FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce

KIV/NET

datum odevzdání: 30. 5. 2021

Ondřej Drtina
A20N0077P
drtinao@students.zcu.cz

Obsah

1 Řešené téma

1.1 Zvolené téma

Jako téma své semestrální práce jsem si zvolil systém umožňující indexaci a následné prohledávání dat. Pro zpracování tohoto tématu jsem se rozhodl zejména proto, že jsem chtěl spojit semestrální práci s předmětem KIV/IR, jež má téma fixní pro všechny studenty. V rámci uvedeného předmětu však není definován programovací jazyk, ve kterém má být práce zhotovena - mohu tedy využít C#.

1.2 Zadání

Níže jsou uvedeny základní body, jež musí splňovat semestrální práce z předmětu KIV/NET (viz `https://courseware.zcu.cz/portal/studium/courseware/kiv/net/samostatna-prace/index.html`):

- část aplikace musí být zpracována v jazyce C#
- musí být využity principy objektového programování
- aplikace musí pracovat s persistentními daty
- aplikace obsahuje grafické uživatelské rozhraní ve formě WPF, WinForms
- program musí být dodán s dokumentací obsahující zadání, analýzu, implementaci a uživatelskou příručku

Vzhledem ke spojení s předmětem KIV/IR musí být splněny i požadavky kladené na semestrální práci z daného předmětu, mezi které patří zejména (viz `https://courseware.zcu.cz/portal/studium/courseware/kiv/ir/samostatna-prace.html`):

- získání dat ze zvoleného webu (crawling)
- implementace tokenizace, preprocessingu
- vytvoření in-memory invertovaného indexu
- implementace tf-idf, kosinové podobnosti

- vrácení x nejlepších výsledků na dotaz (dle relevance)
- vyhledávání s pomocí logických operátorů AND, OR, NOT
- program musí být dodán s dokumentací (programátorskou i uživatelskou)

2 Analýza

2.1 Získání dat - podmínky

Nejprve jsem si pro semestrální práci z předmětu KIV/IR musel zvolit web, ze kterého budu získávat informace, jež budou následně zpracovávány. Po prozkoumání struktury několika webů jsem si stanovil podmínky, které musí vybraný web splňovat. Popis jednotlivých kritérií a zdůvodnění jejich důležitosti je předmětem následujících podkapitol.

2.1.1 Minimální výskyt JavaScriptu

Na cílovém webu se nesmí vyskytovat JS či se vyskytuje v minimální míře.

V dnešní době se mi nepodařilo najít web obsahující články, který by JS nepoužíval vůbec. Nicméně jsem se pokusil zvolit web takový, z nějž je možné získat celý obsah článku bez interakce s JavaScriptem. Podmínka byla vytvořena zejména proto, že většina parserů webových stránek s JS neumí pracovat a jeho vyžadování by zbytečně zvýšilo náročnost finální aplikace.

2.1.2 Konstantní struktura všech stránek

Cílový web musí mít obdobnou strukturu všech stránek obsahujících články.

Při mechanickém zpracování obsahu článků na stránce je vhodné mít jeden univerzální algoritmus, jež zajistí získání potřebných dat. V případě, že by měl každý z dostupných článků výrazně odlišnou strukturu, bylo by získání dat velmi obtížné či nemožné.

2.1.3 Povolení autora webu

Autor (či autoři) webu nesmí explicitně zakazovat parsování informací z jeho (či jejich) stránek.

Snažil jsem se najít web, na kterém autor nemá uvedeno, že zakazuje strojové zpracování informací uvedených na jeho webu. Takových webů je minimum, je však k diskusi, jak moc je strojové získávání dat etické.

Jistě by nebylo vhodné dávat zpracovaný obsah webu k dispozici široké veřejnosti, jelikož ze strany autora by docházelo k úniku zisku z reklamy. Vzhledem k univerzitnímu využití je však tato činnost, dle mého názoru, akceptovatelná.

2.2 Získání dat - zhodnocení

Data jsem se rozhodl získávat ze stránek `https://diit.cz/`, jelikož splňují dříve definované požadavky. Tedy články se zobrazí i bez aktivního JavaScriptu, stránky mají relativně konstantní strukturu a jejich autor explicitně nezakazuje strojové zpracování.

2.3 Způsob uchování dat

Bylo nutné definovat struktury, ve kterých budou zpracovaná data uchovávána. Ze strany vyučujícího KIV/IR byl doporučen klasický textový soubor (přípona txt).

Vzhledem k jednoduchosti následného zpracování textového souboru pomocí jazyka C# jsem se rozhodl využít txt formátu, přičemž ještě bylo potřeba uvážit způsob rozdělení do souborů. Zpravodajské weby zpravidla obsahují náhledy článků s krátkým popisem (tzv. perex). V rámci svého produktu při získávání dat z webu uchovávám hlavní text článku i perex.

2.4 Algoritmy zpracování dat

Před začátkem tvorby produktu bylo potřeba obeznámení se s algoritmy, jež lze použít pro vyhledávání informací v dokumentech.

2.4.1 Preprocessing

Před prováděním vyhledávání v datech je vhodné data předzpracovat. Tedy například odebrat ze získaných dat tzv. stop slova, která předávaným informacím nepřidávají žádný význam (např. "je"). Další typickou úlohou preprocessingu je stemming, který má za cíl získat základní tvar slova.

Některé ze zmíněných algoritmů byly popsány v rámci cvičení z předmětu IR, stačilo si je tedy pouze před implementací zopakovat.

2.4.2 TF-IDF a kosinová podobnost

Po preprocessingu jsou dostupná data i dotazy ohodnoceny TF-IDF algoritmem. Při vyhodnocování dotazů dojde k výpočtu kosinové podobnosti mezi daty a dotazy, přičemž za nejvíce relevantní dokument k dotazu je považován dokument, jež dosáhl nejvyšší hodnoty kosinové podobnosti.

S algoritmem jsem byl obeznámen v rámci předmětu KIV/IR, v rámci analytické části jsem si znovu prošel příslušnou přednášku.

2.4.3 Booleovské vyhledávání

Jelikož semestrální práce musí vyjma vyhledávání pomocí vektorového modelu umožňovat i vyhledávání booleovským modelem, bylo v rámci analytické části vyžadováno studium algoritmů, které umožní zpracování dotazů obsahujících boolean operátory.

Dotazy je v rámci semestrální práce dovoleno zpracovat pomocí knihovny, avšak rozhodl jsem se pro vlastní implementaci.

3 Vlastní rozšíření programu

V rámci semestrální práce z KIV/IR bylo možno implementovat rozšíření za bonusové body. V rámci své SP jsem implementoval následující rozšíření (bodové ohodnocení převzato ze zadání):

- File-based index (1b)
- GUI/webové rozhraní (2b)
- zvýraznění hledaného textu v náhledu výsledků (1b)
- dokumentace psaná v TEXu (1b)
- vlastní implementace parsování dotazů bez použití externí knihovny (1-?b)
- napovídání keywords (1b)
- indexování webového obsahu (2-3b)

Detailní popis jednotlivých rozšíření je předmětem následující kapitoly.

3.1 File-based index

Při vytvoření indexu je výsledek uložen do souboru "index.drsearch", jež se nachází v kořenové složce programu. V případě testovacího indexu je pro uložení indexu využit soubor "indexTestData.drsearch", který je rovněž v kořenové složce.

Struktura souborů je shodná. Na první řádce souboru je uveden indexovací režim, který programu sděluje, zda bylo indexování provedeno pro booleovský model či vektorový model. Daná řádka rovněž obsahuje informace charakterizující typ dat, které byly indexovány - data získaná z internetu crawlerem / testovací data.

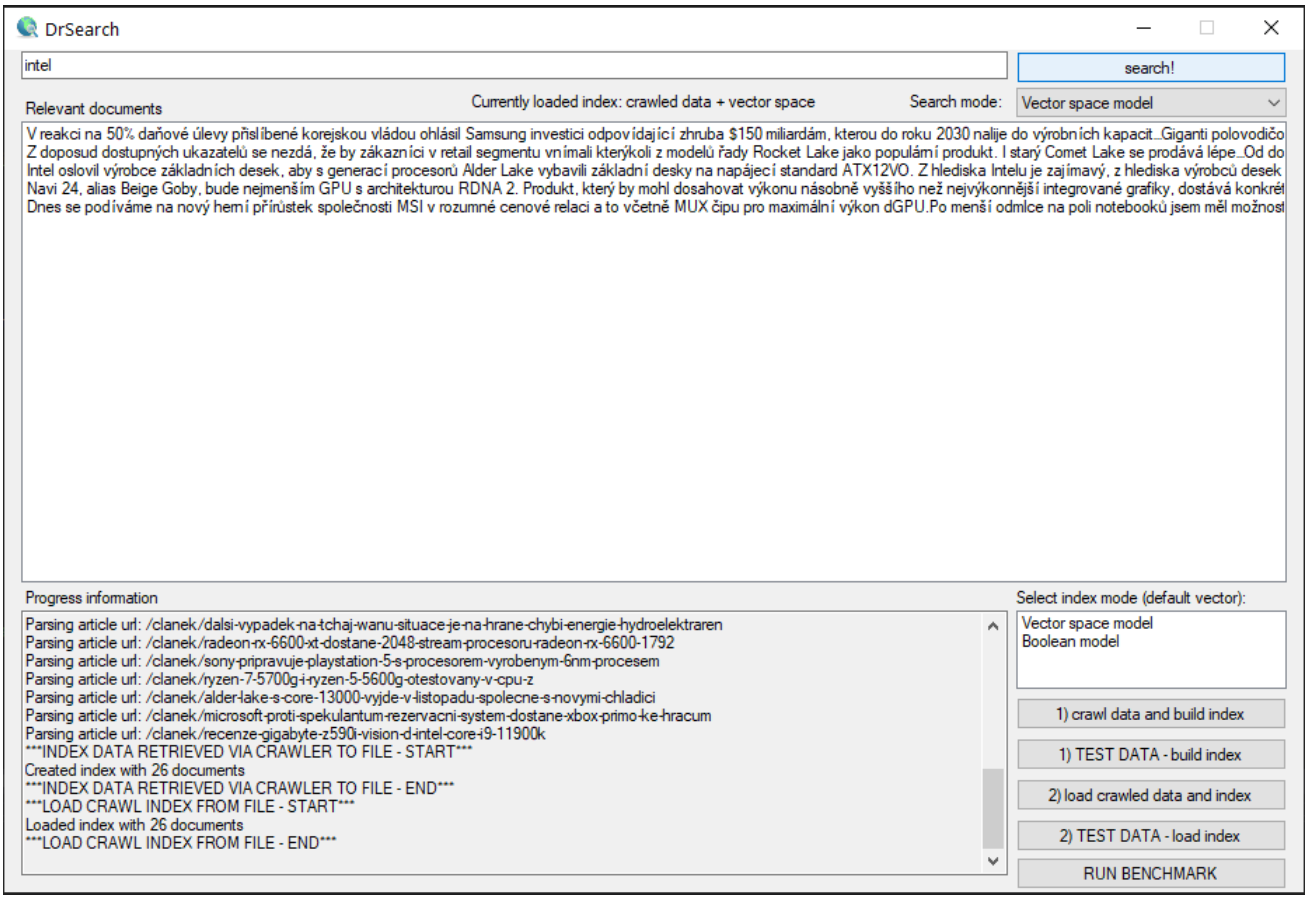
řádek charakterizující režim tedy může nabývat hodnot:

- INDEX_DRSEARCH_VECTOR_SPACE_MODE_CRAWL
  => data získaná webu + vektorový model
- INDEX_DRSEARCH_VECTOR_SPACE_MODE_TEST
  => testovací data + vektorový model

- INDEX_DRSEARCH_BOOLEAN_MODE_CRAWL
  => data získaná webu + booleovský model
- INDEX_DRSEARCH_BOOLEAN_MODE_TEST
  => testovací data + booleovský model

3.2 GUI/webové rozhraní

Vzhledem ke spojení semestrální práce s předmětem KIV/NET bylo uživatelské rozhraní vytvořeno prostřednictvím WinForms. Z gui může uživatel provést indexování dat, načtení indexu a samozřejmě samotné vyhledávání. Za nadstandardní prvek považuji výpis právě probíhajících operací a možnost spustit benchmark přímo z uživatelského rozhraní. Výsledek je možno vidět na obrázku 3.1.



Obrázek 3.1: Ukázka uživatelského rozhraní programu

3.3 Zvýraznění hledaného textu v náhledu výsledků

Při vyhledávání je v detailu výsledků hledaný text zvýrazněn červenou barvou, jež ho odlišuje od zbytku textu.

3.4 Dokumentace psaná v TEXu

Tento dokument je psán v TEXu, konkrétně jsem využil online editor Overleaf.

3.5 Vlastní implementace parsování dotazů bez použití externí knihovny

Vyhledávání, a samozřejmě i indexování, probíhá bez využití jakékoliv knihovny. Při vyhledávání pomocí vektorového i booleovského modelu jsou využity pouze vlastní implementace algoritmů.

3.6 Napovídání keywords

Při psaní výrazu do vyhledávání se uživateli zobrazuje nápověda v poli "Did you mean".

3.7 Indexování webového obsahu

V GUI po klepnutí na tlačítko "crawl data and build index"dojde ke stažení dat a vytvoření indexu s daty, jež byla získána z webu `https://diit.cz/`. Je možno zaindexovat i část dat získaných z webu, uživatel nemusí čekat na stažení obsahu celého webu.

4 Programátorská příručka

Kapitola popisuje dekompozici programu, účel jednotlivých tříd a jejich obsah. Detailní popis metod je samozřejmě ve třídách dostupný prostřednictvím komentářů.

4.1 Třídy pro získání a zpracování dat z webu

Předmětem kapitoly je popis tříd, jež zajišťují získání dat z webu `https://diit.cz/` a jejich následné zpracování.

4.1.1 CrawlerCore.cs

Třída obsahuje metody, které získají data ze zvoleného webu. Po získání dat jsou ostatní metody ve třídě využity k analýze kódu a vytvoření odpovídajících objektů Page, resp. Article. Pro získání dat je využita knihovna HtmlAgilityPack.

4.1.2 Page.cs

Reprezentuje obsah jedné stránky získané crawlerem. Třída tedy obsahuje atributy, které uchovávají kompletní obsah stránky ve formě html i pouhého textu, jež je dostupný na stránce (bez html tagů). Objekt rovněž obsahuje i List<Article>, jež shromažďuje veškeré články dostupné na dané stránce ve formě odpovídajících objektů.

4.1.3 Article.cs

Představuje jeden článek dostupný na zvoleném webu. Atributy této třídy umožňují uložení url článku, jeho textu i html formy. Atributů je více, pro více informací viz popis v souboru Article.cs.

4.2 Třídy pro výpočty

Kapitola popisuje třídy, které program využívá pro výpočty kosinové podobnosti. Rovněž jsou popsány objekty, které s kosinovou podobností souvisí.

4.2.1 TfIdfCore.cs

Obsahuje metody pro výpočet TF-IDF skóre.

Pro výpočet TF je využit obecně známý vzorec:
TF-WEIGHT = 1 + log(TF-RAW), kde:

- TF-RAW - počet výskytů slova v rámci dokumentu

Výpočet IDF se provede vydělením celkového počtu dokumentů počtem dokumentů, ve kterém se dané slovo vyskytuje a následným zlogaritmováním výsledku. Výpočet lze tedy vyjádřit vzorcem:
IDF = log(počet_dokumentů/v_kolika_dok_slovo)
TF-IDF je pak pouze produktem vynásobení dvou zmíněných hodnot.

4.2.2 CosSimMatch.cs

Reprezentuje výsledek vyhledávání, který byl na základě výpočtu kosinové podobnosti, vyhodnocen jako relevantní pro vyhledávanou frázi.

Obsahuje tedy především atributy umožňující uchování odkazu na dokument odpovídající vyhledávanému výrazu, vyhledávaný výraz a číselný údaj nesoucí výsledek výpočtu kosinové podobnosti pro dané vyhledávání a daný dokument.

4.3 Třídy pro práci s GUI

Následující kapitoly popisují třídy, jež se v aplikaci starají o vykreslení grafického uživatelského rozhraní a provádění změn v GUI.

4.3.1 DashboardForm.cs

Třída zajišťující vykreslení hlavního okna programu. Kromě metod, které GUI vykreslí, obsahuje i metody, které mohou volat instance jiných tříd když chtějí zajistit provedení změny v GUI. Mezi takové metody patří například addToRelDocsLBBuffer(), jež zajistí zobrazení dokumentů relevantních k hledanému výrazu.

Metod, které umožní změnu GUI z jiné instance je více. Vlákno uživatelského rozhraní periodicky provádí kód metody UpdateGUI(), jež zkontroluje, zda některé vlákno nepožaduje změnu a případně provede pož. změny.

4.3.2 DetailForm.cs

Zajišťuje vykreslení okna, jež obsahuje detaily týkající se určitého článku. Jedná se například o nadpis článku, datum publikace a celý obsah článku. Konstruktor této třídy samozřejmě přebírá odpovídající informace.

Okno s detailem se zobrazí po klepnutí na některý z výsledků v sekci "Relevant documents".

4.3.3 LoginForm.cs

Vykreslí okno umožňující přihlášení uživatele. Po úspěšném přihlášení se uživateli zobrazí dashboard. Z okna se lze přesunout na registraci, případně změnu hesla a smazání účtu.

4.3.4 SignupForm.cs

Umožňuje registraci uživatele. Pro registraci je požadován login a heslo.

4.3.5 ChangePassForm.cs

Zajišťuje změnu hesla u již existujícího účtu.

4.4 Třídy pro práci s indexem

Sekce obsahuje popis tříd, které program využívá pro zpracování File-based indexu a další operace související s indexováním.

4.4.1 IndexManager.cs

Obsahuje metody, které zajistí vytvoření File-based indexu. Pro data získaná z webu lze za vstupní bod vytvoření indexu považovat metodu performDataIndex() a pro testovací data je využita metoda performTestDataIndex(). Obě metody po zpracování odpovídajících dat volají metodu saveIndexToFile(), jež zajistí zápis indexu do odpovídajícího souboru.

Shodná třída umožňuje i načtení File-based indexu. Tento proces v případě dat získaných z webu zajišťuje metoda performCrawlIndexReload(). Načtení indexu testovacích dat je prováděno metodou performTestIndexReload(). Obě z uvedených metod následně volají parametrizovanou metodu reloadIndexFromFile(), jež obsahuje samotnou logiku čtení indexu.

4.4.2 TestDocIndex.cs

Instance této třídy reprezentuje jeden testovací dokument určený pro indexování. Obsahuje například atributy umožňující uchování nadpisu článku a jeho obsahu, i datum publikace článku.

4.4.3 TestDocQuery.cs

Objekt reprezentuje jeden výraz, jež má být v indexovaných dokumentech vyhledán. Názvy atributů jsou tedy shodné s atributy, jež se vyskytují ve vstupním json souboru.

4.5 Ostatní třídy

Předmětem je popis tříd, jež nelze zařadit do žádné z předcházejících kategorií.

4.5.1 DatabaseManager.cs

Třída pracuje se souborem, jež reprezentuje databázi ve formě sqlite. Do databáze jsou ukládány informace o uživatelích programu. Každý uživatel je reprezentován loginem a heslem, které je v DB uloženo ve formě MD5 hashe.

Třída splňuje CRUD, tedy:

- CREATE - vytvoření záznamu uživatele - createUser()
- READ - ověření, zda odpovídá login a hash hesla - validLogin()
- UPDATE - aktualizace uživatelského hesla - updateUserPass()
- DELETE - odstranění uživatele - deleteUser()

4.5.2 SupportTools.cs

Třída obsahuje výhradně statické metody, které se používají v ostatních částech programu na více místech. Jsou zde obsaženy například metody pro:

- načtení obsahu textového souboru
- převod dat získaných z webu na objekty typu Article
- získání počtu jednotlivých slov v určitém dokumentu

Metod je samozřejmě více, pro více informací viz dokumentační komentáře v příslušném souboru.

5 Uživatelská příručka

Obsahuje požadavky pro spuštění programu a popis práce s programem.

5.1 Požadavky

Program byl vytvořen v jazyce C#, pro jeho úspěšné spuštění je tedy vhodné mít nainstalovaný aktuální .NET Framework (viz `https://dotnet.microsoft.com/download/dotnet-framework`). Windows 10 mají .NET Framework již integrovaný, uživatelé s tímto operačním systémem tedy nepotřebují instalovat žádný dodatečný software.

Požadavky kladené na CPU a RAM nebyly stanoveny a věřím, že vyhledávání je možné i na slabších strojích než byl testovací notebook s i5-5200U a 12GB RAM. Je však očekávatelné, že vytvoření indexu bude na pomalejších strojích zdlouhavé a v případě testovacích dat je vhodné využít již vytvořený index a ten pouze načíst.

5.2 Spuštění programu

Pokud splňujete požadavky deklarované v předchozí sekci, můžete program spustit dvojklikem na soubor "DrSearch.exe".

5.3 Provádění operací

Obsahuje popis kroků, kterými lze dospět k vykonání určité operace.

5.3.1 Vytvoření uživatelského účtu

Pro práci s aplikací je nejprve potřeba vytvořit uživatelský účet. Poté, co se zobrazí okno "Sign in", stiskněte tlačítko "sign up", čímž zahájite proces registrace. Následně si zvolte přihlašovací jméno a heslo, vyplňte příslušná pole. Pokud je zvolené uživatelské jméno k dispozici, uvidíte informační okno a následně budete připuštěni k přihlášení.

Pokud je Vámi zvolené uživatelské jméno již zabrané, zobrazí se upozornění - zvolte si prosím jiné jméno. Upozornění se zobrazí i v případě, že jste zadali odlišná hesla - zkontrolujte si zadaná hesla.

5.3.2 Přihlášení uživatele

Pokud již máte vytvořený uživatelský účet, přepněte se do okna "Sign in"a vyplňte požadované údaje (login a heslo). Poté stiskněte tlačítko "sign in". V případě, že jsou Vámi zadané přihlašovací údaje správné, dojde k přihlášení a zobrazí se Vám dashboard aplikace. V opačném případě uvidíte upozornění, jež informuje o skutečnosti, že bylo zadáno špatné jméno či heslo.

5.3.3 Změna hesla uživ. účtu

V okně "Sign in"nejprve vyplňte Vaše přihlašovací údaje, poté klepněte na tlačítko "change password". Pokud jsou stávající přihlašovací údaje validní, budete vyzváni k zadání nového hesla a jeho zopakování. Následně stačí stisknout tlačítko "change password", potvrdit a dojde ke změně hesla.

V případě, že původní uživatelské údaje nejsou platné, uvidíte upozornění na tuto skutečnost a změna hesla nebude možná.

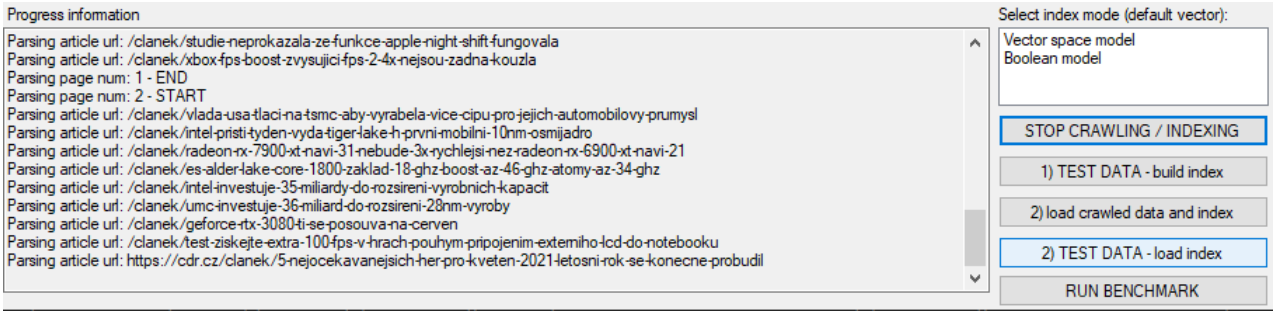
5.3.4 Odstranění uživ. účtu

V okně "Sign in"nejprve vyplňte Vaše přihlašovací údaje, poté klepněte na tlačítko "delete account". Pokud jsou stávající přihlašovací údaje validní, program požádá o potvrzení smazání účtu. V případě potvrzení z Vaší strany dojde ke smazání účtu.

V případě, že původní uživatelské údaje nejsou platné, uvidíte upozornění na tuto skutečnost a smazání účtu nebude možné.

5.3.5 Stažení dat z webu a jejich zaindexování

Po spuštění programu vyberte požadovaný index v "Select index mode"a stiskněte tlačítko s nápisem "1) crawl data and build index". Program začne procházet zvolený web a průběžně ukládat získané výsledky do souborů ve složce "downloaded_data". Průběh je možno vidět v okně "Progress information"(viz 5.1).



Obrázek 5.1: Získávání dat webu

Získávání informací lze kdykoliv zastavit stiskem tlačítka "STOP CRAWLING / INDEXING", čímž dojde k přerušení crawlingu a získaná data budou zaindexována.

5.3.6 Načtení indexu dat získaných z webu

Stiskněte tlačítko s nápisem "load crawled data and index". Před načtením indexu je samozřejmě nutné, aby byl index vytvořený. V případě, že index není nalezen, zobrazí se upozornění.

5.3.7 Zaindexování testovacích dat

Vyberte požadovaný index v "Select index mode"a stiskněte "1) TEST DATA - build index". Dojde k vytvoření indexu s testovacími daty.

5.3.8 Načtení indexu testovacích dat

Stiskněte tlačítko "2) TEST DATA - load index"a sledujte průběh v "Progress information"okně.

5.3.9 Spuštění benchmarku

UPOZORNĚNÍ: před spuštěním benchmarku je nutné načíst index s testovacími daty, režim vector space model!

Po načtení odpovídajícího indexu je proveden benchmark, který vygeneruje ve složce s programem txt soubor s názvem "res_DRTINA.txt".

5.3.10 Vyhledávání

UPOZORNĚNÍ: pokud chcete vyhledávat pomocí vektorového i booleovské vyhledávání, je nutné načíst index vytvořený v režimu vector space model. Pokud je načten pouze booleovský index, je umožněno pouze booleovské hledání (index neobsahuje váhy).

Zadejte požadovaný výraz do pole pro vyhledávání, jež se nachází v horní části programu. Následně vyberte v záložce "Search mode:"požadovaný způsob vyhledávání. Proces zahájíte klepnutím na tlačítko "search!"