Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Automatizované stažení a analýza webových stránek

Diplomová práce

Autor práce: Bc. Marek Laušman

Studijní obor: Aplikovaná Informatika

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.

Hradec Králové 2022

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury

.....

Marek Laušman

29. dubna 2022

Poděkování

Děkuji vedoucí diplomové práce Mgr. Daniele Ponce, Ph.D. za metodické vedení práce.

Anotace

Tato diplomová práce je zaměřená na porovnávání značkovacích zvyklostí webových stránek získaných automatickým stažením z webového archivu webových stránek Archive.org. Práce je rozdělena na pět hlavních částí. První se zabývá kategoriemi získávaných informací z webových stránek. Ve druhé části jsou popsány informace a jejich automatizované získávání z webových stránek. Třetí část se zabývá návrhem praktické části - tzn. návrh použitých technologií pro implementaci webového scraperu a stanovení výzkumných otázek. Čtvrtou částí je samotná implementace webového scraperu. Pátá část se zabývá aplikací webového scraperu na archiv webových stránek Archive.org a následným výzkumem webových stránek z hlediska HTML elementů důležitých pro SEO, přístupnost webových stránek a částečně i sémantiky webů.

Anotation

Title: Automated website download and analysis

This diploma thesis is focused on comparing the tagging habits of websites obtained by automatic download from the web archive - Archive.org. The thesis is divided into five main parts. First one deals with the possible categories of information obtained from websites. The second part describes the information and its automated retrieval from websites. The third part deals with the design of the practical part - ie. the design of used technologies for web scraper implementation and determination of research questions. The fourth part is the implementation of the web scraper. The fifth part deals with the application of a web scraper to the Archive.org website archive and the subsequent research of the website in terms of HTML elements important for SEO, website accessibility, and partly also the semantics of websites.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Cílem této diplomové práce je uvést případného čtenáře do problematiky webového scrapingu za představení podpůrných technologií, které jsou dále použité pro demonstraci práce na diplomovém webovém scraperu. Dále se práce zaměřuje na kategorie získávaných informací z webu a na následnou analýzu a vyhodnocení získaných informací z webových stránek v závislosti na vývoji v čase. Práce se zaměřuje na popis a obecný vývoj webového scraperu a následně na vývoj webového scraperu v programovacím jazyce Python, který byl zvolen jako vhodný kandidát pro vývoj daného scraperu. Práce je členěna na pět hlavních kapitol, které spolu navzájem úzce souvisí. První kapitola se zaměřuje na možné kategorie získávaných informací z webových stránek. Konkrétně se jedná o popis optimalizace SEO, přístupnost webových stránek a sémantiky webu. Druhá kapitola se zabývá vysvětlením informace a obecným získáváním dat z webových stránek na internetu. Třetí část se věnuje převážně praktické části této diplomové práce, jejíž hlavním úkolem je navrhnout konkrétní návrh webového scraperu a srovnání použitelných technologií pro implementaci webového scraperu podobného typu, a přípravu výzkumných otázek. Ve čtvrté kapitole je řešena samotná implementace webového scraperu a princip ukládání dat z webového archivu webových stránek Archive.org. Poslední kapitola se zabývá aplikací vytvořeného webového scraperu a následnou analýzou získaných dat.

Existuje spousta již vytvořených nástrojů pro scraping, dokonce i bez napsání jediného příkazu v kódu, nicméně bylo rozhodnuto, že scraper musí být vyvinut přímo na míru kvůli požadovaným specifickým HTML značkám, které mají být získány. Scraper tedy bude vyvinut ve frameworku (v rámci) Scrapy především díky zkušenosti autora s tímto framworkem. Strukturovaná data budou ukládána do databáze PostgreSQL.

2 Cíl práce, metodika

2.1 Cíl práce

Cílem této diplomové práce je charakterizovat značkovací zvyklosti vzhledem k značkovacím doporučením s využitím automatizovaného stahování webových stránek. Automatizovaným stahováním webových stránek se rozumí scraper napsaný v programovacím jazyce Python za použití frameworku Scrapy, který bude procházet postupně jednu URL po druhé a získávat požadované informace z nich. Výsledkem této práce by tedy měl být návrh a implementace scraperu, porovnání technologií sloužících pro vývoj scraperu, popsání scrapingu obecně, popsání optimalizace SEO, přístupnosti webových stránek, sémantického webu a výsledné analýzy získaných dat z určitých webových stránek.

2.2 Metodika

Pro účely této diplomové práce bude naprogramován webový scraper v programovacím jazyce Python a konkrétním frameworku Scrapy, který bude procházet několik desítek vybraných domén (celkově přibližně 31 000 záznamů webových stránek), z webového archivu webových stránek Archive.org, postupně stránku po stránce. Z webů budou získávána taková data, která úzce souvisí s optimalizací webových stránek SEO, přístupností stránek a i částečně sémantikou webových stránek. Získaná strukturovaná data budou postupně ukládána do databáze PostgreSQL do dvou tabulek.

Následně bude nutné stanovit si výzkumné scénáře a poté vytvořit konečnou analýzu, která se dané scénáře pokusí demonstrovat. Výzkumné otázky budou tedy vymýšleny tak, aby z jejich analýzy bylo možné vyčíst vývoj jednotlivých HTML elementů nebo přístupů v čase, popřípadě aby bylo možné vysledovat specifické trendy.

3 Kategorie získávaných informací z webových stránek

Jak již bylo zmíněno v úvodu, tato práce se zabývá primárně scrapováním vybraných informací z webových stránek. Předmětem této práce není práce s datovými sady ani tzv. OpenDaty.

3.1 Přístupnost webových stránek

Přístupnost webových stránek v jednoduchosti je soubor norem, které zajišťují bezproblémové používání webové stránky bez ohledu na rozdílný sotfware, hardware, nebo zdravotní stav uživatele. Typicky se jedná například o nevidomého člověka, kterému obsah webových stránek předčítá zařízení, na kterém se webová stránka zobrazila. U textu je to jednoduché, nicméně například u obrázků je nutné správně specifikovat atribut *alt*, který bude danému uživateli přečten. V dnešní době je přístupnost webu velmi důležitá, jelikož každý uživatel používá jiný hardware. Tento problém je řešitelný pomocí responzivity webu.
Přístupnost webových stránek závisí na jednotlivých komponentách, které spolu ale nějakým způsobem souvisí a navzájem se podporují. Mezi tyto komponenty patří například [5]:

- Obsah webu - cokoliv, co se na webových stránkách nachází (například veškeré texty, obrázky, skripty, atd.).
- Uživatelské agenty - software používaný lidmi pro přístup k veškerému obsahu webových stránek, včetně multimediálních přehrávačů, prohlížečů mobilních telefonů, hlasových prohlížečů, atd..
- Nástroje autorů - software, který autoři používají k vytvoření webového obsahu, včetně všech editorů kódu, blogů, atd.

Normy hrají poněkud zásadní roli ve vývoji webových stránek v dnešní rychlé době. Některé požadavky na přístupnost jsou složitější na implementaci, ale velkou část požadavků dokáže splnit úplně každý. Jak již bylo zmíněno dříve, primárně jde o responzibilitu a o jednoduchost ovládání uživatelem s nějakým zdravotním znevýhodněním [5].

3.1.1 Příklady přístupnosti webových stránek

Alternativní text obrázku

Nejznámější a nejjednodušší metodou zlepšení přístupnosti webových stránek je vyplnit alternativní popisek u obrázku pomocí atributu *alt* v HTML značce *<img>*. Lidé se zrakovým postižením si tak mohou poslechnout přes asistenční zařízení, co se na obrázku nachází a lépe si tak představit obsah webových stránek. Alternativní text je tedy nutné vyplnit tak, aby dokonale popisoval, co se nachází na daném obrázku.

Alternativní text audia

Alternativní text se překvapivě nenachází pouze u obrázků, ale také u zvukových souborů. Lidé se sluchovým postižením si mohou přečíst veškerý obsah zvukového souboru. Existuje mnoho služeb, které přesně tento problém řeší, jako například 3playmedia.com.

Ostatní

Mezi další požadavky přístupnosti patří například správný barevný kontrast v grafice webových stránek. Dále možnost plnohodnotného používání webové stránky i bez myši, jako například pomocí klávesnice. Jako další požadavek může být správné vyplnění titulku stránky.

3.2 Search Engine Optimization (SEO)

Search Engine Optimization (SEO - optimalizace webu pro vyhledávače) znamená proces zlepšování viditelnosti webových stránek ve známých vyhledávačích, jako jsou Google, Seznam, Yahoo a podobně. Obecně platí, že čím lépe jsou webové stránky optimalizovány, tím spíše na dané stránky zákazník narazí a nakoupí nebo poptá daný produkt/službu. V jednoduchosti by se tedy dalo říct, že SEO slouží jako marketingový nástroj pro přilákání nových potenciálních zákazníků na webové stránky.

3.2.1 Jak SEO funguje

Pro lepší pochopení, jak celkově SEO funguje, je moudřejší si nejprve říci něco o fungování vyhledávačů jako takových.

Google vyhledávač a jemu podobné víceméně fungují na bázi čtyř základních mechanismů [6]:

1. Objevování - neboli hledání webových stránek je zajištěno pomocí softwarového robota, kterému se odborněji říká bot, webbot nebo robot.

2. Ukládání - ukládání adres, souhrnů stránek a souvisejících důležitých informací. Servery, na kterých jsou tato data ukládána se nazývají indexové servery.

3. Hodnocení - pro seřazení uložených stránek podle důležitosti, k tomu Google používá mechanismus PageRank [7].

4. Zobrazení výsledku - po zadání vyhledávacího dotazu se vrátí stránky seřazené podle relevantnosti hledaných klíčových slov.

Jak již bylo zmíněno výše, optimalizace pro vyhledávače je sama o sobě neplacená činnost, která má za cíl přivést co nejvíce uživatelů na webové stránky pomocí organické cesty. Opakem organické cesty jsou placené příspěvky PPC (pay-per-click), které jsou označeny štítkem "reklama"a nachází se na prvních třech a posledních třech pozicích ve výsledcích vyhledávání, tudíž mají tendenci mít větší tzv. proklikovost než organické výsledky vyhledávání. V jednoduchosti, každý inzerent platí za každý proklik reklamou na webové stránky.

3.2.2 Techniky SEO optimalizace

Dle [8] neexistuje příliš mnoho odborných prací zabývajících se přímo technikami SEO obecně. Dostatečným zdrojem informací by ale mohly být společnosti zabývající se internetovým marketingem, které většinou na webových stránkách provozují svůj blog, kde sdílí zkušenosti se SEO optimalizací přímo z praxe. Mezi hlavní techniky SEO tedy patří výzkum klíčových slov, indexování, on-site optimalizace a off-site optimalizace.

3.2.3 Zjištění klíčových slov

Při tvorbě textů na web a webu jako takového je nutné myslet na hlavní klíčová slova spojená s předmětem webových stránek. Klíčová slova totiž do jisté míry určují relevantnost webové stránky pro zobrazení na daný vyhledávací dotaz ve vyhledávači. Typicky je doporučeno používat klíčová slova v rozumné míře napříč celými webovými stránkami. Nejen do textu, ale i do kódu webových stránek pomůže vyhledávači najít danou webovou stránku. Typicky se klíčová slova píší do titulku, url adresy, nadpisů, do alternativních textů obrázků, do meta popisku a do mnoha dalších.

3.2.4 Indexovatelnost

Indexovatelnost je nezbytná pro uložení webových stránek do indexu prohledávacím robotem vyhledávače a tedy pro zobrazení ve výsledcích vyhledávání. Existuje několik online nástrojů, které zanalyzují kompletně webové stránky a jsou schopni definovat, proč je a proč není webová stránka indexovatelná. Typicky se jedná o správné vygenerování sitemap.xml souboru a souboru robots.txt (který bude podrobněji popsán v následující kapitole).

3.2.5 On-site optimalizace

Jak již název napovídá on-site optimalizace (optimalizace na webových stránkách) se zabývá optimalizací přímo webových stránek jako takových. Jedná se tedy o správné vyplnění titulků, meta popisků, nadpisů úrovně H1, alternativních popisků vložených obrázků atd.

3.2.6 Off-site optimalizace

Off-site optimalizace zase napovídá optimalizaci pro vyhledávače mimo webové stránky, tzn. bez úprav webových stránek jako takových. Mezi hlavní a nejefektivnější taktiky off-page optimalizace patří [9]:

- Budování zpětných odkazů (link building) - jednou z hlavních taktik off-size optimalizace je tzv. linkbuilding (budování zpětných odkazů), který je metodou budování zpětných odkazů. V praxi to znamená například zmiňování daných webových stránek na různých fórech, budování podpůrných stránek (microsites), výměna odkazů s jinými weby nebo aktivita na sociálních sítích.

- Budování značky (brand building) - budování značky společnosti zároveň zvyšuje i důvěryhodnost společnosti, která je brána v potaz při vyhledávání na Google.

- Obsahový marketing (content marketing) - obsahový marketing patří jak do on-page optimalizace, tak zároveň do off-page optimalizace. Je to přístup k tvoření obsahu na webu i mimo něj. Typicky se jedná o příspěvky do blogu, e-knihy nebo do různých specifických průzkumů.

- Sociální sítě - budování sociálních sítí není faktor přímo ovlivňující pozici při vyhledávání, nicméně i tak je důležitým marketingovým kanálem. Jednak na-

pomáhají k budování značky a druhak se profil na sociálních sítích může objevit po zadání vyhledávacího dotazu i na Google, což ve finále bude teoreticky vést k navštívení webových stránek.

- Influencer marketing - víceméně vše v off-page optimalizaci souvisí s již zmíněným budováním značky nebo budováním zpětných odkazů. Influencer marketingem se tedy zvyšuje povědomí o dané společnosti a zároveň se tím budují zpětné odkazy, přes které uživatelé sledující daného influencera navštivují webové stránky.

- Recenze - Recenze na Google na danou společnost jsou jedním z hlavních faktorů rozhodujících o navštívení webových stránek a potenciální koupi produktu.

3.2.7 HTML značky z pohledu SEO

Titulek webové stránky

Titulkem webové stránky se rozumí HTML značka <title>, která se nachází v hlavičce <head> dané webové stránky.

```
<head>
    <title>
        Staveko Trade - Luxusni hlinikove pergoly a stinici technika
    </title>
</head>
```

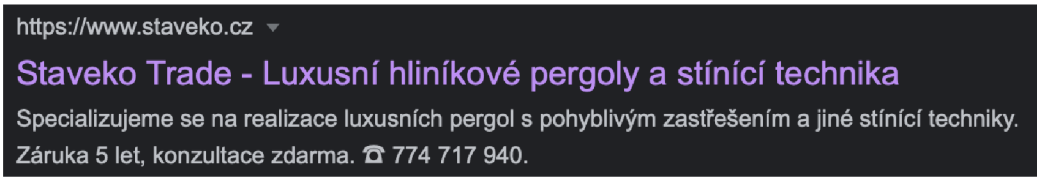
Zdrojový kód 1: Ukázka <title> tagu v hlavičce webových stránek. Zdroj [Autor]

Správně vyplněný titulek, který zachovává jistá dopoučení a pravidla. Je velmi důležitý zejména z pohledu optimalizace pro vyhledávače SEO, která je mnohem detailněji popsána několik řádků zpět. Řadí se mezi nejdůležitější on-page SEO prvky a přitom jeho správné vyplnění je velmi triviální. Titulek stránky se může zobrazit celkem na třech umístěních. Typicky se jedná o:

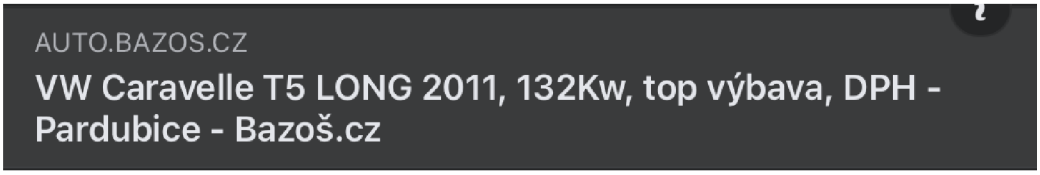
- Nadpisy ve výsledcích vyhledávání - viz. obrázek 1

- Odkazy z externích stránek, například na sociálních sítích - titulek se propisuje do textové části odkazu - viz. obrázek 2

- Text v záložce stránky ve webovém prohlížeči - viz. obrázek 3



Obrázek 1: Ukázka titulku ve výsledcích vyhledávání. Zdroj [Autor]



Obrázek 2: Ukázka titulku na odkazu na sociální síti Facebook. Zdroj [Autor]

Obrázek 3: Ukázka titulku na kartě záložky v prohlížeči. Zdroj [Autor]

Popisek webové stránky

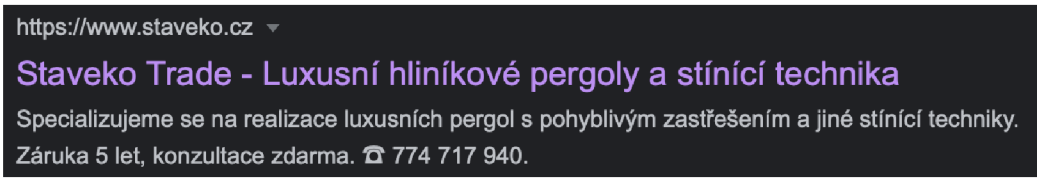
Druhou velmi důležitou HTML značkou pro optimalizaci webových stránek je značka *<meta name="description»*, která se stejně jako <title> nachází v hlavičce HTML dokumentu a používá se k vložení popisu dané webové stránky. Konkrétně tedy patří do rodiny *<meta>* tagů s konkrétním atributem name obsahující textový řetězec *"description"*.

```
<head>
    <meta name="description"
    content="Specializujeme se na realizace luxusnch pergol
    s pohyblivym zastresenim a jin stinici techniky.
    Zaruka 5 let, konzultace zdarma.  774 717 940">
</head>
```

Zdrojový kód 2: Ukázka popisku na webových stránkách. Zdroj [Autor]

Značka jako taková nemá od roku 2009, dle Googlu, přímý vliv na pozici ve výsledcích vyhledávání, nicméně stále je velmi důležitá. Ve výsledcích vyhledávání (SERP) se totiž zobrazuje popisek jako takové lákadlo na danou webovou stránku. Ideální rozmezí délky popisku je 150-160 znaků a měl by obsahovat nejdůležitější popis dané

webové stránky, respektive čím se určitá stránka zabývá. Dále by měl stejně jako u titulku obsahovat nejdůležitější klíčová slova dané stránky, konkurenční výhody a nějaké CTA (call to action - výzva k akci) fráze [10]. Stejně jako u titulku, pokud je popisek delší než 160 znaků, bude z nějaké části odstřihnut, čemuž je většinou nejlepší se vyhnout. Každé jednotlivé stránce či podstránce je doporučeno použít jiný popisek, který dokonale vystihuje danou stránku. Na obrázku č. 4 níže, je popisek bílý text umístěný pod fialovým titulkem ve spodní polovině obrázku.



Obrázek 4: Ukázka popisku ve výsledcích vyhledávání. Zdroj [Autor]

Hlavní nadpisy webové stránky

Obecně nadpisy <h1> až <h6> jsou velmi důležitým prvkem každé webové stránky. Konkrétně mají určitý vliv na SEO optimalizaci a zejména strukturu a hierarchii obsahu webových stránek. Nadpisy se nachází v těle <body> HTML dokumentu. Jednotlivé nadpisy pomáhají pochopit obsáhlejší a komplexnější dokumenty. To samé platí pro crawlery, které prochází jednotlivé webové stránky a zařazují je do své databáze. Čím nižší číslo v HTML elementu <h...>, tím důležitější a větší nadpis. Jednotlivá čísla tedy určují velikost v hierarchii dokumentu. Nejdůležitějším faktorem při psaní správných nadpisů na webových stránkách je, stejně jako u titulku a popisku, použití hlavních klíčových slov dané stránky. Ukázka č. 3 zobrazuje ukázku jednoduchých nadpisů v HTML kódu:

```
<body>
    <h1>Nadpis 1. urovne</h1>
    <h2>Nadpis 2. urovne</h2>
    <h3>Nadpis 3. urovne</h3>
    <h4>Nadpis 4. urovne</h4>
    <h5>Nadpis 5. urovne</h5>
    <h6>Nadpis 6. urovne</h6>
</body>
```

Zdrojový kód 3: Ukázka nadpisů webových stránek. Zdroj [Autor]

Obrázky na webových stránkách

Je obecně známo, že obrázky jsou velmi důležitými prvky na webových stránkách. Na pomyslné škále důležitosti prvků na webu jsou ihned za texty, jelikož obrázek často dokáže přilákat více lidí než text a zároveň je schopen i prodat produkt. Dle [11] Google obecně hodnotí lépe obsah webových stránek obsahující obrázky než pouhý text. U obrázků je hodnoceno mnoho kritérií, mezi které patří:

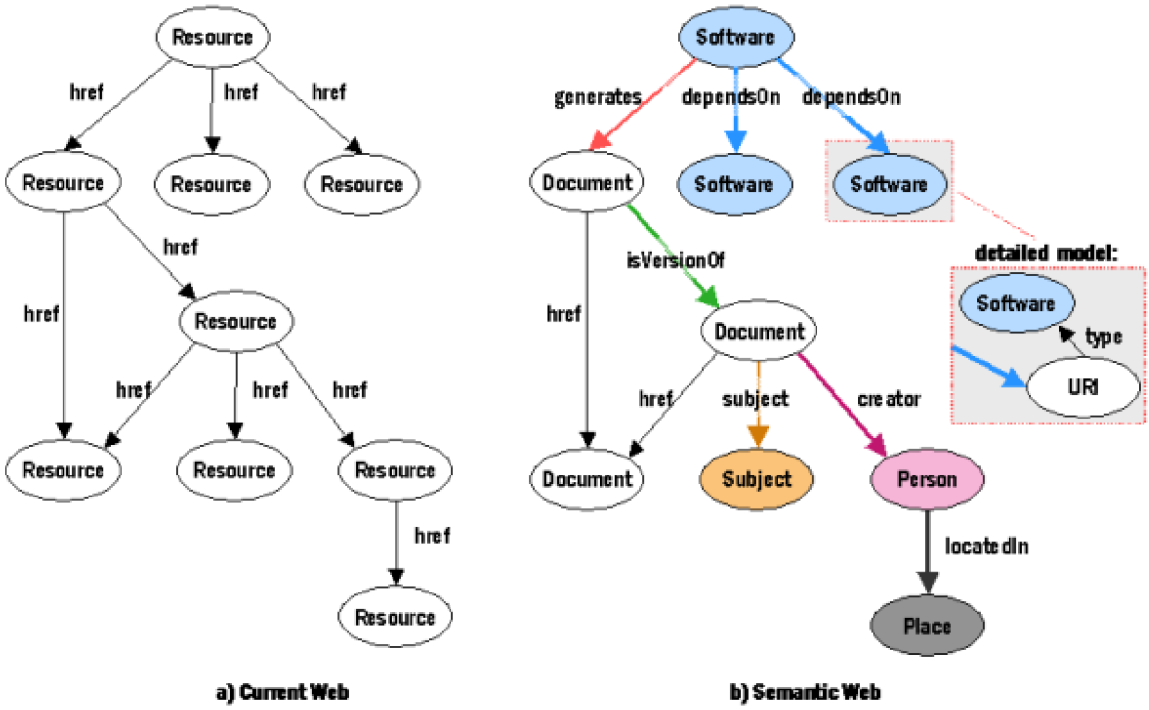
- Faktory přímo na obrázku (on-picture)
  - Velikost obrázku
  - Formát obrázku
  - Název obrázku
  - Struktura URL
- Okolí a metadata
  - Pozice
  - Titulek
  - Alt
- Propojení
  - Odkazování na obrázky
  - Zpětné odkazy

3.3 Sémantický web

Definice sémantického webu: „Sémantický web není samostatný web, ale pouze rozšíření již existujícího webu, ve kterém mají informace přesně definovaný význam a lepší umožnění spolupráce počítačů a lidí.“ [12]

Sémantický web je tvořen a strukturován podle předem stanovených pravidel, což umožňuje mnohem efektivnější vyhledávání informací na internetu. Při realizaci sémantického webu je nutné implementovat standardy RDF (sémantická složka architektury), XML (strukturální složka architektury) a URI (syntaktická složka architektury).

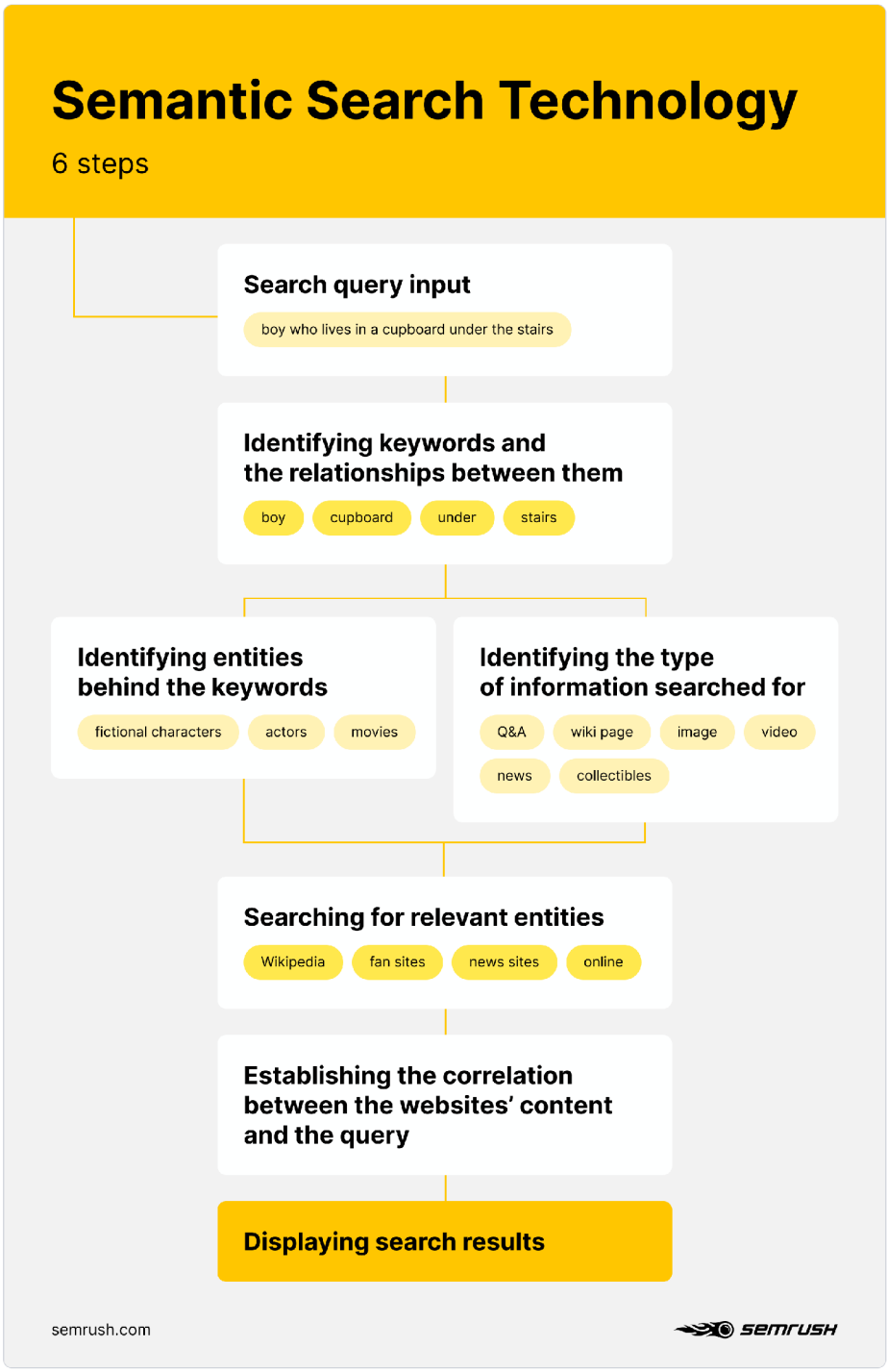
Na obrázku č. 5 je vidět rozdíl mezi klasickými webovými stránkami a sémanticky optimalizovanými webovými stránkami.



Obrázek 5: Rozdíl mezi klasickým a sémantickým webem. Zdroj [[1]]

3.3.1 Sémantické vyhledávání

Sémantické vyhledávání (semantic search) znamená vyhledávání s konkrétním významem, ne pouze lexikální vyhledávání, kde vyhledávač vrací výsledky vyhledávání pouze podle shody hledaných slov. Ideální příklad může být vyhledávací dotaz: *kluk, který žije ve skříni pod schody*. Google vrátí výsledky, které se týkají Harryho Pottera - konkrétní filmové postavy, která jednu dobu opravdu žila ve skříni pod schody [2]. Fungování sémenatického vyhledávání je vyobrazeno na obrázku č. 6 níže.



Obrázek 6: Technologie sémantického vyhledávání. Zdroj [2]

Nejnovější aktualizací v oblasti sémantického vyhledávání je technologie BERT (Bidirectional Encoder Representations from Transformers). BERT je schopen trénovat a zlepšovat svůj nejmodernější systém hledání nejlepších výsledků na dané vyhledávání. Funguje na chápání celého kontextu vyhledávacího dotazu [2].

V současnosti je trochu komplikovanější určit správná klíčová slova na webových stránkách. Na klíčová slova je nutné se dívat ve smyslu celého rozsahu kontextu ne pouze na jednotlivá slova.

3.3.2 Optimalizace pro sémantické vyhledávání

Existuje několik technik, které stejně jako v klasické optimalizaci SEO, zásadně ovlivňují sémantické vyhledávání, což má velký vliv na naleznutí všech webových stránek. Níže je uvedený krátký přehled základních technik optimalizace pro sémantické vyhledávání [2]:

- Naleznutí klíčových slov ve smyslu kontextu - jak již bylo zmíněno výše, je nutné najít taková klíčová slova, která odpovídají tématu webových stránek, ne pouze jednotlivým nesouvisejícím klíčovým slovům. Ideálně pro jiná klíčová slova stejného významu vytvořit separátní stránky, které daná klíčová slova pokryjí.

- Pochopení vyhledávacího dotazu - pokud je například vyhledávacím dotazem sousloví: apple or xiaomi, Google v mnoha případech nabídne jako první informační články, které srovnávají mobily apple a xiaomi. Až následně nabídne nákup daných mobilů.

- Použití sémantických HTML značek - je vhodné používat takové značky, které indikují obsah. To znamená například <article>, <header> nebo <footer>. Z těchto značek je ihned vidět, co za obsah se v nich bude v nejvyšší pravděpodobností nacházet.

- Použití strukturovaných dat - Google a některé další vyhledávače rozumí strukturovaným datům jako například Schema.org.

- Využití znalostních grafů (Knowledge Graph) - znalostní graf je síť sémantických metadat, která interpretuje kolekci souvisejících entit. Znalostní graf je lépe popsán v zde [13].

3.3.3 ARIA role

ARIA role úzce souvisí se sémantickým webem jako takovým. Poskytují totiž sémantický význam různým HTML tagům, díky čemuž je poté možné číst obsah webu pomocí čteček obrazovky. Obecně je možné role ARIA použít u HTML prvků, které

existují bez plné podpory prohlížeče, nebo neexistují vůbec. Díky specifikaci role bude s objektem nakládáno tak, jak by se od objektu daného typu/role očekávalo. Jak již bylo zmíněno dříve, HTML disponuje některými sémantickými elementy, jako například <input> nebo <footer>. Většina těchto sémantických elementů obsahuje definici role ve výchozím stavu. To znamená, že například <input> disponuje atributem "type", kde lze specifikovat jaký typ vstupu to bude - například textový vstup nebo radio tlačítko. U elementů, které se neřadí mezi sémantické není možné defaultně specifikovat jejich typ. Právě k tomu slouží popisované ARIA role, díky kterým lze přidělit konkrétní roli ne-sémantickému elementu. V HTML značce je potřeba specifikovat atribut *role="nejaka-role"* [14].
Existuje 6 základních kategorií ARIA rolí, mezi které patří:

- Role struktury dokumentu - používají se k popisu struktury obsahu. Tyto role ale není doporučeno používat, jelikož již existují konkrétní HTML sémantické elementy, které tuto roli přesně plní. Mezi některé základní role patří:
  - toolbar
  - tooltip
  - feed
- Role widgetů - používají se k definici interaktivních vzorů u elementů. Některé role zase není doporučeno používat, jelikož HTML disponuje sémantickými elementy, které tuto problematiku řeší. Mezi některé základní role patří:
  - scrollbar
  - slider
  - tabpanel
- Role orientačních bodů - podobně jako role struktury dokumentů se používají k identifikace organizace a struktury dokumentu. Čtečky obsahu tyto role používají k získání navigace pomocí klávesnice k různým částem webové stránky. Přílišné použití těchto rolí vede k vytvoření "šumu"u čteček obrazovky.
  - banner
  - form
  - main

- Živé regionální role - role používané k definici obsahu, který se bude dynamicky měnit v čase.
  - alert
  - log
  - status
- Okenní role - role používané k definici sekundárních oken nad hlavním oknem webové stránky.
  - dialog
  - alertdialog
- Abstraktní role - role používané k efektivnější organizaci dokumentu.

4 Informace a jejich automatizované získávání z webových stránek

Tato kapitola se bude zabývat definicí informace, technikami získávání informací z internetu, samotnou problematikou webového scrapingu a značkovacími jazyky.

4.1 Co je informace

Informace je v jednoduchosti určité vědění o někom nebo něčem, které má specifický význam pro příjemce dané informace. Data jsou definována jako soubor faktů nebo statistik a mohou mít podobu například textu, čísel, obrázků, grafů. Oproti informaci jsou data neuspořádaná a dělí se primárně na 2 hlavní typy:

- Kvantitativní data - číselný údaj (příkladem je cena vybraného produktu, hmotnost balíku, atd.)
- Kvalitativní data - popisná data v textové formě (příkladem je jméno a příjmení, adresa bydliště, atd.)

4.2 Získávání informací z internetu

Tato práce je zaměřená z velké části na získávání informací z webových stránek na internetu. Typicky se jedná o získávání relevantních dat z nespecifikované organizované kolekce dokumentů.

4.2.1 Manuální proces získávání informací

Informace je možné získat několika způsoby. Dříve by bylo dostatečné získat informace například klasickým ručním hledáním a opisováním dat z různých webových stránek, jelikož weby nebo e-shopy nebyly tak obrovské a příliš často se neaktualizovaly. Tato technika by byla v dnešní době velmi neefektivní a příliš časově náročná.

4.2.2 Získávání informací pomocí rozhraní API

Webové stránky mohou disponovat API rozhraním, pomocí kterého jsou uživatelé schopni přistupovat k již strukturovaným datům nacházejících se na daných webových stránkách. Výhodou API rozhraní je korigovatelnost dat, ke kterým uživatelé

přistupují. Lze například poskytnou pouze určitá data, která uživatel bude moct nějakým způsobem využít. Naopak je možné přístup k nějakým datům zakázat nebo zpoplatnit. Nevýhodou API rozhraní může být počet dotazů na dané API, které bude jistě omezené. Možnost více dotazů je potom často zpoplatněná. Obecně je ale získávání dat pomocí API rozhraní nejjednodušší a často nejrychlejší formou získávání dat z internetu.

4.2.3 Webový scraping

Jak již bylo zmíněno dříve, v současnosti je velmi neefektivní a časově náročné data z webových stránek získávat manuálně. Každý, kdo pracuje s velkými daty (tzv. big data), příkladem je například výzkumný pracovník, věděc, webový srovnávač apod., potřebuje získání dat nějakým způsobem zefektivnit a hlavně zrychlit. Potřebují mít získávaná data stále aktuální. K tomu slouží webový scraping. Webový scraping je tedy technika získávání dat z internetu pomocí softwarového robota a uspořádání dat do formátů jako jsou CSV, HTML nebo třeba JSON. Na trhu programovacích jazyků webového scrapingu určitě vede Python, který obsahuje nespočet již vytvořených knihoven, které celý vývoj webového scrapera značně urychlují a zjednodušují. Typickým příkladem použití webového scrapingu může být například získávání tisíců až milionů recenzí uživatelů na vybrané produkty, což by poté mohlo být nápomocné například v marketingu k analýze sentimentu. Dalším příkladem by mohla být například analýza vývoje cen produktů na e-shopech.
Nejefektivnější technikou pro získávání dat z tedy internetu je v dnešní době takzvaný webový scraping. Jak webový scraping, tak weby jako takové, se za posledních několik let značně zlepšily svoji kvalitu. Webový scraping je teď ve fázi, kdy téměř kdokoliv si je schopný vytvořit takového vlastního softwarového robota, který si data získává kontinuálně sám a z kterýchkoliv webů. Ať už za použití různých volně dostupných nástrojů, nebo samotným naprogramováním.

4.3 Soubor robots.txt

Soubor robots.txt je obyčejný textový soubor umístěný na webové stránce na url /robots.txt. Je to soubor obsahující pravidla pro navštěvující webové roboty řídící se standardem Robots Exclusion Standard. Níže v ukázce kódu č. 4 je uvedený příklad sloužící jako demonstrace souboru.

```
User-agent: TestBot
Allow: /admin/

User-agent: *
Disallow: /admin/
Disallow: /external/
Disallow: /modules/
```

Zdrojový kód 4: Ukázka Robots.txt souboru. Zdroj [Autor]

V prvním případě je TestBotovi povoleno navštívit url /admin/. Jak je vidět, v souboru lze specifikovat různá pravidla pro různé roboty. Ve druhém případě soubor robots.txt říká, že je všem ostatním webovým robotům zakázáno navštěvovat url /admin/, /external/ a /modules/.

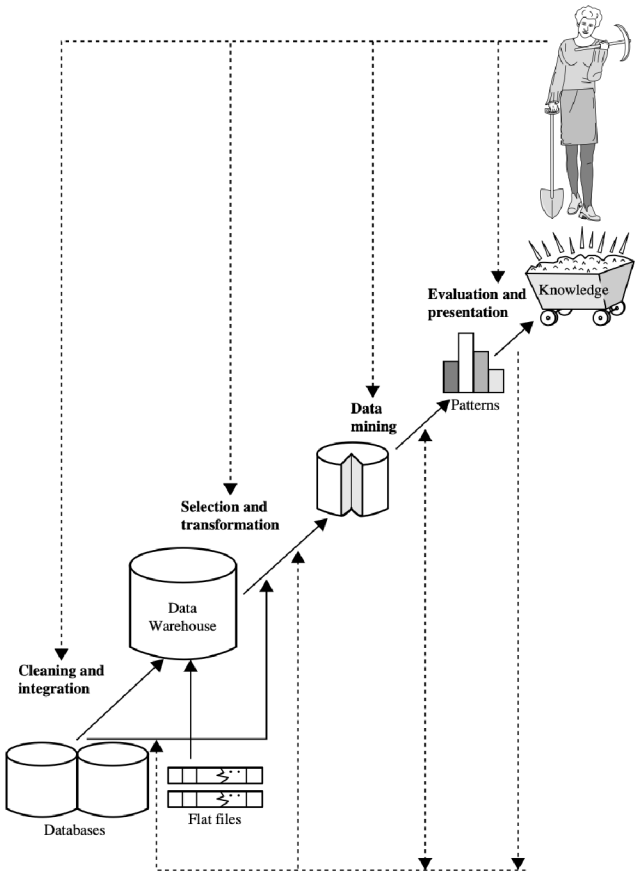
4.4 Data mining

Data mining, dolování vědomostí z dat, je technika systematického prohledávání nějakého textu nebo webové stránky za účelem získání výsledku námi zadaného cíle. Proces získávání vědomostí z dat se skládá z následujících kroků [3]:

1. Čištění dat (odstranění šumu a nekonzistentních dat)
2. Integrace dat (možná kombinace více datových zdrojů)
3. Výběr dat (z databáze se získávají relevantní data k analýze)
4. Transformace dat (data jsou transofrmována a konsolidována do forem vhodných pro mining souhrnných nebo agregačních operací)
5. Dolování dat (proces, kde se používají inteligentní metody pro získání datových vzorů)
6. Ohodnocení vzorů (identifikace skutečně zajímavých vzorců reprezentující vědomosti)
7. Prezentace znalostí (vizualizace a reprezentace získaných znalostí)

První čtyři kroky jsou pouze formy přípravy dat pro dolování dat. Pátý krok, samotné dolování dat, je proces objevování zajímavých vzorců a znalostí z velkého množství

předem určených dat. Zdroje dat mohou být nějaké databáze, webové stránky, datové sklady, úložiště informací a podobně. Poslední dva kroky jsou vyhodnocení a prezentace dat a znalostí.



Obrázek 7: Proces získávání dat. Zdroj [3]

Pro dolování dat neexistuje víceméně žádné omezení ve smyslu toho, jaká data mohou být dolována. Jak již bylo zmíněno výše, dolovány mohou být nejrůznější databáze, datové sklady a transakční data. Dalšími druhy dolovaných dat jsou například sekvenční data, data závislá na čase (například historická data), inženýrská designová data (návrhy budov), odkazy, multimediální data a mnoho dalších. Každý druh dat s sebou nese různé výzvy, které je nutné brát v potaz a které je nutné řešit. Například je nutné vyřešit, jak pracovat se speciálními strukturami dat, které představují stromy, grafy atd. [3]

4.4.1 Jak webový scraping funguje

Každá webová stránka je napsána ve zdrojovém kódu HTML, což je možné vidět přes průzkumníka ve webových prohlížečích jako Google Chrome nebo Safari. HTML je založené na skládání různých elementů, které plní různé funkce. Pomocí technologie XPath a CSS selektorů je poté možné se k HTML elementům na dané webové stránce dostat a získat jejich obsah. Poté už je potřeba jen technologie, která získávání potřebných dat zautomatizuje. Konkrétně se jedná o softwarového bota, který na základě instrukcí daných programátorem prochází vybrané webové stránky jednu po druhé, analyzuje a poté stahuje jejich obsah, který poté ukládá například do databáze. V jednoduchosti je při vytváření vlastního softwarového robota nutné správně zvolit vstupní adresy URL, vybrat správné cesty k HTML elementům a zvolit vhodnou taktiku ukládání stahovaných dat.

4.4.2 Technologie XPath

XPath je počítačový dotazovací jazyk sloužící k adresování částí XML dokumentů. Použitím jazyka XPath je možné vybírat konkrétní jednotlivé části XML dokumentu a nějakým způsobem s nimi pracovat [15].

Dotaz může vypadat takto:

```
//*[@id="some_id"]/nav[1]/ul[2]/li[3]/a/span/text()
```

Zdrojový kód 5: Ukázka použití technologie XPath. Zdroj [Autor]

4.4.3 CSS selektor

Jedná se o obyčejný vzor elementů a dalších výrazů, které prohlížeči sdělují, kde se HTML prvky nacházejí a které prvky by měly být vybrány.
Typicky css selektor může vypadat například takhle:

```
ol li h2 a::text
```

Zdrojový kód 6: Ukázka použití CSS selektoru. Zdroj [Autor]

4.5 Legalita scrapování dat

Ačkoliv existuje nespočet různých nástrojů a metod pro automatizované získávání dat z internetu, které již dlouhé roky pomáhají mnoha lidem v jejich práci, legalita webového scrapingu je v právní oblasti stále "šedou oblastí". V právu neexistuje konkrétně žádný zákon, který by webový scraping přesně definoval a řešil. Je tedy nutné se při webovém scrapingu řídit základními právními teoriemi a zákony, jako je například porušení autorských práv, porušení smlouvy nebo porušení movitých věcí.
Obecně jakákoliv data získaná jakoukoliv nezákonnou činností se řídí několika zákony. Například osoba, která přistupuje pomocí softwarového robota k datům, která jsou chráněná a důvěrná, může být stíhána podle zákona o počítačových podvodech a zneužívání, ale pouze pokud je škoda vyšší než 5 000 amerických dolarů. Typicky se jedná o přistupování k placenému obsahu a následné "přeprodávání"získaného placeného obsahu a nerespektování upozornění od majitele dané webové stránky [16].

4.5.1 Podmínky použití

V právní oblasti je definováno, že vlastník webových stránek může zabránit automatickému stahování dat z jeho webových stránek, pokud to výslovně zakáže v podmínkách použití dané webové stránky. Nedodržení těchto podmínek může tedy vést k již zmiňovanému porušení smlouvy. Pro účinnou vymahatelnost je ale nutné tyto podmínky vyloženě odsouhlasit. Pokud tedy nastavené podmínky uživatel, který chce daná data stahovat, nepotvrdí, například kliknutím na nějaké tlačítko, není možné ho za to žádným způsobem stíhat. Z toho vyplývá, že tedy pouhé oznámení o vysloveném zákazu podmínek použití nemusí nic znamenat [16].

4.5.2 Porušení autorských práv

Obecně je známo, že se získanými daty si uživatel může víceméně dělat co se mu zlíbí, kromě opětovné publikace získaných dat nebo následného zpeněžení těchto dat. To by z právního hlediska mohlo vést k porušení autorských práv. Musí se ale vysloveně jednat o data, na které vlastník webu vlastní autorská práva, jelikož například recenze na produkty od návštěvníků e-shopu není možné nijak vlastnit. To samé platí pro nápady lze vlastnit pouze konkrétní interpretaci těchto nápadů/myšlenek, nikoliv však nápad jako takový [16].

4.5.3 Poškození webu

Webový scraping funguje na bázi posílání velkého množství požadavků (requestů) na dané URL adresy. To může obecně webové stránky ve velké míře zpomalovat nebo dokonce shodit. Nicméně poškození musí být materiální a snadně dokazatelné, jinak je tato situace právně nevyhmatatelná [16].

4.6 Etika webového scrapingu

Existuje spoustu úhlů pohledu na problematiku a etiku webového scrapingu. Kromě nezákonných činností může webový scraping vést i k nechtěnému poškození webu, majitele nebo zákazníků webu.
Příkladem by mohla být sitauce, kdy automatické získávání dat z webu může například vést k nechtěnnému odhalení obchodního tajemství dané webové stránky nebo společnosti. Příkladem může být automatické procházení webových stránek zaměřených na pracovní inzeráty, kdy průběžným počítáním inzerátů je teoreticky možné odhadnout podíl a výnosy webu. To může vést k finanční ztrátě majitele webu.
Dalším příkladem by mohla být situace, kdy například automatické získávání dat a následné porovnávání dat s jinými zdroji může vést k neúmyslnému odhalení identity těch, kteří dataa vytvořili. To může právně vést k situaci, kdy zákazníci webových stránek nemusí souhlasit s použitím jejich dat s třetí stranou. Tato porušení ochrany soukromí a práv poté mohou vést k takovým důsledkům pro majitele webu, kdy se návštěvníci bojí například o svoje online soukromí [16].

4.7 Scraping Single Page a Multi Page aplikací

Přístup k webovému scrapingu SPA a MPA aplikací se poměrně hodně liší. Není možné scrapovat SPA stejným způsobem jako MPA. Následující dvě podkapitoly popisují vícestránkové a jednostránkové aplikace a jejich rozdíly v přístupech webového scrapingu.

4.7.1 Multi Page Aplikace

Multi Page Application (MPA) je vícestránková webová aplikace, která funguje na principu nového vykreslování stránky pokaždé, nastane-li nějaká změna. Změnou je chápáno nějaké přesunutí dat na server. Typickým příkladem vícestránkové aplikace jsou obrovské e-shopy, příkladem je eBay nebo Amazon, které potřebují například

víceúrovňové menu nebo potřebují jednoduše více stránek [17].

Scraping vícestránkové aplikace

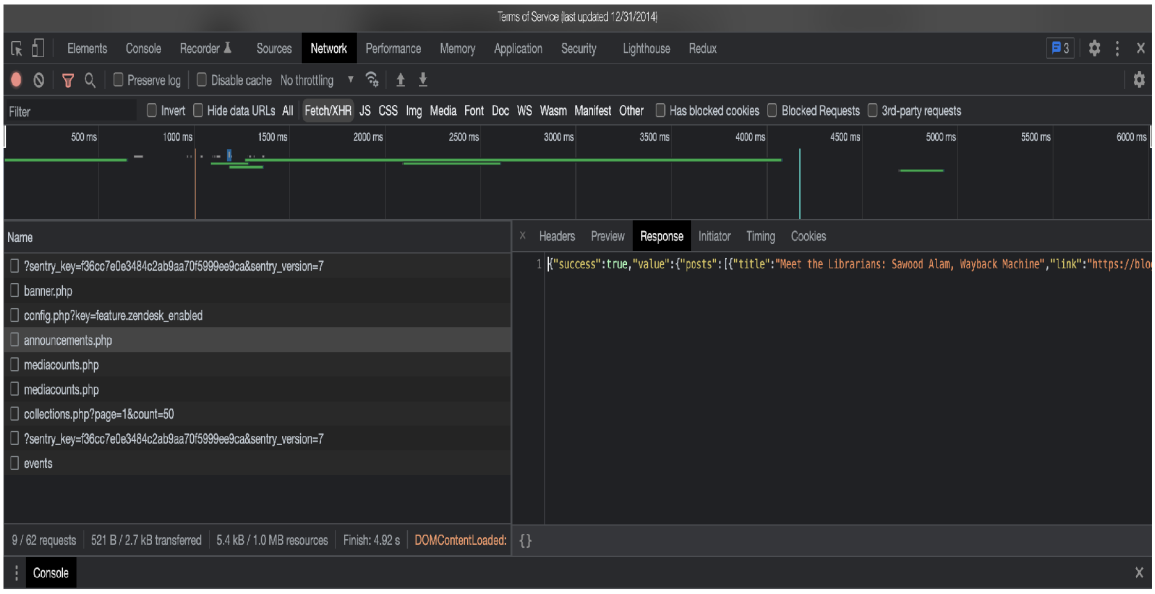
Samotné scrapování vícestránkové aplikace je viceméně detailněji popsáno v dřívější kapitole o způsobech získávání informací z internetu [odkaz na kapitolu]. Ve zkratce jde o klasické scrapování HTML elementů z webových stránek pomocí css selektorů nebo technologie xpath.

4.7.2 Single Page Aplikace

Single Page Application (SPA) je jednostránková webová aplikace, která je načítána pouze do jedné stránky. V SPA se žádná nová stránka po nějaké vyvolané akci nevykresluje, pouze se pomocí Javascriptu dynamicky mění elementy v DOM (rozhraní umožňující přistupovat k jednotlivým HTML elementům) a tím mění veškerý obsah webová aplikace [17].

Scraping jednostránkové aplikace

Scrapování jednostránkové aplikace je trochu komplikovanější než scrapování vícestránkové aplikace. Jak bylo popsáno dříve, jednostránková aplikace se skládá pouze z jedné stránky, která je postupně překreslována. Z toho důvodu je nesmysl z takové webové stránky scrapovat HTML elementy jako takové. Je nutné v Developers Console (vývojářská konzole ve webovém prohlížeči - po stisku klávesy F12) přejít na záložku Network (Síť) a poté XHR requests (v Google Chrome je to Fetch/XHR). Většina jednostránkových aplikací totiž načítají svůj obsah skrze XHR requesty (XMLHttpRequest požadavky). Každý XHR požadavek obsahuje Headers (hlavičky), Preview (náhled), Response (odezva ze serveru), Initiator (iniciátor), Timing (časové údaje požadavku) a Cookies. Při scrapování je zajímavá zejména Response, kde se nachází data vracená ze serveru. Obrázek č. 8 níže zobrazuje umístění Response ve vývojářské konzoli ve webovém prohlížeči Google Chrome.



Obrázek 8: Vývojářská konzole při scrapování SPA. Zdroj [Autor]

4.8 Značkovací jazyky

Značkovacími jazyky (markup languages) se rozumí takové jazyky, které jsou schopny anotovat digitální dokument. Tím většinou specifikují, co nějaká věc znamená, nebo jakým způsobem se má zobrazit. Nejprve se značky používaly pouze pro specifikaci vizuální podoby dokumentu. V dnešní době se ale značky používají i pro definování sémantiky, která je detailněji popsána v kapitole Sémantický web. Značkovací jazyk je tedy jazyk, který pomocí značek definuje elementy v rámci digitálního dokumentu [18].

Mezi nejznámější značkovací jazyky patří:

- HTML (HyperText Markup Language)
- XML (eXtensible Markup Language)
- XHMTL (eXtensible Hypertext Markup Language)
- KML (Keyhole Markup Language)
- MathML (Mathematical Markup Language)
- SGML (Standard Generalized Markup Language)

V této kapitole budou detailněji popsány jen dva nejpopulárnější značkovací jazyky, a to HTML a XML.

4.8.1 HTML

HTML je značkovací jazyk používaný pro vývoj webových stránek. Veškerý obsah webové stránky je nějakým způsobem specifikován pomocí HTML elementů. Každá HTML značka se značí znaky < a >. Jednotlivé HTML značky (tagy) se dělí do dvou základních skupin:

- Párové - párové značky musí obsahovat otevírací a uzavírací značku (příklad - <html></html>)
- Nepárové - jak již název napovídá, u nepárových značek není nutné řešit otevírací a uzavírací značku (příklad - <input>)

Níže uvedený kód č. 7 zobrazuje jednoduchý příklad HTML kódu:

```
<!DOCTYPE html>
<html lang="cs">
    <head>
        <meta name="title" content="Titulek stranky">
        <meta name="description" content="Popisek stranky">
        <link href="/media/style.css" rel="stylesheet">
        <script src="/media/script.js"></script>
    </head>
    <body class="homepage">
        <div class="main" id="main">
            <p>Toto je homepage webove stranky...</p>
        </div>
    </body>
</html>
```

Zdrojový kód 7: Ukázka struktury HTML dokumentu. Zdroj [Autor]

Jak je vidět na ukázce výše, HTML dokument se skládá z hlavní kořenové značky <html> a dále se dělí na hlavičku (<head>) a tělo (<body>). V hlavičce se nachází HTML značky specifikující funkce, popisky, styly apod. V těle dokumentu se zase nachází veškerý obsah vytvořených webových stránek. Většina HTML značek může obsahovat mnoho doplňkových atributů. Například značka obsahuje atribut class, který odliší tento od ostatních. Poté je možné například nastylovat jeden konkrétní . Atribut ID zastává obdobnou funkci jako class. Hlavním rozdílem je fakt, že ID musí být v rámci dokumentu zcela unikátní, kdežto stejný atribut class

může obsahovat více HTML elementů najednou. Vytvořený HTML soubor obsahuje koncovku .html.

4.8.2 XML

XML je značkovací jazyk stejně jako HTML, nicméně hlavní rozdíl mezi nimi je, že XML nedisponuje již předdefinovanými značkami. XML si totiž definuje zcela vlastní značky přesně podle potřeb uživatele. Hlavním důvodem popularity tohoto značkovacího jazyka je fakt, že do této struktury lze jednoduše ukládat, vyhledávat v ní a sdílet. Další důležitou vlastností je nezávislost XML na platformě. To znamená, že ať už se XML soubor otevře v jakémkoliv operačním systému, nebo na internetu nebo lokálně, vždy to bude stejné XML. Na tomto značkovacím jazyku je založeno několik dalších jazyků, jako je například XHTML, MathML, SVG, apod. Vytvořený XML soubor obsahuje koncovku .xml [19].
Níže v ukázce kódu č. 8 je uveden jednoduchý příklad typického XML souboru. V tomto případě se jedná o výstřižek ze sitemap webových stránek staveko.cz. Soubor sitemap.xml je detailněji popsán v dřívější kapitole.

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>http://www.staveko.cz/</loc>
        <priority>0.8</priority>
        <changefreq>weekly</changefreq>
    </url>
    <url>
        <loc>http://www.staveko.cz/demo</loc>
        <priority>0.6</priority>
        <changefreq>daily</changefreq>
    </url>
</urlset>
```

Zdrojový kód 8: Ukázka struktury XML dokumentu. Zdroj [Autor]

Zobrazení XML souboru je většinou ve formě nezpracovaného čistého XML. Nicméně je ale možné zobrazení obsahu XML souboru nějakým způsobem nastylovat. Potřeba je pouze definovat CSS soubor, pomocí xml-stylesheet instrukce, který se aplikuje na dokument a soubor nastyluje. Viz ukázka kódu č. 9 níže:

```
<?xml-stylesheet type="text/css" href="styles.css"?>
```

Zdrojový kód 9: Ukázka stylování XML dokumentu. Zdroj [Autor]

5 Návrh

Cílem návrhu praktické části bylo vybrat vhodné technologie, programovací jazyky, návrh databázové struktury pro scraper a zvolení vhodných výzkumných otázek, které nějakým způsobem obecně popíší značkovací zvyklosti webových stránek přibližně za posledních 20 let. Z porovnání webových stránek by mělo být vidět, se drží nějakých standardů souvisejících s optimalizací SEO, přístupností a nebo sémantikou webových stránek.

5.1 Archive.org

Prvním krokem bylo nutné vybrat cíl scrapování pro který bude scraper vytvořen. Nakonec byl vybrán web www.archive.org, který funguje jako archiv knih, videí, obrázků, textů, software, ale zejména jako archiv webových stránek. Služba pro archivaci webových stránek se nazývá Wayback Machine, která tedy umožňuje přístup všem uživatelům k archivovaným webovým stránkám. Typicky je uživateli umožněno zadat konkrétní URL adresu webových stránek a datum. Ve zmiňovaném archivu je doposud uloženo před 670 miliard webových stránek a toto číslo každým dnem roste. Archiv.org ukládá veškeré HTML, CSS a celkově vše o webových stránkách a následně je zobrazuje. Je tedy možné sledovat vývoj nějakých konkrétních webových stránek v jednotlivých měsících či letech. Zároveň je možné se podívat na webové stránky, které již na internetu nejsou dostupné. Motivací pro vytvoření tohoto digitálního archivu byl důraz na zachování digitálních artefaktů, jelikož v uchovaných artefaktech tkví vědění, a také možnost přístupu úplně všech uživatelů bez vyjímky k tomuto vědění [20].
Z webu web.archive.org tedy bude vyscrapováno několik desítek vybraných webových stránek včetně jejich veškeré historie. Po následném vyscrapování bude možné zkoumat různé přístupy k vývoji webových stránek v závislosti na daných letech, jako je například analýza titulku stránky nebo průměrný počet slov na stránku.

5.2 Návrh použitých technologií

Jako první krok po zvolení scrapovaného cíle bude nutné zvolit použité technologie pro scraper. Scraper tedy bude naprogramován v programovacím jazyce Python za pomocí konkrétního frameworku - Scrapy, který je detailněji popsán v praktické části, konkrétně v kapitole Přehled nástrojů pro získávání informací z internetu. Technologie

Scrapy bude vybrána zejména díky jednoduchosti implementace a lehké zkušenosti autora.

5.3 Návrh databázové struktury

Pro práci s databází bude vybrána databáze PostgreSQL. Databázová struktura nebude muset být nijak příliš složitá, jelikož jde pouze o to vyscrapovaná data z webových stránek nějakým způsobem uložit a následně s nimi pracovat.
Pro potřeby scraperu budou vytvořeny pouze dvě jednoduché tabulky bez jakýchkoliv vzájemných propojení a vazeb. Na obrázku diagramu č. X níže se tedy nachází dvě tabulky *visited* a *html-tag*.

- Tabulka visited slouží pouze jako seznam již navštívených stránek, ze kterých je dále nutné vyscrapovat požadovaná data a zároveň aby scraper věděl, že tyto navštívené stránky nemá scrapovat vícekrát než jednou, pokud se již nachází v databázi.
- Tabulka html-tag slouží pro uložení všech scrapovaných HTML tagů, se kterými se dále bude nějakým způsobem pracovat.

5.4 Přehled nástrojů pro získávání informací z internetu

Nástrojů pro webový scraping existuje nespočet. V této kapitole bude popsán framework Scrapy, python knihovna Beautiful Soup a lehký úvod do scrapingu bez nutnosti psaní kódu (tzn. volně dostupné již vytvořené scrapery).

5.4.1 Scrapy

Scrapy je open-source kolaborativní framework sloužící pro extrakci cílových dat z webových stránek jednoduchým, rozšiřitelným a hlavně rychlým způsobem. Scrapy je spravováno společností Zyte a mnohými dalšími přispěvovateli [21].

Instalace

Pro instalaci frameworku je nutné mít na koncovém zařízení stažený a nainstalovaný pouze Python programovací jazyk a pip. Poté se příkazem uvedeným v ukázce č. 10 níže v konzoli vytvoří projekt.

```
pip install scrapy
```

Zdrojový kód 10: Instalace Scrapy pomocí pip

Tím se vytvoří nový projekt, který obsahuje složku spiders, kde se vytváří tzv. pavouci, a python soubory init.py, items.py, middlewares.py, pipelines.py, settings.py a konfigurační soubor scrapy.cfg.

Spiders

Pavouci jsou třídy, které popisují z jakého webu a jak mají být data scrapována. Jinými slovy, v pavoucích se specifikuje veškeré chování procházení webů a získávání dat z nich. Existuje několik základních typů pavouků [21]:

- Spider - nejjednodušší pavouk, který nemá žádné speciální funkce či vlastnosti. Je pouze nutné specifikovat první URL, na které má pavouk začít scrapovat vybraná data.
- CrawlSpider - nejběžnější pavouk pro procházení webových stránek, jelikož obsahuje sadu pravidel, dle kterých se řídí a následuje odkazy.
- XMLFeedSpider - pavouk používaný k procházení a parsování XML feedů podle názvu uzlů.
- CSVFeedSpider - pavouk podobný předchozímu pavoukovi, rozdílem je pouze fakt, že prochází jednotlivé řádky dokumentu a ne uzly.
- SitemapSpider - pavouk umožňující procházení webů pomocí URL adres uložených v souboru Sitemap.

Proces procházení webů a získávání dat se skládá ze čtyř hlavních kroků [21]:

1. Vygenerování počátečních požadavků na procházení prvních url adres a specifikování callback funkce (funkce zpětného volání), která se má volat po stažení odpovědi z těchto požadavků
2. Ve zmíněné callback funkci se odchytne response (odpověď) požadavku, zanalyzuje se a vrátí se objekty itemu, objekty požadavku nebo iterace těchto objektů.
3. V callback funkci se zanalyzuje obsah stránky pomocí selektorů a generují se položky s analyzovanými daty.

4. Nakonec jsou položky z pavouka typicky uloženy do databáze nebo do nějakého námi specifikovaného souboru.

Ukázka jednoduchého pavouka níže na ukázce kódu č. 11. Pavouk má za úkol začít prohledávat adresu www.something.com, kde se například nacházejí URL nějakých kategorií (třeba produktů). Pavouk veškeré URL adresy jednotlivých kategorií získá a poté je pomocí follow all všechny projede a získá z nich například nějaké dále nespecifikované informace.

```
class TestSpider(scrapy.Spider, ABC):
    """Spider for something"""

    name = 'test_spider'
    allowed_domains = ['something.com']
    start_urls = [
        'https://www.something.com']

    def __init__(self, **kwargs):
        super(TestSpider, self).__init__(**kwargs)

    def parse(self, response, **kwargs):
        category_urls = response.css('div > div.name >
        a::attr(href)').getall()
        yield from response.follow_all(category_urls,
        callback=self.parse_category)
```

Zdrojový kód 11: Ukázka prohledávacího pavouka. Zdroj [Autor]

Items

Scrapovaná data jsou většinou získávána jako nestrukturovaná. Cílem třídy Item v souboru Items.py, je nějakým způsobem specifikovat požadovanou strukturu stahovaných nestrukturovaných dat.

Ukázka implementace Items:

```
class ProductScraperItem(scrapy.Item):
    """Product attributes"""
    source_name = scrapy.Field()

    section = scrapy.Field()
    subsection = scrapy.Field()
    subsubsection = scrapy.Field()

    title = scrapy.Field()
    short_description = scrapy.Field()
    long_description = scrapy.Field()
    price_with_vat = scrapy.Field()
    price_without_vat = scrapy.Field()
```

Zdrojový kód 12: Ukázka implementace třídy Item. Zdroj [Autor]

Pipelines

Soubor pipelines.py slouží k řízení toku dat ze scraperu například do databáze. Poté, co je item vyscrapován pavoukem je poslán do pipeline (potrubí), ve které se provede několik metod, které mají typicky za úkol item zprocesovat. Zde se také rozhoduje, zda vstupní item má být zprocesován dál nebo například vyhozen z procesu. Typicky slouží pipeline ke čtyřem základním úkonům [21]:

- Ukládání dat do databáze nebo souboru
- Validace dat
- Očištění HTML dat
- Sledování duplicit

Je možné pipeline rozdělit do více pipeline. Například pro každého pavouka jiná pipeline. Níže v ukázce č. 13 je uveden příklad jednoduché pipeline. Jak již název metod napovídá, metoda open_spider se provede ihned při otevření pavouka a vytvoří instanci dané pipeline. Při provedení close_spider se opouští pavouk a uzavírá se spojení s danou pipeline. Poslední metoda process_item má za úkol zprocesovat item, který prošel konkrétní pipeline. V tomto případě tato metoda uloží konkrétní item do

databáze, pokud se nachází v již dříve navštívených uložených URL adresách. Pokud se tam nachází, daný item bude vyhozen a nadále se s ním nebude pracovat.

```
# Pipeline instance creation in spider
def open_spider(self, spider: HeadSpider) -> None:
    spider.DiplomkaScraperPipeline = self

def close_spider(self, spider: HeadSpider):
    self.connection.close()

def process_item(self, item: HTMLTagItem, spider: HeadSpider):
    logging.debug("HeadSpider_PROCESS_ITEM")

    if item['url'] not in self.get_html_tags_urls():
        self.store_html_tags_to_db_execute(item)
        self.html_tags_urls.append(item['url'])
    elif item['url'] in self.get_html_tags_urls():
        DropItem(item['url'])

    return item
```

Zdrojový kód 13: Ukázka implementace pipeline. Zdroj [Autor]

Settings

Settings.py je velmi důležitým souborem, kde se dají nastavit nastavení pro celý scraper. Typicky se zde nastavují pipeline pro jednotlivé pavouky, moduly pavouků, vypisování logů do konzole nebo například zakázání nebo povolení cookies, zpoždění stahování dat a podobně. Ukázka nastavení v setting.py v ukázce č. 14

```
LOG_ENABLED = True
LOG_LEVEL = "INFO"
MEMDEBUG_ENABLED = True
BOT_NAME = 'diplomka_scraper'

SPIDER_MODULES = ['diplomka_scraper.spiders']
NEWSPIDER_MODULE = 'diplomka_scraper.spiders'

# Obey robots.txt rules
ROBOTSTXT_OBEY = True

ITEM_PIPELINES = {
    'diplomka_scraper.pipelines.DiplomkaScraperPipeline': 300,
}
```

Zdrojový kód 14: Ukázka implementace souboru settings.py. Zdroj [Autor]

5.4.2 Beautiful Soup

Beautiful Soup je knihovna v programovacím jazyce Python patřící k nejznámějším nástrojům používaných k automatizovaného získávání dat z XML a HTML souborů.

Instalace

Pro instalaci knihovny Beautiful Soup, dále jen BS, je opět nutné mít stažený a nainstalovaný Python na zařízení. Dále je potřeba použít pip, easy-install nebo setup.py pro samotnou instalaci BS.

```
pip install beautifulsoup4
easy_install beautifulsoup4
python setup.py install
```

Zdrojový kód 15: Možnosti instalace knihovny Beautiful Soup

Beautiful Soup mapuje celý HTML dokument do stromu objektů Pythonu. Je potřeba brát v potaz čtyři základní Python objekty [22]:

- Tag - klasický HTML nebo XML tag ve scrapovaném HTML dokumentu. Obsahuje různé atributy, mezi které patří jakékoliv atributy daného tagu, tedy například 'name', 'id', a podobně.

```
document = BeautifulSoup('<p><i>Testovaci text</i></p>',
'html.parser')
tag = document.i
tag.name
# 'i'
```

Zdrojový kód 16: Ukázka práce s objektem Tag. Zdroj [Autor]

- NavigableString - text obsažený v dříve zmiňovaném tagu.

```
document = BeautifulSoup('<p><i>Testovaci text</i></p>',
'html.parser')
tag = document.i
tag.string
# 'Testovaci text'
```

Zdrojový kód 17: Ukázka práce s objektem NavigableString. Zdroj [Autor]

- BeautifulSoup - objekt reprezentující celý scrapovaný HTML dokument. Pracuje se s ním víceméně stejně jako s Tagem.

```
document = BeautifulSoup("<item><url>Testovaci URL
</url></item>", "xml")
print(document)
# <?xml version="1.0" encoding="utf-8"?>
# <item><url>Testovaci URL</url></item>
```

Zdrojový kód 18: Ukázka práce s BeautifulSoup objektem. Zdroj [Autor]

- Comment - objekt sloužící k přístupu ke komentářům v HTML dokumentu.

```
document = BeautifulSoup("<p><!--Ignorovat tento kod-->
</p>", 'html.parser')
comment = document.p.string
comment
# 'Ignorovat tento kod'
```

Zdrojový kód 19: Ukázka práce s objektem Comment. Zdroj [Autor]

5.4.3 Scraping bez psaní kódu

Jak už bylo zmíněno dříve, v dnešní době již není nutné umět programovat pro vytvoření vlastního scraperu. Ovšem je možné, že dostupné již vytvořené scrapery budou nějakým způsobem omezené, nebo například nebude možné nastavit scraper přesně podle představ. Mezi nejznámější a nejpoužívanější již vytvořené scrapery patří například [23]:

- ParseHub
- OctoParse

Tyto vytvořené scrapery fungují většinou tak, že je nutné stáhnout a nainstalovat jejich desktopovou aplikaci a zadat URL, ze které je nutné stáhnout data. Zároveň je možné specifikovat elementy, které chce uživatel získat ze zadané webové stránky. ParseHub a Octoparse jsou jedním z největších hráčů na poli již vytvořených dostupných scraperů. Fungují primárně tak, že je nutné stáhnout a nainstalovat jejich desktopovou aplikaci. Poté zadat URL adresu, kterou je potřeba scrapovat. Aplikace poté stahuje všechna možná data a ukládá je na jejich vlastní servery. K datům je poté možné přistupovat přes JSON, Excel a nebo API. Typicky pak ještě obě aplikace disponují umělou inteligencí, která je na základě hlubokého učení a velkého množství dat schopná získat většinou přesně ta data, která uživatel chce ve strukturovaném přehledném stavu.
ParseHub i Octoparse lze používat zcela zdarma, avšak s velmi velkými omezeními, nicméně pro nějaké domácí projekty to nejspíše vystačí. Webové stránky obou scraperů obsahují plno tutoriálů jak o webovém scrapingu jako takovém, tak hlavně o používání aplikace jako takové [ParseHub a Octoparse dokumentace].

5.5 Příprava výzkumných otázek

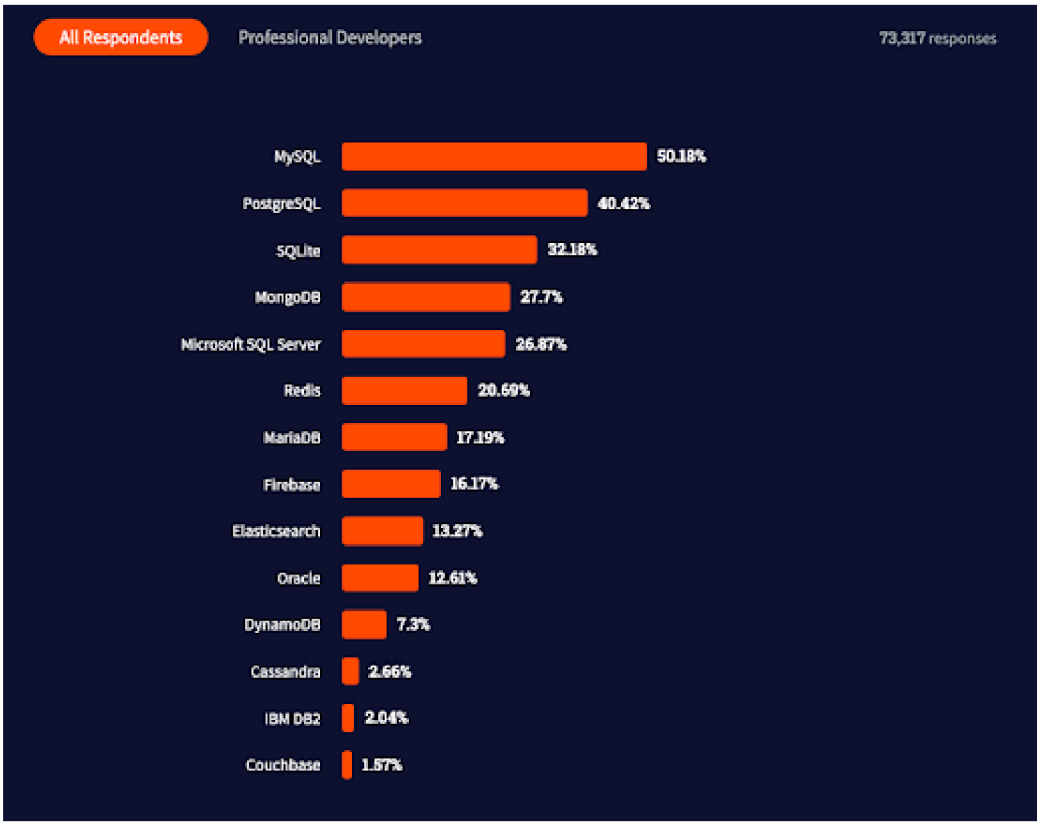
Dalším krokem bude navrhnutí výzkumných otázek, jejichž výsledkem by mělo být jakési obecné popsání toho, jak byly webové stránky vytvářeny dříve a jak se jejich vývoj postupně vyvíjel v čase. Analýza bude doprovázena známými doporučeními pro dané HTML elementy z pohledu SEO, přístupnosti a sémantiky webových stránek. Typicky se otázky budou zabývat značkovacími zvyklostmi, úzce spojenými se SEO optimalizací, přístupností a částečně i sémantikou webu, na webových stránkách vzhledem k vývoji webových stránek v čase.

6 Implementace

V kapitole Implementace bude popsán princip ukládání získávaných dat scraperem a samotná implementace scraperu.

6.1 Princip ukládání scrapovaných dat

Jak již bylo zmíněno v kapitole Návrhu, pro ukládání požadovaných scrapovaných dat byla použita databáze PostgreSQL. Jak je vidět na obrázku č. 9 níže, PostgreSQL je aktuálně na druhé pozici nejpopulárnějších databází na světě.



Obrázek 9: Procentuální využití databází. Zdroj [4]

Pro ukládání dat jsou využity dvě tabulky - visited a html_tag, které jsou detailněji popsány v kapitole o návrhu scraperu. Víceméně se ale všechna důležitá data o webové stránce ukládají jako jeden samostatný záznam do tabulky.

6.2 Implementace scraperu

Již několikrát bylo zmíněno, že pro vývoj webového scraperu bude použit programovací jazyk Python a současně s tím framework Scrapy. Scrapy je podrobněji popsán v kapitole Návrhu, konkrétněji v podkapitole Přehled nástrojů pro získávání informací z internetu.

6.2.1 Získání URL adres

Nejprve bylo nutné získat URL adresy jednotlivých záznamů webových stránek z již dříve zmíněného webového archivu Archive.org, aby následně mohl být vytvořen pavouk, který všechny získané URL adresy projde a uloží požadovaná data, která budou nadále zkoumána. Po bližším zkoumání webového archivu stránek Archive.org bylo zjištěno, že archiv nabízí API, na kterém jsou vystaveny všechny záznamy stránek pro určitou webovou stránku. Byla tedy vytvořena metoda, která získá data z API a následně je přes pavouka uloží do databáze, konkrétně do tabulky *visited*. Implementace metody je uvedena v ukázce kódu č. 20 níže:

```
def download_urls(site: str, fmt: str):
    url = "https://web.archive.org/cdx/search/cdx?url=%s
    &output=%s" % (site, fmt)
    response = requests.get(url)
    url_list: list = []
    for i in json.loads(response.text):
        url_list.append("https://web.archive.org/web/%s/%s"
        % (i[1], i[2]))

    return url_list
```

Zdrojový kód 20: Implementace metody pro získání URL adres z archivu. Zdroj [Autor]

Metoda obsahuje na vstupu název webové stránky (např. bonprix.cz) a formát (např. JSON). Poté se dotáže GET metodou na dané API a získá veškeré obsahované URL adresy.

6.2.2 Pavouk pro získání URL adres

Následně bylo nutné vytvořit pavouka, který využije metodu zmíněnou v předchozí podkapitole, projde dané URL adresy a uloží je do databáze. Pavouk postupně projde jednotlivé URL, jak je vidět na ukázce kódu č. 21.

```
def start_requests(self):
    for site in self.sites:
        for url in download_urls(site, self.fmt):
            yield scrapy.Request(url)
```

Zdrojový kód 21: Odesílání požadavků na dané URL adresy. Zdroj [Autor]

Následně se provede metoda parse, která je uloží do databáze - ukázka kódu č. 22. Pavouk ukládá source_name, což je zdrojová webová stránka (např. bonprix.cz), url stránky a kdy byla navštívena.

```
def parse(self, response, **kwargs):

    item = HTMLTagItem()

    today = datetime.today()

    item["source_name"] = ""
    for site in self.sites:
        if site in response.url:
            item["source_name"] = site
    item["url"] = response.url
    item["last_scraped"] = today

    yield item
```

Zdrojový kód 22: Parse metoda timestamp pavouka. Zdroj [Autor]

6.2.3 Pavouk pro získání dat z webových stránek

Po provedení prvního pavouka, který má za úkol získat jednotlivé URL adresy bude spuštěn druhý pavouk, který každou stránku detailně projde a získá požadované informace, ze kterých budou následně provedeny stanovené analýzy. Tento pavouk získává 69 různých informací pro každý navštívený záznam webové stránky. Víceméně se

jedná o celou hlavičku dokumentu a následně několik HTML elementů ze zbytku dokumentu. Následné analýzy jsou vytvořeny jen z některých z nich, další jsou získávány zejména pro budoucí rozšíření a práci s pavoukem a daty. V tomto pavoukovi jsou také řešeny jednoduché potřebné výpočty usnadňující následné analýzy dat. Jeden z nich je uveden v ukázce kódu č. 23. V uvedené ukázce je řešen automatický výpočet počtu výskytu klíčových slov v textu v procentech.

```
temp: int = 0
if keywords is not None:
    keywords_array = keywords.split(",")
    for keyword in keywords_array:
        if keyword.lower() in body_without_stopwords:
            temp += 1
elif keywords_2 is not None:
    keywords_2_array = keywords_2.split(",")
    for keyword in keywords_2_array:
        if keyword.lower() in body_without_stopwords:
            temp += 1

 # how many % are keywords from the whole text body
    if len(body_without_stopwords) != 0:
        html_tag_item['keywords_percent'] = (temp / len(body_without_stopwords))
    else:
        html_tag_item['keywords_percent'] = 0
```

Zdrojový kód 23: Parse metoda timestamp pavouka

Mezi další automatické výpočty při procházení webových stránek patří výpočet délky titulku nebo popisku, počet hlavních nadpisů (h1) nacházejících se na dané stránce, počet slov a znaků veškerého textu, počet obrázků a alternativních popisků, počet CSS souborů, atd. Získané byly taky nejpoužívanější slova na dané stránce, zde bylo nutné vytvořit si svůj seznam tzv. stopwords (nechtěná slova), která byla vyškrtnuta z výsledných nejpoužívanějších slov. Jedná se o slova typu: asi, nebo, jsem, proč, atd.

7 Aplikace

Implementovaný scraper bude nyní použit pro scraping několika desítek webových stránek (domén) z již dříve zmiňovaného webového archivu webových stránek Archive.org. Celkově bude vyscrapováno přibližně 31 623 webových stránek. Webové stránky budou zkoumány hlavně z hlediska HTML elementů důležitých pro optimalizaci SEO, přístupnosti webových stránek a částečně i sémanticky webů.
Kapitola zabývající se výslednými poznatky z analýzy vybraných zkoumaných HTML elementů na webových stránkách. Potřebná teorie k pochopení jednotlivých HTML značek se nachází v teoretické části práce v kapitole Kategorie získávaných informací z webových stránek.

7.1 Analýza titulku webové stránky

Jedním z připravených zkoumaných scénářů byla analýza titulku webových stránek. Zkoumala se především délka titulku, pro kterou platí doporučení ideálně 55-65 znaků. Taková délka totiž umožní zobrazit ve výsledcích vyhledávání všechny znaky a předejde se tím "useknutí"textu a zobrazení nikým nechtěných tří teček (...). Obecným, ale nejefektivnějším doporučením je bohatý obsah titulku související s hlavními klíčovými slovy a hlavním tématem dané webové stránky. Dohromady by měl být titulek jakási souvislá smysluplná věta. Přílišné zmiňování stejných slov by mohlo vést k negativním důsledkům, zejména k horším zobrazovacím pozicím. Dále Google dává větší důraz na slova na prvních pozicích ve větě, tudíž je logicky výhodnější zmiňovaná hlavní klíčová slova vložit na přední pozice titulku. Text titulku by také neměl obsahovat například pomlčku (-) pro oddělení více slov. Namísto toho je doporučené použít znak roury (|). Pokud není titulek vyplněn, Google ho v mnoha případech vyplní sám, a to ve většině případů nevede k žádným uspokojivým výsledkům [24].
Výsledky analýzy
V tabulce č. 1 je možné zpozorovat, že z celkového počtu 31 623 stránek má titulek vyplněno přibližně 87,65 % webů. Z webů, které mají vzorně vyplněný titulek je pouze 15,49 % webů, které tedy splňují zmiňované doporučení k počtu znaků. Ze čtvrtého řádku tabulky lze soudit, že titulky neobsahují žádné duplicity, a to z celých 100 % zkoumaných webových stránek. Z posledního řádku je zřejmé, že titulek na většině webových stránkách obsahuje příslušná klíčová slova dané stránky z 18,54 %. Hlavní

klíčová slova stránky se získávají z veškerého textu nacházejícího se na konkrétní webové stránce.

Analýza <title> tagu	Ano	Ne
Obsahují weby <title> tag?	87,65%	12,35%
Splňují weby doporučenou délku 55-65 znaků?	15,49%	84,51%
Obsahuje <title> tag duplicity?	0,00%	100,00%
Obsahuje <title> tag nejdůležitější keywords?	18,54%	81,46%

Tabulka 1: Analýza <title> tagu. Zdroj [Autor]

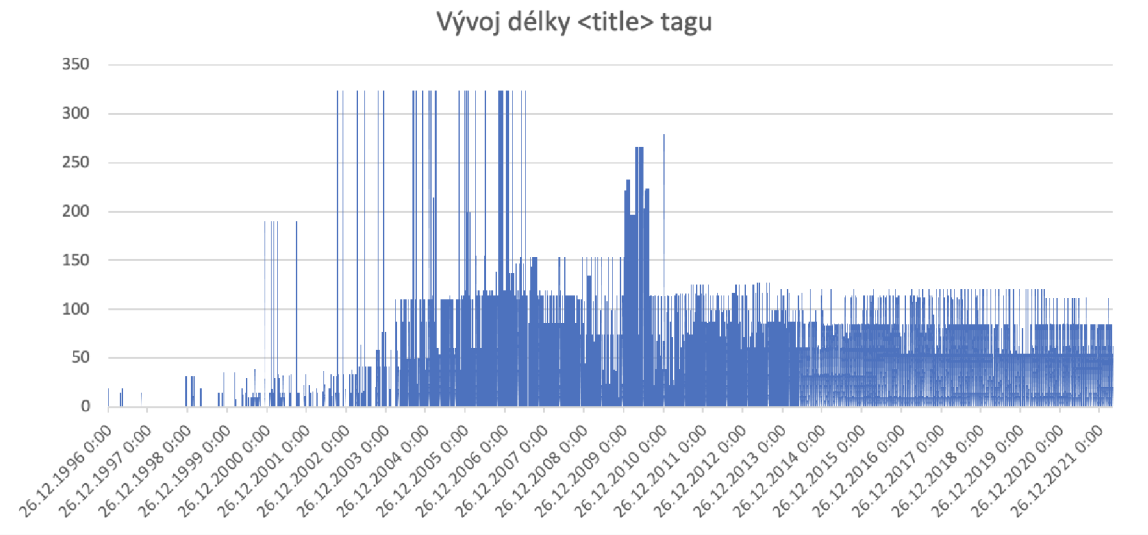
V tabulce č. 2 je zase znázorněno rozdělení titulků dle počtu znaků. Největší zastoupení má titulek s počtem znaků mezi 10 a 20 znaky. To je velmi podprůměrné a s tímto počtem znaků ani není možné správně vytvořit dle doporučením SEO optimalizace. Obecně jakýkoliv titulek pod přibližně 50 znaků nebo nad 65 znaků bude nejspíše prohledávacím robotem označen jako neefektivní pro optimalizaci SEO.

Analýza počtu znaků v <title> tagu	Obsahují	Neobsahují
65+ znaků	12,62%	87,38%
55-65 znaků (doporučené)	15,49%	84,51%
40-55 znaků	6,99%	93,01%
30-40 znaků	8,86%	91,14%
20-30 znaků	3,99%	96,01%
10-20 znaků	49,45%	50,55%
0-10 znaků	2,59%	97,41%

Tabulka 2: Analýza počtu znaků <title> tagu. Zdroj [Autor]

Zkoumán byl také HTML tag meta title, který byl občas zaměňován za klasický HTML tag <title>, nicméně není tomu tak a je to špatný přístup k optimalizaci pro vyhledávače. Dle průzkumu byl ale vyplněn pouze u necelého půl procenta všech zkoumaných webových stránek v průběhu let, tudíž není úplně důležité ho nějak podrobněji zkoumat.
Na obrázku č. 10 je vyobrazen graf, který ukazuje trend vyplnění titulků v jednotlivých letech od roku 1999 po rok 2022. HTML značka <title> začala být podporována

v roce 2002 na Internetu Exploreru. Klíčový bod v tomto grafu je bod v roce 2004, kdy titulek začal mít mnohem větší měřítko, než tomu bylo doposud. Ačkoliv byl titulek vyžadován již od roku 2002, je zřejmé, že na vývoji webových stránek se tato skutečnost projevila až po dvou letech.



Obrázek 10: Vývoj délky <title> tagu v čase. Zdroj [Autor]

7.2 Analýza popisku webové stránky

Druhým zkoumaným scénářem byla analýza HTML tagu *<meta name="description»*, tedy popisku webové stránky. V tomto případě se zkoumaly stejné věci jako u titulku webové stránky, tedy zkoumalo se především jestli jednotlivé webové stránky vůbec obsahují popisek, dále délka popisku, pro kterou platí doporučení ideálně 150-160 znaků a obsah klíčových slov a duplicit.

V tabulce č. 3 v tabulce je možné zpozorovat, že z celkového počtu 31 623 stránek má meta description tag vyplněno přibližně 38,80 % webů. Z webů, které mají vzorně vyplněný popisek je pouze 21,59 % webů, které splňují zmiňované doporučení k počtu znaků. Průměrným počtem znaků v popisku je 133 znaků. Na posledním řádku lze vidět, že popisek na většině webových stránkách obsahuje příslušná klíčová slova dané stránky z 29,86 %. Hlavní klíčová slova stránky jsou získávána z meta keywords tagu nacházejícího se na konkrétní webové stránce.

Analýza meta description tagu	Ano	Ne
Obsahují weby meta description tag?	38,80%	61,20%
Splňují weby doporučenou délku 150-160 znaků?	21,59%	78,41%
Obsahuje tag nejdůležitější keywords?	29,86%	70,14%

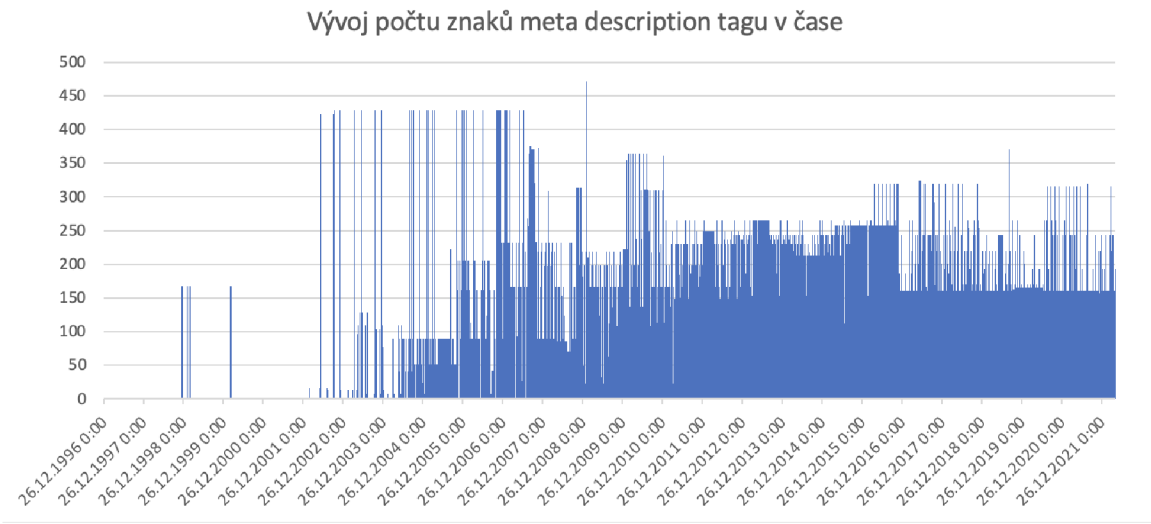
Tabulka 3: Analýza meta description tagu. Zdroj [Autor]

V tabulce č. 4 je zase znázorněno rozdělení popisků dle počtu znaků. Největší zastoupení mají popisky s počtem znaků mezi 100 a 150 znaky. Doporučenou délku pro optimalizaci SEO splňuje dle tabulky pouze 21,60 % zkoumaných webových stránek.

Analýza počtu znaků v meta desc tagu	Obsahují	Neobsahují
160+ znaků	20,32%	79,68%
150-160 znaků (doporučené)	21,60%	79,40%
100-150 znaků	29,98%	70,02%
50-100 znaků	13,15%	86,85%
0-50 znaků	14,95%	85,05%

Tabulka 4: Analýza počtu znaků meta description tagu. Zdroj [Autor]

Na obrázku č. 11 je vyobrazen graf, který ukazuje vyplnění popisků a délky popisků v jednotlivých letech od roku 1999 po konec roku 2021. Podle webu www.caniuse.com začala být HTML značka meta podporována v roce 2006, ale pouze na Mozille Firefox. Na obrázku je viditelné, že popisek byl v menším měřítku používán již před podporou internetového vyhledávače Mozilly Firefox.



Obrázek 11: Vývoj délky meta description tagu v čase. Zdroj [Autor]

7.3 Analýza hlavních nadpisů

Mezi hlavní doporučení při vytváření nadpisů patří:

- Použití pouze jednoho <h1> nadpisu na jedné webové stránce - webové crawlery Googlu používají tyto nadpisy k identifikace tématu dané stránky
- Délka <h1> by měla mít maximálně 80 znaků
- Použití klíčových slov - stejně jako u titulku a popisku je doporučeno používat klíčová slova dané stránky
- Použití správné hierarchie nadpisů - to znamená postupně použít <h1>, <h2>, <h3>, a tak dále
- Výběr nadpisu by neměl záviset na jeho velikosti, ale na sémantickém významu - velikost nadpisů se dá jednoduše změnit, naproti tomu sémantický význam se změnit nedá

Z výsledné analýzy lze vypozorovat, že hlavní nadpisy <h1> obsahuje pouze 22,56 % webů. Zbylých 77,44 % scrapovaných stránek neobsahuje žádné <h1> nadpisy. Jak bylo zmíněno dříve, hlavní nadpisy by měly obsahovat nejdůležitější klíčová slova zkoumané stránky a měly by shrnout celou stránku. Z tabulky č. 5 vyplývá, že přibližně 18,54 % zkoumaných webových stránek opravdu nějká klíčová slova obsahuje. Zbylých 81,46 % neobsahuje žádná klíčová slova [25].

Analýza nadpisů na webu	Ano	Ne
Obsahují weby H1 tag?	22,56%	77,44%
Obsahuje tag nejdůležitější keywords?	18,54%	81,46%

Tabulka 5: Analýza H1 tagu. Zdroj [Autor]

7.4 Analýza textů na webových stránkách

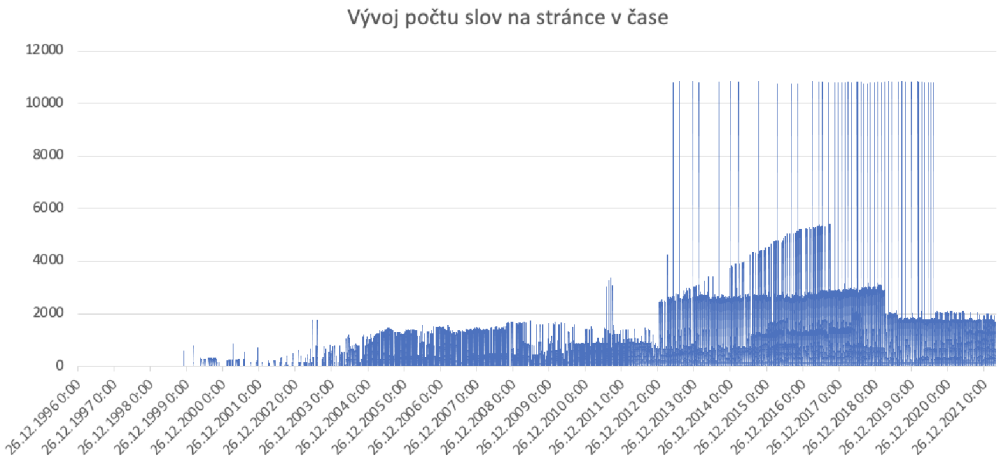
Stejně jako předchozí analýzy, i tato úzce souvisí s optimalizací SEO a má na ní poměrně velký vliv. Konkrétně je analyzována délka textu v jednotlivých slovech a v jednotlivých znacích. Obecně platí, že texty na webových stránkách by neměly být ani příliš krátké, ani moc dlouhé. Dlouhé texty by se mohly zdát jako dobrý nápad, nicméně vyhledávací bot Googlu by poté mohl mít problém zařadit danou webovou stránku k určitému tématu. To by vedlo k dosahování horšího hodnocení webové stránky a k následným horším pozicím ve vyhledávači. Obecně pro texty umístěné na webových stránkách platí, že by se měly zaobírat pouze tématem dané webové stránky, měly by obsahovat nejdůležitější klíčová slova a měly by být správně strukturovány do nadpisů, odstavců, a podobně. Ideální délka dle Googlu by měla být více než 800 slov, ale méně než 2000 slov na dané webové stránce. Je doporučeno, aby se v textech objevovaly klíčová slova přibližně v 2,5 % - 5 % z veškerého textu. Obdobně jako u titulku, popisku a nadpisů platí, že každá stránka by měla obsahovat text vztahující se pouze k dané webové stránce a daný text by měl být unikátní pro každou stránku [26].

Předmět zkoumání	Hodnota
Průměrný počet slov na stránce	802
Průměrný počet znaků na stránce	4941
Počet stránek obsahujících klíčová slova	13,34%
Průměrná hodnota výskytu klíčových slov v textu	0,28%

Tabulka 6: Analýza textu na stránkách. Zdroj [Autor]

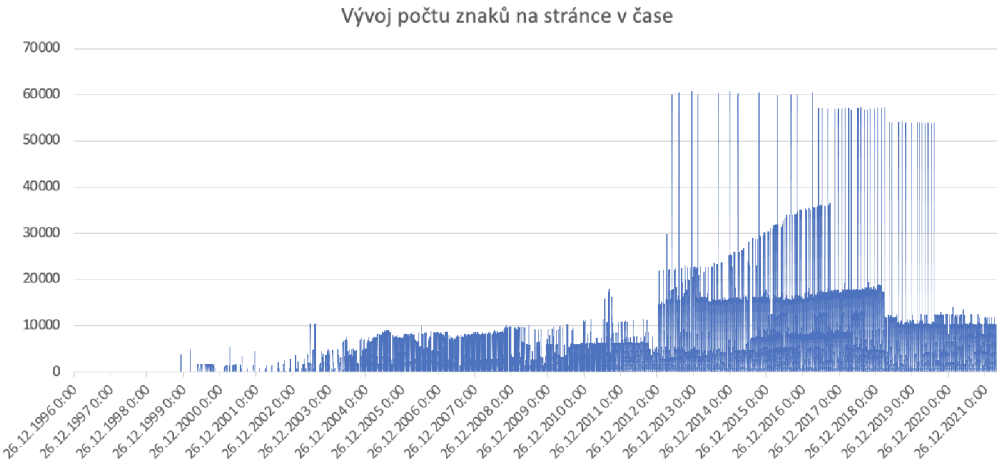
Na obrázku č. 12 se nachází dvojrozměrný sloupcový graf, který demonstruje vývoj počtu slov na webových stránkách v závislosti na čase. Průměrný počet slov za všechny stránky v každém roce je přibližně 803. Z grafu zároveň vyplývá, že až

přibližně v roce 2004-2005 se začal nějakým způsobem brát větší důraz na obsah stránky jako takový. To by mohlo být způsobené příchodem webu 2.0 na scénu. Jako začátek webu 2.0 by mohla být považována konference "Web 2.0 Conference"pořádaná O'Reilly Media a MediaLive 5. října 2004 v San Franciscu [27].



Obrázek 12: Vývoj počtu slov na stránce v čase. Zdroj [Autor]

Na obrázku č. 13 je vidět podobný sloupcový graf, který zobrazuje vývoj počtu znaků na webových stránkách v závislosti na čase. Z grafu lze vidět, že téměř kopíruje graf vývoje počtu slov. Průměrný počet znaků na stránce za všechny stránky je přibližně 4941.



Obrázek 13: Vývoj počtu znaků na stránce v čase. Zdroj [Autor]

7.5 Analýza struktury webových stránek

Tato analýza spíše než se SEO, souvisí se sémantikou webových stránek. Konkrétně je sledováno a analyzováno rozdělení struktury stránek do 3 základních struktur - header, main a footer. Tento scénář je analyzován zejména proto, jelikož se dříve, před příchodem těchto sémantických HTML elementů, používaly elementy s označením třídy (class) header, main a footer. Je to tedy zastaralý princip strukturalizace HTML dokumentu jako celku. Je doporučeno používat zmíněné sémantické elementy radši než přes elementy, jelikož třída divu může být jakákoliv, může obsahovat překlep a podobně. Nicméně sémantický element <header> jasně říká, že zde se nachází hlavička dokumentu a nic jiného. To samé platí pro <main> a <footer>. Element <main> zase specifikuje hlavní obsah dané webové stránky a <footer> patičku webové stránky, kde se většinou nachází například informace o copyrightu nebo kdo dané webové stránky navrhl nebo vytvořil. Dle [caniuse.com] začal být <header> v nějaké větší míře podporován až v roce 2010. Element <main> v roce 2013 a <footer> stejně jako <header> začal být masivněji podporován v roce 2010.
Z grafu č. 7 níže lze vidět, že nejvíce byl používaný tag s označením *class* nebo *id* header, a to ve 32,32 % všech zkoumaných webových stránek. To samé akorát platící pro main byla použita pouze v 1,10 % případů a footer v 9,67 %. K poměrně velkému překvapení tagy <header>, <main> a <footer> nebyly použity ani v jednom případě.

Předmět zkoumání	Hodnota
Počet stránek obsahující div header	32,32%
Počet stránek obsahující div main	1,10%
Počet stránek obsahující div footer	9,67%
Počet stránek obsahující tag header	0%
Počet stránek obsahující tag main	0%
Počet stránek obsahující tag footer	0%

Tabulka 7: Analýza textu na stránkách. Zdroj [Autor]

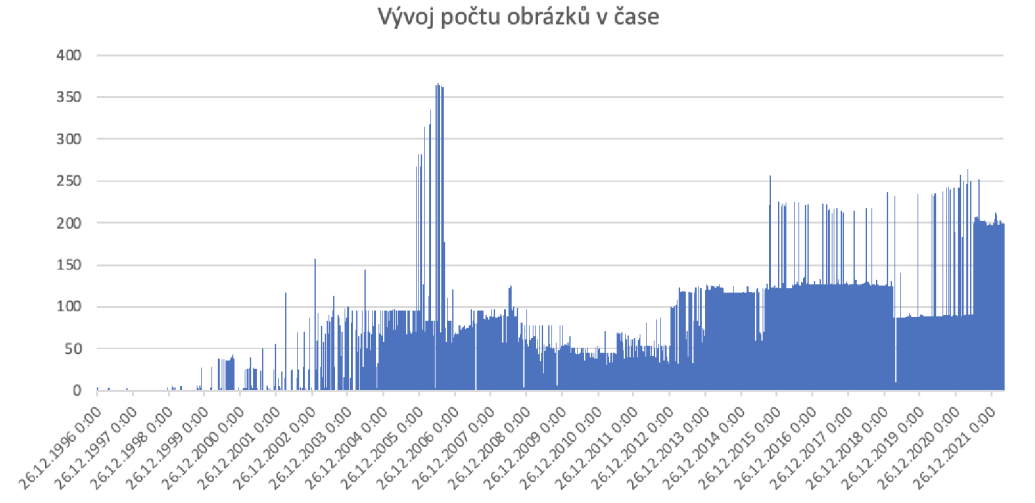
7.6 Analýza obrázků na webových stránkách

Tato konkrétní analýza zase souvisí spíše s přípustností webových stránek, než se SEO optimalizací nebo sémantikou webu a bude se zabývat primárně počty obrázků na stránce a počty vyplněných atributů *alt*. Nebude zkoumána struktura URL, název, ani velikosti souborů, jelikož takovéto informace nebylo možné vyscrapovat ze zmiňovaného archivu webových stránek.

Níže v tabulce č. 8 je možné zpozorovat, že průměrný počet obrázků na jednu stránku je 42. To je poměrně velké číslo a na obrázku č. 14 je vidět, že je to způsobené řadou vyscrapovaných webů, které mnoho let používaly obrázky v opravdu velké míře a to například i pro menu na stránkách a tím se tento průměr prudce zvedl. Nicméně průměrný počet *altů* na stránce je 30, to je 71,43 % obrázků a jak již bylo zmíněno v teoretické části - alty by měly být vyplněny ve všech případech, jelikož zvyšují skóre optimalizace SEO a zároveň splňují doporučení pro přístupnost webu.

Předmět zkoumání	Hodnota
Průměrný počet obrázků na stránce	42
Průměrný počet altů na stránce	30
Počet stránek obsahujících obrázek	88,20%
Počet stránek obsahující alt	88,20%

Tabulka 8: Analýza textu na stránkách. Zdroj [Autor]



Obrázek 14: Vývoj počtu znaků na stránce v čase. Zdroj [Autor]

8 Závěr

Ze zadání diplomové práce bylo hned očividné, že bude nutné nalézt již vytvořený nebo vytvořit vlastní nástroj pro získání specifických informací z webových stránek. Jako zdroj webových stránek a zároveň tedy i potřebných informací byl použit webový archiv webových stránek web.archive.org. Archiv byl vybrán zejména proto, aby bylo možné zaznamenat změny, popřípadě nějaké trendy, v čase a sledovat tak vývoj daných přístupů k vytváření webových stránek, nebo různých HTML elementů. Ačkoliv již existuje mnoho nástrojů pro scrapování informací, rozhodl jsem se vytvořit svůj vlastní scrapovací nástroj a to zejména kvůli vysokým poplatkům za využití existujících nástrojů, jelikož bylo nutné scrapovat několik desítek tisíc záznamů webových stránek ze zmiňovaného archivu webů. Scraper byl vyvinut v programovacím jazyce Python za použití frameworku Scrapy a PostgreSQL databáze.
Dalším krokem bylo stanovení a vypracování výzkumných scénářů, které se zabývají analýzou získaných dat se zaměřením na SEO, přístupnost a sémantiku webových stránek.
V teoretické části bylo vysvětleno, co je to informace a popsáno jaké informace a jakým způsobem z webových stránek budou získávány. Dále je popsán obecně webový scraping, etika a legalita webového scrapingu a seznámení s některými hlavními technologiemi používaných pro webový scraping. V kapitole zabývající se kategoriemi získávaných informací bylo popsána optimalizace SEO, přístupnost a částečně i sémantika webových stránek.

9 Shrnutí a doporučení

V průběhu vývoje diplomového scraperu byl proveden pokus srovnání rychlosti scrapování webových stránek v programovacím jazyce Python a Go. Scraper napsaný v Go se ukázal přibližně o 45 % rychlejší v procházení webových stránek. Při vypracování dalšího scraperu by byl už spíše použit Go.

9.0.1 Python versus Go scraping

Když se řekne webový scraping, většina zasvěcených lidí si nejspíš představí programovací jazyk Python. Python je v oboru webového scrapingu velmi známým hráčem a obsahuje nespočet knihoven, které samotné scrapování ve velké míře usnadňují. Nicméně v poslední době se na scéně webového scrapingu objevuje programovací jazyk Go, který by měl být až o 50 % rychlejší než zmiňovaný Python. Otázkou tedy je, ovládne Go prostředí webového scrapingu? Odpověď zní, zatím nejspíše ne, jelikož Python má obrovskou popularitu a podporu. Nicméně se stoupající popularitou Go dost možné, že za několik let Python překoná.

9.0.2 Python

K implementaci demonstračního příkladu pro webový scraping byla vybrána Python knihovna Beautiful Soup. Především díky její jednoduchosti a rychlejší implementaci ve srovnání s frameworkem Scrapy. Níže je uvedena část kódu na ukázce č. 24.

```
def get_row(row):
    session = get_session()
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
        AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
        Safari/537.36'}
    session.headers = headers

    with session.get(row["url"]) as response:
        print(row["url"])

        results = BeautifulSoup(response.content, "html.parser")
        quote_price = results.find("span", class_="Trsdu(0.3s) Fw(b)
        Fz(36px) Mb(-4px) D(ib)")

def get_rows(rows):
    with concurrent.futures.ThreadPoolExecutor(max_workers=5) as executor:
        executor.map(fetch_row, rows)
        executor.shutdown(wait=True)
```

Zdrojový kód 24: Ukázka scrapování v programovacím jazyce Python za použití BS. Zdroj [Autor]

9.0.3 Go

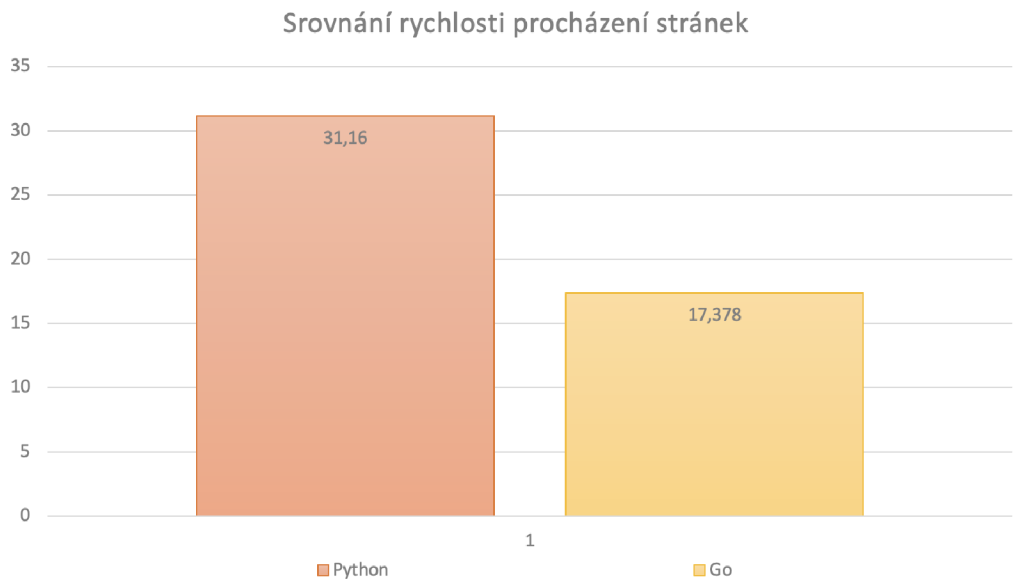
Pro implementaci jednoduchého příkladu sloužícího jako demonstrace rychlosti scrapování v programovacím jazyce Go byl použit framework Colly [ODKAZ]. V příkladu č. 25 níže je uvedena část kódu v jazyce Go, který by měl ušetřit přibližně 45 % scrapovaného času a zároveň vyscrapovat mnohem více dat než Python.

```
c := colly.NewCollector(
        colly.Async(),
)
c.Limit(&colly.LimitRule{DomainGlob: "*", Parallelism: 5})

c.OnHTML("#quote-header-info", func(e *colly.HTMLElement) {
        name := e.ChildText("h1")

        temp := strings.Split(name, "(")
        name = temp[0]
        symbol := temp[1][:len(temp[1])-1]

        row := rows[symbol]
        row.Name = name

        rows[symbol] = row
})
c.OnResponse(func(r *colly.Response) {
        fmt.Println("Scraping je dokoncen")
})

c.OnRequest(func(r *colly.Request) {
        fmt.Println("Navstevuji tuto URL", r.URL)
})
```

Zdrojový kód 25: Ukázka scrapování v programovacím jazyce Go za použití Colly. Zdroj [Autor]

9.0.4 Zhodnocení

V obou jazycích byl proveden pokus - sledovat scrapování sta URL adres v čase. Oba scrapery jsou nastaveny na pět vláken. Každý scraper byl spuštěn 5x a výsledné hodnoty se zprůměrovaly. Konečné hodnoty v grafu jsou v sekundách. Z grafu č. 15 níže je patrné, že příklad napsaný v jazyce Go je přibližně o 45 % efektivnější, než kód řešící stejnou problematiku napsaný v jazyce Python. Celkově by tedy Go bylo schopné vyscrapovat více jak dvojnásobek scrapovaných webových stránek za téměř poloviční čas. Blíží se éra webového scrapování v Go?



Obrázek 15: Srovnání doby scrapingu v Pythonu a Go. Zdroj [Autor]

Framework Colly nebyl zkoumán nijak moc dopodrobna, nicméně po provedení tohoto pokusu by pro vypracování diplomového scraperu byl nejspíše zvolen jazyk Go. Zejména díky zmíněné mnohem vyšší rychlosti scrapování. Ušetřila by se tím téměř polovina scrapovaného času.

Literatura

[1] W3C Semantic Web Activity. Available from: https://www.w3.org/2001/12/semweb-fin/w3csw

[2] Semantic Search: How It Impacts Your SEO Results. Available from: https://www.semrush.com/blog/semantic-search

[3] Data Mining: Concepts and Techniques | ScienceDirect. Available from: https://www.sciencedirect.com/book/9780123814791/data-mining-concepts-and-techniques

[4] The Most Popular Databases for 2022. Jan. 2022, section: blog. Available from: https://learnsql.com/blog/most-popular-databases-2022/

[5] Initiative (WAI), W. W. A. Accessibility Principles. Available from: https://www.w3.org/WAI/fundamentals/accessibility-principles/

[6] Davis, H. *Search Engine Optimization*. "O'Reilly Media, Inc.", May 2006, ISBN 978-1-4919-0588-3, google-Books-ID: hYSMBAAAQBAJ.

[7] PageRank. Sept. 2021, page Version ID: 20486742. Available from: https://cs.wikipedia.org/w/index.php?title=PageRank&oldid=20486742

[8] Katumba, S.; Coetzee, S. Employing Search Engine Optimization (SEO) Techniques for Improving the Discovery of Geospatial Resources on the Web. *ISPRS International Journal of Geo-Information*, volume 6, no. 9, Sept. 2017: p. 284, ISSN 2220-9964, doi:10.3390/ijgi6090284, number: 9 Publisher: Multidisciplinary Digital Publishing Institute. Available from: https://www.mdpi.com/2220-9964/6/9/284

[9] What Is Off-Page SEO? A Comprehensive Guide. Available from: https://www.semrush.com/blog/off-page-seo

[10] Hošták, M. SEO analýza vybraného webu [online]. 2013 [cit. 2022-04-29], sUPERVISOR: Martin Sova. Available from: https://theses.cz/id/kzm9b2/

[11] Image SEO: Optimize images for search engines - Seobility Wiki. Available from: https://www.seobility.net/en/wiki/Image_SEO

[12] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. THE SEMANTIC WEB. *Scientific American*, volume 284, no. 5, 2001: pp. 34–43, ISSN 0036-8733, publisher: Scientific American, a division of Nature America, Inc. Available from: https://www.jstor.org/stable/26059207

[13] What is a Knowledge Graph? | IBM. Available from: https://www.ibm.com/cloud/learn/knowledge-graph

[14] WAI-ARIA Roles - Accessibility | MDN. Available from: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles

[15] XML Path Language (XPath). Dec. 2012. Available from: https://web.archive.org/web/20121209085946/http://www.w3.org/TR/xpath/

[16] Krotov, V.; Silva, L. Legality and Ethics of Web Scraping. Sept. 2018.

[17] Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better? Available from: https://lvivity.com/single-page-app-vs-multi-page-app

[18] Markup Language Definition. Available from: https://techterms.com/definition/markup_language

[19] XML introduction - XML: Extensible Markup Language | MDN. Available from: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

[20] Wayback Machine General Information – Internet Archive Help Center. Available from: https://help.archive.org/help/wayback-machine-general-information/

[21] Scrapy 2.6 documentation — Scrapy 2.6.1 documentation. Available from: https://docs.scrapy.org/en/latest/

[22] Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation. Available from: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[23] 8 Best Web Scraping Tools. Section: Web Scraping. Available from: https://hevodata.com/learn/8-best-web-scraping-tools/

[24] Meta Title - Definition + Best Practices - Seobility Wiki. Available from: https://www.seobility.net/en/wiki/Meta_Title

[25] How to use Headings - SEO Best Practices - Seobility Wiki. Available from: https://www.seobility.net/en/wiki/H1-H6_headings

[26] Content is still King - but what is good content? - Seobility Wiki. Available from: https://www.seobility.net/en/wiki/Content_is_King

[27] Web 2.0. Sept. 2021, page Version ID: 20479119. Available from: https://cs.wikipedia.org/w/index.php?title=Web_2.0&oldid=20479119

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2020/2021

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc. Marek Laušman**
Osobní číslo: **I2000048**
Adresa: Horní Ředice 286, Horní Ředice, 53375 Dolní Ředice, Česká republika

Téma práce: Automatizované stažení a analýza webových stránek
Téma práce anglicky: Automated download and analysis of web pages

Vedoucí práce: Mgr. Daniela Ponce, Ph.D.
Katedra informačních technologií

Zásady pro vypracování:

Cíl práce: Charakterizovat značkovací zvyklosti vzhledem k značkovacím doporučením s využitím automatizovaného stahování webových stránek
Osnova:
1. Úvod
2. Cíl práce, metodika
3. Kategorie získávaných informací z webových stránek
4. Informace a jejich automatizované získávání z webových stránek
5. Návrh
6. Implementace
7. Aplikace
8. Závěr
9. Shrnutí a doporučení

Seznam doporučené literatury:

GLEZ-PEÑA, Daniel, et al. Web scraping technologies in an API world. *Briefings in bioinformatics*, 2014, 15.5: 788-797.
CHAULAGAIN, Ram Sharan, et al. Cloud based web scraping for big data applications. In: *2017 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2017. p. 138-143.
LIU, Wei; XIAO, Jianguo. Incremental structured web database crawling via history versions. In: *International Conference on Web Information Systems Engineering*. Springer, Berlin, Heidelberg, 2010. p. 524-533.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: