Deep Web

Projektová Dokumentace

1 Popis projektu

V dnešní době velké množství dat na internetu není přímo přístupné. Často se setkáváme s vyhledávacími formuláři, které je potřeba vyplnit a oni nám na základě toho poskytnou nějakou množinu výsledků. Tyto skryté databáze se v tomto kontextu označují jako deep web, zatímco přímo přístupná data se označují jako surface web - to je část webu, která je standardně indexovaná dnešními vyhledávacími enginy. Předpokládá se, že deep web obsahuje oproti surface webu mnohonásobně více dat.

Ačkoli je možné pomocí těchto formulářů v takovéto skryté databázi vyhledávat, nedostáváme úplnou informaci o podobě databáze. Navíc většina takových formulářových rozhraní vrací pouze malé množství výsledků, řekněme *k-nejlepších* výsledků.

Otázkou tedy nastává, jakým efektivním způsobem můžeme takovou databázi zrekonstruovat ať už ve smyslu získání co největšího množství dat, nebo alespoň provést dostatečně velký vzorek databáze, na základě kterého bychom byli schopni vytvořit dostatečně důvěryhodné závěry o obsahu databáze.

1.1 Formulářové rozhraní

Formuláře mohou být realizovány různými způsoby, ale standardně se setkáváme s klasickými prvky webových formulářů které poskytuje HTML.

Prvně můžeme vybírat pomocí radio button. Radio button může poskytovat Binární volbu (ano/ne) či případně právě jednu volbu nějaké kategorie a to buď ordinální nebo nominální. Na stejném principu fungují formulářové prvky jako datalist čí select, tedy že poskytují právě jednu volbu.

Dalším velmi populárním prvkem je checkbox. Samotný checkbox poskytuje vždy binární volbu typu (ano/ne) ale standardní způsob formuláře obsahuje několik checkboxů které spadají do jedné kategorie, takže tedy můžeme vybrat více možností. V takovém případě vybrání více checkboxů reprezentuje logickou operaci OR.
Standardně formuláře poskytují také prvek typu input, tedy pole, do kterého můžeme vložit nějakou (standardně alfanumerickou) hodnotu. Mimo zmíněné existují i další populární formulářové prvky, jako například slider, který se snaží reprezentovat výběr hodnoty spojité

veličiny, nebo dokonce intervalu. Sliderem, ani žádnými dalšími formulářovými prvky se ovšem v této práci zabývat nebudeme.

Nakonec ještě zmiňme, že dané prvky se častokrát vyskytují v různých skupinách - například skupiny checkboxů - které sice v rámci jedné kategorie reprezentují možnost výběru několika binárních hodnot pomocí operace OR, tak dvě různé skupiny vůči sobě navzájem jsou spojeny pomocí binární operace AND.

1.2 Data sampling a jeho úskalí na deep webu

Při data samplingu se snažíme vytvořit náhodný výběr dat z datasetu. Na základě něj pak můžeme vytvořit statistické výsledky ohledně daného datasetu. Takovýto náhodný výběr je snadný v případě, že máme k dispozici celý data set a existuje velké množství přístupů jak toho dosáhnout. Takový přístup ovšem v případě databáze, která je schovaná na webu za formulářem, nemáme. Navíc často jsme omezení počtem vyhledávání, které můžeme prostřednictvím formuláře zadat. To je dáno především tím, že dnešní webové stránky se brání přetížení, tedy po větším množství dotazů za krátký časový interval může daná stránka přestat přijímat na nějaký čas další dotazy. V následující sekci představíme algoritmus, který se s těmito problém snaží vypořádat, tedy že se snaží poskytnout efektivní řešení poskytující kvalitné náhodné výběry z dané skryté databáze.

2 Způsob řešení

Jako první myšlenka se jeví naivní řešení. Vyzkoušet všechny kombinace dotazů pomocí formuláře, stáhnout tedy tak celou databázi která je dostupná přes formulář, a následně ji využít jakožto data set v požadovaných statických metodách.Takové řešení je ale v praxi často nemožné. Jak bylo popsáno v sekci 1.1, i základní formulářové prvky mohou vytvořit komplikované vyhledávací dotazy. Běžnou praxí navíc je, že formuláře mohou obsahovat až desítky prvků typy checkbox. Vyzkoušení všech kombinací se potom rovná projití 2^n možností, což může být pro výpočetně neúnosné. Musíme navíc brát v potaz, že jeden takový dotaz přes takovýto webový formulář může trvat i několik sekund (více v sekci 5).

2.1 HIDDEN-DB-SAMLER

Zmíněné komplikace vedou na využití nějakého pravděpodobnostního algoritmu. Jeden z takových algoritmů je popsán v [1] a autoři tento algoritmus nazývají HIDDEN-DB-SAMPLER.

Tento algoritmus předpokládá pouze binární atributy ve formuláři a navíc předpokládá, že tyto atributy jsou vzájemně spojeny logickým AND. Základní myšlenkou tohoto algoritmu je vytvořit náhodný průchod binárním stromem jehož uzly reprezentují výběr jednotlivých parametrů. Znamená to tedy, že pokud postupujeme od kořene a jsme na *m*-té hladině stromu, máme nějakým způsobem vybráno *m* binárních příznaků. Hloubka stromu *n* potom tedy reprezentuje celkový počet binárních příznaků. Zaveďme parametr *k* který reprezentuje počet výsledků které chceme aby dané qurey *Q* vrátilo. Potom můžeme algoritmus popsat následujícím způsobem:

1. Vytvořme náhodnou permutaci binárních příznaků (A_1, A_2,...,A_n) , položme query Q = {}, *i* = 1.
2. Náhodně zvolme x_i , $x_i \in \{0, 1\}$ a položme Q = Q AND (A_i = x_i)
3. Nechme vyhledat dotaz Q pomocí formuláře.
4. Pokud je počet výsledků menší než k, vraťme se do bodu 1.
5. Pokud je počet výsledků v2tší než k, položme *i* = *i* +1 a vraťme se do bodu 2.
6. Vraťme *k*-nejlepších výsledků.

Autoři krok algoritmu ještě rozšiřují krok 6. a to takovým způsobem že výsledek přijmou pouze s určitou pravděpodobností, v opačném případě se vrací do bodu 1. Pravděpodobnost přijetí se zvětšuje s tím kolikátá je iterace vnitřního cyklu. Díky tomu jsou schopni přijmout výsledky, které měli dlouho konvergenci s vyšší pravděpodobností, než ty, které byly nalezeny ihned. Tento faktor výrazně zmenšuje zkreslení tohoto samplovacího algoritmu.

V rámci naší implementace jsme tento algoritmus mírně modifikovali kvůli parametrům webového formuláře, přes který probíhal data sampling skryté databáze. Modifikace algoritmu jsou zmíněny v sekci 3.1.

3 Implementace

V tomto projektu jsme se rozhodli zrekonstruovat databázi která se skrývá za formulářem na webové adrese https://knihy.heureka.cz/. Faktem je, že tato databáze-narozdíl od zmíněného předpokladu-poskytuje možnost ji celou projít i bez formuláře, tedy že nevrací pouze *k*-nejlepších výsledků, ale všechny. To ovšem nebyl problém, protože jsme v rámci implementace použili parametr *k*, který algoritmus informoval o tom, že nemá stahovat více než daný počet výsledků v případě, že jich na daný dotaz bylo vráceno více.

Implementace celé aplikace je v jazyce Python 3. Aplikace je postavena ze třech částí. První částí je samotný crawler, který stahuje data z dané webové stránky. Druhá část aplikace reprezentuje modifikovaný algoritmus HIDDEN-DB-SAMPLER (viz. sekce 2.1), který vytváří dane query a v podobě URL ho následně poskytuje crawleru. Poslední částí aplikace je uživatelské rozhraní, kde je možné zobrazit stažená data a také zobrazit statistické informace o těchto datech.

3.1 Crawler

Crawler byl vytvořen s pomocí knihovny Beautifulsoup určené k parsování HTML a XML dokumentů. V rámci implementace bylo třeba prozkoumat strukturu HTML dokumentů na webu https://knihy.heureka.cz/ spolu s tím jaké informace je možné z výsledků daného vyhledávání získat. Vzhledem k tomu, že tento web sumarizuje informace o prodeji knih v internetových obchodech, byli jsme schopni z výsledků zrekonstruovat následující informace: název knihy, autor, nakladatelství, počet stran, žánr, hodnocení, datum vydání a cenové rozpětí.

Vzhledem k tomu, že informace o jednotlivých knihách na tomto webu nejsou často konzistentní - obsahují chyby a nevyplněné informace, tak crawler mimo shany o naparsování daných informací ukládá i nenaparsovanou část informací, která lze za pomoci parsování těžko zrekonstruuovat.

3.2 Samplovací algoritmus

Samplovací algoritmus je velmi podobný algoritmu HIDDEN-DB-SAMPLER. Museli jsme se vypořádat s několika modifikacemi, především proto, že daná webová stránka má formulář který se skládá z několika skupin check boxů, které v rámci jedné skupiny fungují jako binární příznaky spojované logickým operátorem OR, ale skupiny jako celek, jsou spojovány logickém operátorem AND. Mimo to formulář obsahoval textový input.

Jako nejvhodnější způsob jsme zvolili variantu, kdy vnímáme jednu skupinu checkboxů pouze jako jeden příznak A_i, $A_i \in M_i$, kde M_i je množina všech různých výběrů checkboxů ze skupiny i. Tedy $|M_i| = 2^{p_i}$ kde p_i je počet checkboxů ve skupině i. Mimochodem pokud toto čtete tak, tak u nás máte pivo. Jediný rozdíl tedy spočívá v tom že v kroku 2. algoritmu HIDDEN-DB-SAMPLER popsaného v sekci 2.1 nevybíráme $x_i \in \{0, 1\}$ ale $x_i \in M_i$. To znamená, že náhodně zvolíme checkboxy v dané skupině.

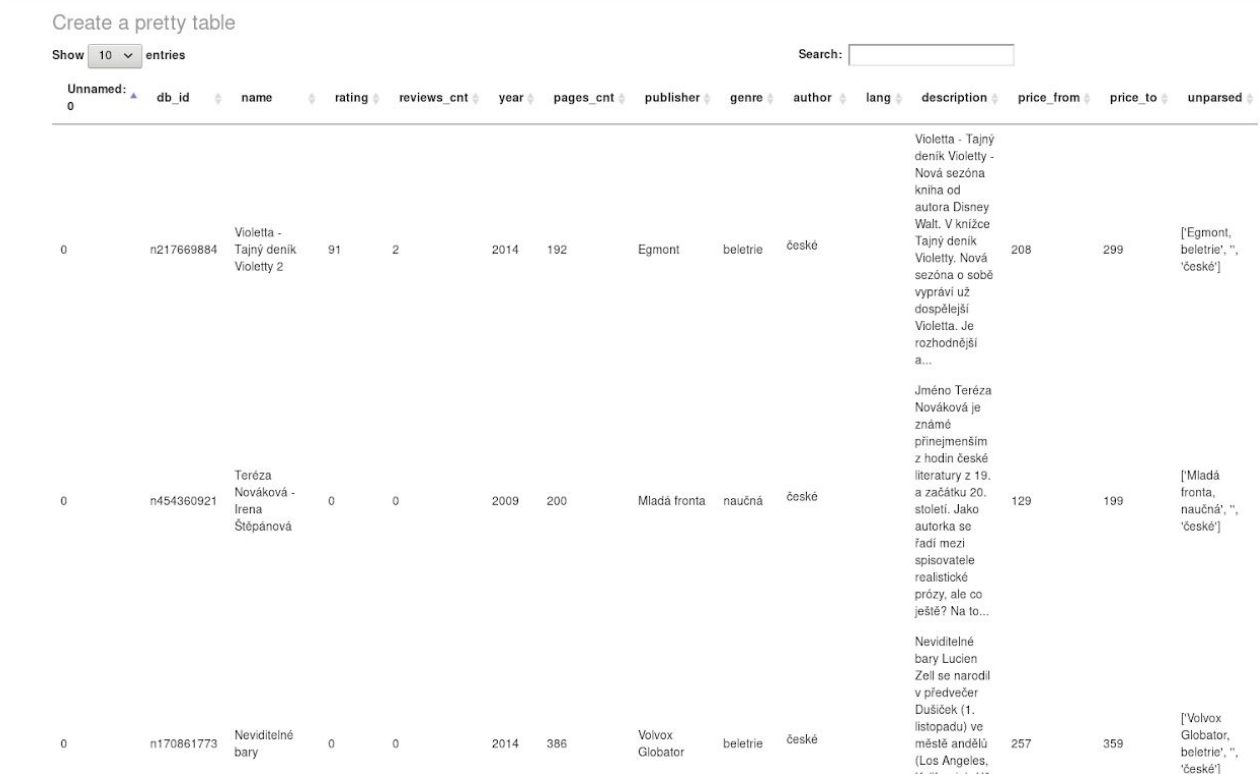
Druhá modifikace, kterou jsme učinili, je slovníkový útok na textový input který se ve formuláři nacházel. Modifikace spočívá v tom, že pokud vyčerpáme všechny checkboxy a stále máme příliš velký počet výsledků, použijeme slovník na tento input. Ve smyslu algoritmu HIDDEN-DB-SAMPLER tedy používáme slovník v momentě, kdy v průchodu dojdeme až do listu.

3.3 Uživatelské rozhraní

Je implementováno ve webovém rozhraní pomocí micro web frameworku Flask. Skládá se z dvou stránek, na té první může uživatel prohlížet předem stáhlé data v tabulce s jednoduchým řazením a vyhledáváním. V té druhé jsou statistiky a to kolik procent databáze se nám podařilo stáhnout a možnost benchmarkingu. Uživatel si zvolí počet výsledků které by chtěl stáhnout a po doběhnutí se mu ukáže čas jak dlouho stahování trvalo.

4 Příklady výstupu

Jak už bylo popsáno v sekci 3.3 uživatel má možnost prohlížet celou databázi co se nám podařilo stáhnout. Tato část GUI vypadá následovně:



Create a pretty table

Show 10 entries Search:

Unnamed: 0	db_id	name	rating	reviews_cnt	year	pages_cnt	publisher	genre	author	lang	description	price_from	price_to	unparsed
0	n217669884	Violetta - Tajný deník Violetty 2	91	2	2014	192	Egmont	beletrie	české		Violetta - Tajný deník Violetty - Nová sezóna kniha od autora Disney Walt. V knížce Tajný deník Violetty. Nová sezóna o sobě vypráví už dospělejší Violetta. Je rozhodnější a...	208	299	['Egmont, beletrie', '', 'české']
0	n454360921	Teréza Nováková - Irena Štěpánová	0	0	2009	200	Mladá fronta	naučná	české		Jméno Teréza Nováková je známé přinejmenším z hodin české literatury z 19. a začátku 20. století. Jako autorka se řadí mezi spisovatele realistické prózy, ale co ještě? Na to...	129	199	['Mladá fronta, naučná', '', 'české']
0	n170861773	Neviditelné bary	0	0	2014	386	Volvox Globator	beletrie	české		Neviditelné bary Lucien Zell se narodil v předvečer Dušiček (1. listopadu) ve městě andělů (Los Angeles,	257	359	['Volvox Globator, beletrie', '', 'české']

Jedná se o velmi jednoduchou tabulku, kde se dá fulltextově vyhledávat. Jedná se o základní funkcionalitu a není tedy možné specifikovat v kterých sloupcích vyhledáváme.

Druhá část GUI je ve formátu statistické stránky.

Statistics

We have got 126053 rows. There are 848346 rows on knihy.heureka.cz right now. We've got 14.86% scrapped.

How many records should i pull?

10000

Start benchmark

Results:

Computing

V této části ukazujeme naší celkovou úspěšnost a možnost již zmíněného benchmarkingu.

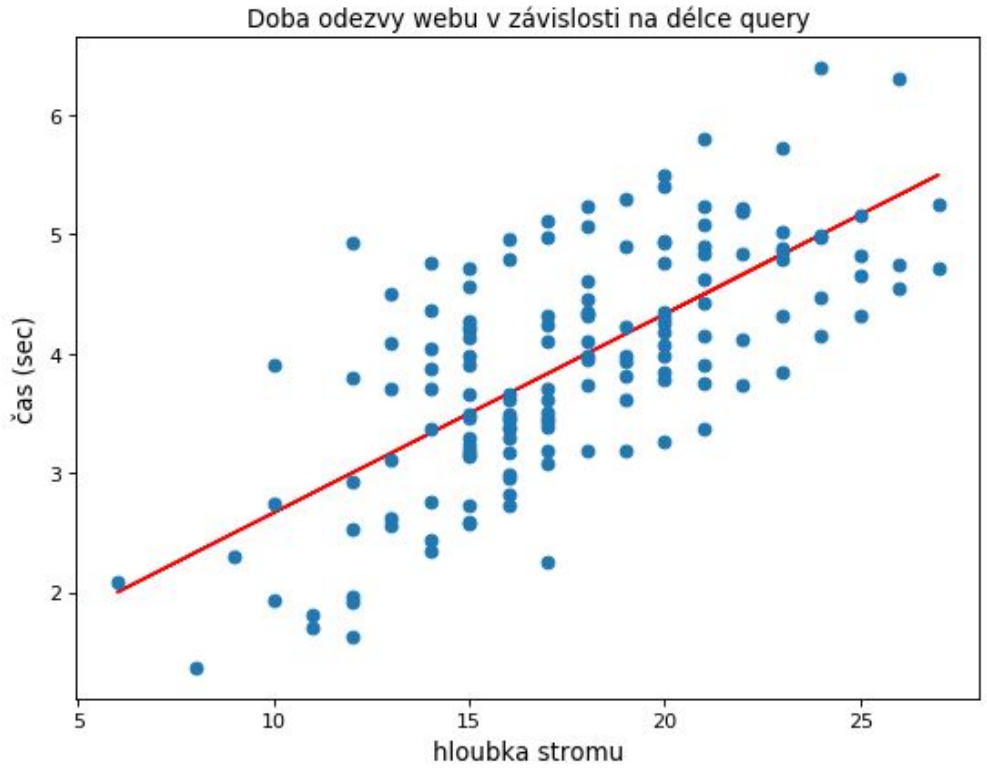
Poslední možnost jak získat naše výsledky je přes .csv soubor. Všechna naše data jsou uložena v .csv, čím se dají využít i nástroje třetích stran, jako například excel a podobně.

5 Experimentální sekce

Na základě naší implementace jsme provedli několik experimentů na webu https://knihy.heureka.cz/. Naším cílem bylo získat co nejpřesnější informaci o tom, jaká doba je potřeba k rekonstrukci celé databáze. Výhodou bylo, že tento web poskytuje informaci o tom, kolik je v databázi položek, takže jsme mohli snadno změřit kolik procent databáze se stáhlo za daný časový interval. Celkový počet záznamu je 848263. Pro všechny experimenty jsme nastavili k=300, tedy nikdy jsme z daného query nezískali více jak 300 výsledků.

5.1 Rychlost odezvy

Jako první jsme se rozhodli změřit rychlost odezvy webu, protože se jedná o nejpomalejší část. Na následujícím grafu je zobrazen vztah mezi délkou query (tedy hloubkou stromu) a odezvou webové stránky. Je patrná lineární závislost (červená přímka). Jedná se 150 náhodných měření. Odezva je velmi dlouhá, proto je zřejmé, že doba potřebná k rekonstrukci databáze bude velmi záviset na době odezvy webu.



Následující tabulka zobrazuje základní statické informace o těchto 150ti náhodných query.

	čas (sekundy)	hloubka stromu
průměr	3.0	13.3
směrodatná odchylka	1.1	8.2
min	1.2	2.0
max	5.6	29.0

Průměrná hloubka stromu je 13 s časem 3 sekundy. Dá se tedy předpokládat, že v nejlepším případě (stejné výsledky se ve vyhledávání neopakují), bude doba potřebná k rekonstrukci databáze závislá především na délce této odezvy a celkovém počtu dat. Vzhledem k tomu že databáze obsahuje 848263 záznamů a web zobrazuje nejvíc 30 položek na jedné stránce. Nejoptimističetější odhad je tedy 848263/30*3 = 84826,3 sekund, což je přibližně 23.5 hodin.

5.2 Ochrana webu proti scrapingu

Valná většina dnešních webů používá nějakou formu ochrany proti častým rychlým opakovaným requestům. https://knihy.heureka.cz/ není výjimkou. Po dosažení limitu dotazů (1000) přestane server odpovídat. Z našich měření vyplývá, že doba po kterou server následně neodpovídá rovnoměrně náhodně v rozmezí 3 až 6 minut. Pokud po tomto intervalu začneme server zatěžovat dalším velkým množstvím dotazů, může tato doba po kterou server neodpovídá stále růst.

Je proto vhodné mezi jednotlivými dotazy nastavit nějaký timeout, abychom se do tohoto stavu nedostali. Z našeho měření vyplývá, že při timeoutu mezi jednotlivými dotazy na 0,5s, můžeme počet dotazů před blokací ze strany webu zvýšit až na 10000. A pokud volíme timeout náhodně v rozmezí 0,5 až 5 sekund, k timeoutu většinou ani nedojde. V případě že ano, můžeme počkat náhodně nějaký časový interval v řádu několika minut a následně v dotazech pokračovat.

Následující tabulka uvádí počet dotazů, mezi kterými je timeout zvolen způsobem popsaným výše a počet blokací ze strany webu.

počet opakovaných dotazů	blokace ze strany webu
500	0
1000	0
10000	0
100000	2

Můžeme tedy říci, že naše schéma timeoutů je voleno takovým způsobem, že k blokaci dochází výjimečně. Podotkněme, že timeout je volen v rozmezí 0,5 - 5 sekund, tedy průměrná doba timeoutu je 2,25 sekundy. Tím se náš odhad ze sekce 5.1 zvětšuje na 155514 sekund, tedy přibližně 43 hodin a 12 minut.

Když si ale uvědomíme, že sice jsme se vyhli většině timeoutů, ale celkovou dobu běhu jsme si prodloužili skoro dvojnásobně. Tak nám přijde omnoho lepším řešením použít timeout 0.5 a každých 10000 dotazů si počkat několik minut než si prodloužit celkovou dobu běhu o několik hodin.

5.3 Duplicity ve ve výsledcích

Protože se velmi často stává, že výsledky na dané query obsahují duplicity, změřili jsme kolik dat je potřeba stáhnout pro to, abychom dostali nějký daný počet unikátních dat. Následující tabulka ukazuje naše výsledky.

počet unikátních dat	počet stažených dat	čas (sekundy)
100	100	13
1000	3427	314
10000	31554	2890

Je tedy patrné, že pouze 30% dat je unikátních. Doba potřebná ke stažení těchto dat přibližně odpovídá našemu odhadu.

5.4 Odhad vs. reálná doba potřebná k rekonstrukci databáze

Na základě sekcí 5.1, 5.2 a 5.3 jsme schopni vytvořit časový odhad k rekonstrukci celé databáze. Ze sekce 5.1 víme, že průměrná doba jednoho dotazu je 3 sekundy, v sekci 5.2 jsme popsali, že mezi jednotlivými dotazy musíme vytvářet timeouty jako prevenci před zablokováním ze strany webu, tato doba je 0,5 sekundy. V sekci 5.3 jsme na základě experimentů zjistili, že musíme stáhnout přibližně trojnásobné množství dat kvůli duplicitám ve výsledcích. Náš předpoklad tedy činí 848263/30*(3+0,5)*3 = 296892 sekund.
To je přibližně 82 hodin a 12 minut.

5.5 Teorie vs. prax

Při výpočtu jsme vynechali několik faktorů a to za prvé, fázi hledání vhodných filtrů, kdy testujeme zda-li můžeme začít stahovat stránku při dané aplikaci filtrů (jestli je tam míň než k výsledků). Tím jsme efektivně vynechali několik desítek tisíc dotazů z celé naší analýzy. Plno z těchto dotazů vrací nulovou hodnotu, to se stávalo hlavně počas slovníkového útoku.

Druhý faktor je, že heuréka se zdá, že si cachuje výsledky to znamená, že pokud uděláme dotaz na nějaký filtr, tak další dotaz na ten samý filtr je rychlejší. To znamená procházení jednotlivých stránek výsledků je omnoho rychlejší a odezva se pohybuje kolem 0,20 - 1,7 sekundy,

Třetí faktor je, že databáze není kompletně vyplněná, to znamená, že tímto způsobem nikdy nejsme schopný vidět celou databázi. Co v důsledku znamená, že za celý běh aplikace jsme viděli kolem 130000 výsledků a reálných unikátních záznamů je dohromady 37712 čím jsme dokázali stáhnout pouze 4,44% záznamů po asi 16 hodinách. Můžeme však předpokládat, že ty nejdůležitější záznamy jsme dokázali najít a stáhnout. A naopak záznamy, které jsme neviděli nemají vyplněné všechny data v databázi, protože nejsou až tak důležité.

6 Diskuze

Dle našeho názoru současné řešení poskytuje dostatečný výkon a spolehlivost pro vzorkování jedné konkrétní stránky a to https://knihy.heureka.cz/. Pro tyto účely je dostačující. Bohužel není dostačující pro celkovou rekonstrukci databáze. Navíc pokud bychom tuto aplikaci chtěli využít na větším množství webových stránek nebo na stránkách, kde databáze obsahují desítky milionů záznamů, bylo by vhodné udělat několik rozšíření.

První vhodné rozšíření by byla větší modulárnost aplikace, tedy aby pro každou webovou stránku stačilo lehce upravit crawler a stránka by mohla být vzorkována. V rámci naší implementace jsme se snažili vytvořit řešení, které by nevyžadovalo příliš velké úpravy pro tuto změnu, tudíž by bylo velmi vhodným rozšířením současné aplikace.

Druhé rozšíření které by bylo vhodné je paralelní běh aplikace na několika zařízeních zároveň. V takovém případě by bylo ale vhodnější jako základ využít některé ze současných řešení vhodných pro danou úlohu, jako například Apache Spark.

7 Závěr

V rámci této práce jsme představili problém skrytých databází na webu. Vysvětlili jsme že je vhodné se zabývat získáváním informací z těchto databází a jejich vzorkování. Uvedli jsme, že díky absenci přímého přístupu to přináší řadu úskalí. Na stranu druhou existuje několik algoritmů které tento problém umí řešit. Jeden z nich jsme implementovali v rámci naší aplikace a prakticky jsme ho využili při vzorkování webové stránky https://knihy.heureka.cz/.

Z našich výsledků je zřejmé, že vzorkování je poměrně snadné, ale stažení celé skryté databáze není reálné (námi použitou implementací pro web https://knihy.heureka.cz/). Problémů je hned několik, ale problém co nás zastihl nejvíce je právě nekompletní databáze a dlouhý čas stahování. První problémem můžeme těžko ovlivnit, pro druhý se nabízí možnost paralelizace nastíněna v sekci 6.