve spravnem tvaru, pridame do
mnoziny, ktera ma tvar [jmeno autora, url homepage, url publikaci]
        else:
          if hledany_odkaz not in url_publikace_vnitrni2:
            url_publikace_vnitrni2.append(hledany_odkaz)
      z = z + 1
    y = y + 1
    #PRIDANI NALEZENE URL ADRESY PUBLIKACI DO SEZNAMU
  url_publikace_vnitrni.append(url_publikace_vnitrni2)
  url_publikace.append(url_publikace_vnitrni)
  i = i +1
```

Příloha 4

Ukázka výstupního souboru v XML formátu

```
<?xml version="1.0" encoding="utf-8" ?>
- <output>
    - <autor>
      <name>Tony Abou-Assaleh</name>
      <homepage>http://tony.abou-assaleh.net/</homepage>
    - <url_publications>
      <url_1>http://tony.abou-assaleh.net/publications</url_1>
      </url_publications>
      </autor>
    - <autor>
      <name>Lada Adamic</name>
      <homepage>http://www-personal.umich.edu/~ladamic/</homepage>
    - <url_publications>
      <url_1>http://www-personal.umich.edu/~ladamic/publications.html</url_1>
      <url_2>http://www-
    personal.umich.edu/~ladamic/blogsinfodiffusion/index.html</url_2>
      </url_publications>
      </autor>
    - <autor>
      <name>Asier Alcazar</name>
      <homepage>http://www-scf.usc.edu/~alcazar/</homepage>
    - <url_publications>
      <url_1>http://www-scf.usc.edu/~alcazar/#_top</url_1>
      <url_2>http://www-scf.usc.edu/~alcazar/#_List_of_publications</url_2>
      </url_publications>
      </autor>
    - <autor>
      <name>Ash Asudeh</name>
      <homepage>http://http-server.carleton.ca/~asudeh/</homepage>
      <url_publications />
      </autor>
    - <autor>
      <name>Tania Avgustinova</name>
      <homepage>http://www.coli.uni-saarland.de/~tania/index.html</homepage>
    - <url_publications>
      <url_1>http://www.coli.uni-saarland.de/~tania/publications.html</url_1>
      <url_2>http://www.coli.uni-saarland.de/~tania/proj.html</url_2>
      </url_publications>
  </autor>
</output>
```

Příloha 5

Ukázka výstupu skriptu při testování

Vstupní data

```
<result>
  <author>
     <name>Hans Goebl</name>
     <university>Universität Salzburg</university>
  </author>
</result>
```

Domovská stránka (Testovací data**):**

http://www.sbg.ac.at/rom/people/prof/goebl/

Domovská stránka s publikacemi (Testovací data**):**

http://www.sbg.ac.at/rom/people/prof/goebl/

Výstup vyhledávacího skriptu:

```
[[[' Hans Goebl', 'http://www.sbg.ac.at/rom/people/prof/goebl/'],
['http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm',
'http://www.sbg.ac.at/rom/people/prof/goebl/dm_publi.htm',
'http://www.sbg.ac.at/rom/people/prof/goebl/publik_r.htm',
'http://www.sbg.ac.at/rom/people/prof/goebl/sl_publi.htm']]]
```

Výstup statistického skriptu:

Domovská stránka:

```
ze seznamu: http://www.sbg.ac.at/rom/people/prof/goebl/
nalezena: http://www.sbg.ac.at/rom/people/prof/goebl/
```

Publikace:

```
ze seznamu:
http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm
nalezena:
http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm
```

Nalezeno!

```
ze seznamu:
http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm
nalezena:
http://www.sbg.ac.at/rom/people/prof/goebl/dm_publi.htm
ze seznamu:
http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm
nalezena:
http://www.sbg.ac.at/rom/people/prof/goebl/publik_r.htm
ze seznamu:
http://www.sbg.ac.at/rom/people/prof/goebl/publikat01.htm
nalezena:
http://www.sbg.ac.at/rom/people/prof/goebl/sl_publi.htm
```

Příloha 6

Příručka pro použití skriptu

Skript se nachází na přiloženém disku ve složce „*diplomova_prace*“ a má název „*main.py*“. Zároveň jsou v této složce soubory nutně potřebné pro spuštění a to soubor „*home_url_autori*“ obsahující URL adresy domovských stránek autorů, soubor „*publikace_url_autori*“ obsahující URL adresy na stránky s výpisem publikací a v neposlední řadě soubor „*xml_vstup.xml*“, který obsahuje jména autorů a jejich pracoviště/univerzitu.

Pro správný chod skriptu je nutné mít nainstalované tyto knihovny:

- import urllib
- import lxml
- import lxml.etree
- import string
- import urllib2
- import xml.dom.minidom

Skript je pak možné spustit z příkazové řádky voláním:

- **python main.py > vystup.xml**

Výstupem skriptu je pak soubor „*vystup.xml*“ obsahující následující strukturu:

```
<?xml version="1.0" encoding="utf-8" ?>
- <output>
        - <autor>
                <name>Tony Abou-Assaleh</name>
                <homepage>http://tony.abou-assaleh.net/</homepage>
              - <url_publications>
                        <url_1>http://tony.abou-assaleh.net/publications</url_1>
                </url_publications>
          </autor>
…
    </output>
```

Příloha 7

Obsah přiloženého CD

- diplomova_prace
  - pisemna_prace
    - diplomova_prace.doc
    - diplomova_prace.pdf
    - přílohy.doc
    - přílohy.pdf
    - desky.doc
  - skript
    - main.py
    - xml_vstup.xml
    - home_url_autori.txt
    - publikace_url_autori.txt