TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

SBĚR, INTERPRETACE A KATALOGIZACE VOLNĚ DOSTUPNÝCH ONLINE DAT

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Michal Štrick**
Vedoucí práce: Ing. Roman Špánek, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal Štrick**
Osobní číslo:	**M12000233**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Sběr, interpretace a katalogizace volně dostupných online dat**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Připravte moduly pro sběr dat dostupných na citačních a vybraných sociálních online zdrojích.
2. Připravte pro stažená data efektivní způsob jejich uložení.
3. Nastudujte dostupné techniky pro zpracování dat (datamining, mining sociálních sítí), které by v datech hledaly skryté souvislosti.
4. Připravte pilotní implementaci z navržených technik a ověřte tím jejich funkcionalitu.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**40–50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Charles Kadushin: Understanding Social Networks: Theories, Concepts, and Findings, OUP USA (19 Jan 2012), ISBN-10: 0195379470, ISBN-13: 978-0195379471**

[2] **Ian H. Witten, Eibe Frank, Mark A. Hall: Data Mining: Practical Machine Learning Tools and Techniques, Publisher: Morgan Kaufmann; 3 edition (3 Feb 2011), ISBN-10: 0123748569 ISBN-13: 978-0123748560**

[3] **B. Eckel: Myslíme v jazyku Java. Knihovna zkušeného programátora. Grada Publishing 2001.**

[4] **G. Lars:HBase: The Definitive Guide.O'Reilly Media; 1 edition (September 20, 2011), ISBN-10: 1449396100, ISBN-13: 978-1449396107**

Vedoucí diplomové práce:	**Ing. Roman Špánek, Ph.D.** Ústav mechatroniky a technické informatiky
Konzultant diplomové práce:	**Ing. Pavel Tyl** Ústav mechatroniky a technické informatiky

Datum zadání diplomové práce:	**10. října 2013**
Termín odevzdání diplomové práce:	**16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Chtěl bych poděkovat všem, kteří se jakýmkoliv způsobem podíleli na vzniku mé práce. V první řadě bych chtěl poděkovat vedoucímu své diplomové práce panu doktorovi. Romanovi Špánkovi za rady, trpělivost a svůj čas. Dále bych chtěl poděkovat panu doktorovi Jiřímu Hnídkovi za poskytnutí místa na školním serveru pro použitou databázi a s pomocí s jejím zprovozněním. Nemalý dík patří i mé rodině, která mě podporovala fyzicky i psychicky a pečovali o mé pohodlí při práci.

V Liberci dne 13. 5. 2014

Abstrakt

Diplomová práce se zabývá vytvořením flexibilní aplikace pro získávání dat z online zdrojů. Konkrétně pak je představen modul pro dolování dat z citačních serverů. Vzhledem k očekávanému množství dat je v práci použita NoSQL databázová technologie. S ohledem na typ záznamů, kdy jsou popsány jak vlastní data sledovaných entit, tak i vztahy mezi nimi, je věnována část práce i popisu vizualizace a dolování dat ze sociálních sítí, konkrétně pak je využita existující implementace Pajek.

Klíčová slova:

Citační servery, HBase, NoSQL databáze, zpracování dat, Pajek

Abstract

The main topic of this thesis is to create a flexible application for retrieving data from online sources. Specifically, it presents a module for data mining from citation servers. Due to the expected amount of data there is NoSQL technology used in this thesis. With regard to the type of records, which are described as data of monitored entities and relationships between them, is given a part of this thesis to vizualization description and social networks data mining, concretely there is used existing implementation Pajek.

Keywords:

Citation servers, HBase, NoSQL database, data processing, Pajek

Obsah

Seznam obrázků a rovnic

1 Cíle práce

Cílem této práce je vytvořit flexibilní aplikaci pro získání určitých dat z online zdrojů. Důraz se pak klade zejména na modularitu této aplikace. Z dostupných online zdrojů dat je pak tato práce zaměřená hlavně na citační servery Scopus a Web of Knowledge, díky modularitě je ale možné připravit moduly i pro jiné zdroje. Dalším cílem této práce je pomocí aplikace zajistit možnou vizualizaci získaných dat s možností jejich dalšího zpracování. Jednotlivé techniky pro zpracování dat pak mohou být realizovány buď přímo ve vytvořené aplikaci, nebo k tomuto účelu může být využita již existující aplikace pro zpracování a vizualizaci dat, přičemž by bylo nutno vytvořit rozhranní mezi získanými daty a použitou aplikací pro zpracování dat.

2 Techniky pro zpracování dat

Internet je zdrojem velkého množství různých dat. Ta mohou být získána například pomocí webového crawleru, nebo mohou být získávána staticky z předem známých zdrojů. Následně je nutné data zpracovat pomocí různých analýz. Typ analýzy záleží na typu dat. Pro některá data je vhodné použít například shlukovou analýzu, z jiných lze vytvořit graf a ten následně analyzovat pomocí různých grafových algoritmů. Pro složitá data je pak vhodné použít neuronové sítě. Obecnými postupy od získání dat po jejich interpretaci se pak zabývá Data mining.

2.1 Data mining

Data mining, neboli vytěžování informací, je skupina metod sloužící ke zpracování dat a získání informací, které jsou v nich obsažené. Je založen na statistice a používá se v komerční i nekomerční sféře. Jedná se o analytický proces, který prohledá data (obvykle velké množství dat) a hledá v nich souvislosti, pomocí nichž dokáže následně zpracovávat nové soubory dat.

Nástrojem data miningu je i předpovídání následujícího vývoje na základě získaných vlastností. To je vhodné například pro předpověď počasí či odhad vývoje cen na burze.

Termín *Data mining* byl představen až v devadesátých letech minulého století, ale vývoj v oblasti dolování informací je mnohem starší. Vychází hlavně z klasické statistiky, umělé inteligence a strojového učení.

Počátky data miningu byly různorodé, což vedlo k zavedení standardizovaných postupů. Vznikly tak dvě hlavní obecné metodologie: SEMMA a CRISP-DM ([10]). Základní postupy jsou však u obou metodologií dosti podobné, proto bude podrobněji popsána pouze metodologie SEMMA.

2.1.1 SEMMA

SEMMA je anglická zkratka *Sample, Explore, Modify, Model, Assess* ([12]). Jedná se o sérii kroků vyvinutých firmou SAS, jedním z největších poskytovatelů statistického softwaru.

Sample, neboli vzorkování, představuje krok redukce velkého množství dat na takovou velikost, která umožňuje rychlou manipulaci pro zvýšení výkonu. Výběr těchto

reprezentačních dat je založen na statistice. Pro sestavení modelu, který bude redukovat původní velkou množinu dat, se doporučuje vytvořit 3 sady dat:

- Trénovací data, která slouží k vytvoření modelu
- Hodnotící data, která slouží pro posouzení výsledků
- Testovací data, která důkladněji posoudí, jak dobře model funguje

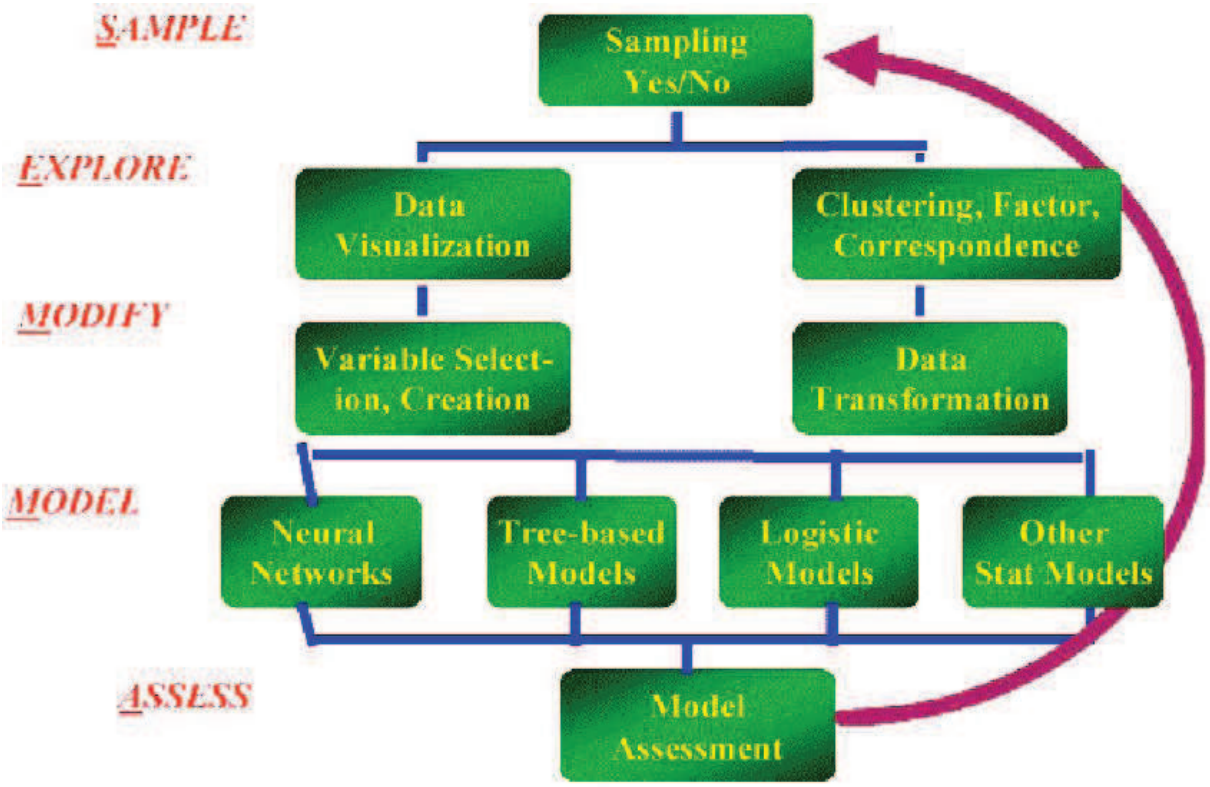
Explore, neboli průzkum, představuje krok, ve kterém dojde k prohledání dat a hledání případných anomálií a skrytých souvislostí. Pro tyto účely slouží statistické techniky jako např. faktorová analýza či shlukování. Může se jednat například o zasílání nabídek zákazníkům vybraných na základě toho, co si v daném obchodu dříve objednávali.

Modify, neboli úprava, představuje krok sloužící k úpravě vstupních dat. Na základě výsledků předchozího kroku zde dochází k manipulaci s daty, jako je rozdělování zákazníků do skupin a význačných podskupin. Také se může jednat o nalezení odlehlých objektů a následně je přiřadit k těm, které jsou jim nejpodobnější. Tento krok je také nutno provést, když dojde k úpravě zkoumaných dat, neboli dojde ke změně struktury zkoumaných objektů.

Model, neboli modelování, představuje krok sloužící k automatickému prohledání skupin dat z předchozího kroku. Mezi modelovací techniky patří například neuronové sítě, stromové modely, logistické modely a další statistické modely jako jsou analýza časových řad či analýza hlavních komponent. Každý model je vhodný pro jiné případy a volba správného modelu záleží na zpracovávaných datech.

Assess, neboli posuzování, je poslední krok, jehož účelem je posouzení výsledků z celého procesu dolování dat. Na jeho základě může dojít k úpravě modelu, podle kterého se redukuje množina vstupních dat (v prvním kroku této metody), popřípadě se upraví zbylé kroky k dosažení lepších výsledků.

Jak již z popisu jednotlivých kroků vyplývá, jedná se o velmi obecný popis metod, které lze uplatnit v mnoha oborech. Kroky této metody se provádějí sekvenčně a cyklicky, jak znázorňuje Obrázek 1. Dochází tak ke stálému zpracovávání dat, kdy se výsledky mohou měnit (resp. aktualizovat) v závislosti na nových datech, která během jedné iterace přibyla. Navíc může docházet i k postupné úpravě jednotlivých kroků a tím i k zlepšení získaných výsledků.



Obrázek 1: kroky metodologie SEMMA (Zdroj: [7])

2.2 Crawlování webu

Webový crawler může být nástrojem pro získávání dat pro další zpracování. Jedná se o relativně jednoduchý program, který automaticky prohledává webové stránky a může je i dále zpracovávat, jako například vytvořit indexy nad jejich obsahem, což je typické zejména pro crawlery internetových vyhledávačů (Google, Yahoo, Seznam, Centrum, ...).

Pokud má crawler za úkol prohledávat různé webové stránky na internetu, musí se předpokládat, že poběží hodně dlouho. Jako první se mu musí dát nějaký startovní bod, ze kterého bude vycházet. Obvykle se jedná o seznam hodně často navštěvovaných a populárních internetových stránek. Crawler následně zadané stránky stáhne, zpracuje a získá všechny odkazy, které ve stažených dokumentech našel. Tyto odkazy pak nastaví jako další zdroje a celý proces opakuje, dokud má nějaké zdroje ke zpracování. Může mít nastavená i jistá omezení, jako například seznam internetových adres, které nemá zpracovávat. Každý zdroj by měl prohledávat pouze jednou, vyhne se tak možnému zacyklení na určité skupině zdrojů, které na sebe navzájem odkazují.

Každý internetový vyhledávač má pak různé zaměření. Může se jednat o obecné zaměření, které mají například Google, bing, Yahoo!, Altavista, Lycos a další. Dále však

mohou být zaměřené na hledání pracovních nabídek, her, věcí lékařského zaměření, multimedií, zpráv, vzdělání, map, e-shopů a mnoho dalších (podrobněji viz [15]). Také mohou být rozděleny podle jazyka, ve kterém vyhledávají. Většinou jsou však nezávislé na použitém jazyku (indexují stránky ve všech jazycích). Každý vyhledávač má pak své vlastní způsoby, jak indexovat obsah stránek. Podle [8] Google indexuje všechna důležitá slova, pro angličtinu pak vynechává slova jako například *a*, *an*, *the*. Lycos pak indexuje pouze nadpisy, nejčastěji používaná slova a prvních 20 řádků textu.

Crawlery se dají použít i k jiným účelům, než je prohledávání webu. Často se používají pro kontrolu html syntaxe a kontrolu funkčnosti všech odkazů. Pomáhá tak zlepšit kvalitu webu ze strany jejího majitele, pokud tuto metodu používá. Tyto crawlery jsou pak jednoduché a relativně rychlé. Dají se koupit, nebo se může použít volná verze, jako například open-source Crawler4j od Google napsaný v Javě, u kterého si však konkrétní funkce musí programátor napsat sám.

2.3 Shluková analýza

Jedna z nejpoužívanějších metod pro analýzu dat v data miningu je shluková analýza, která má za úkol jednotlivé zkoumané objekty přiřadit do shluků na základě vlastností těchto objektů (viz [11]). Základní přístup shlukové analýzy je takový, že každý objekt je jednoznačně zařazen do jednoho shluku. Vzniká tak několik disjunktních shluků (žádné dva shluky nemají společný žádný objekt). Existují však případy, kdy je jeden objekt přiřazen do více shluků. Vzniká tak binární informace o tom, zda je objekt přiřazen k danému shluku, či nikoliv. Pokud bychom sestavili tabulku, jejíž řádky jsou jednotlivé objekty a sloupce jednotlivé shluky, pak by tabulka obsahovala pouze dvě hodnoty, a to 1 (objekt je ve shluku) a 0 (objekt není ve shluku). Analýza může jít však ještě hlouběji, kdy binární informaci změníme na spojitou, značící příslušnost objektu v daném shluku. V tabulce by pak byla čísla z intervalu <0;1>. Součet těchto hodnot pro každý objekt se pak musí rovnat jedné.

Pro přiřazení objektu do shluku je však nejprve stanovit porovnávání podobnosti dvou objektů. Shlukování založené na měření podobnosti se nazývá *konvenční*. Podobnost dvou objektů je funkcí vlastností těchto objektů. Druhá metoda je založena na konceptuální soudržnosti, která je funkcí vlastností objektů, popisného jazyka a okolí. Popisný jazyk je způsob, jakým jsou popsány skupiny objektů, okolí je pak množina vzorů. Tato metoda je tedy založená na tom, že máme k dispozici charakteristiku shluků, do kterých mají být objekty zařazeny.

Předpokladem shlukování je obvykle zadání počtu shluků, do kterých se mají objekty rozřadit. V mnoha případech však o počtu shluků nemá uživatel žádnou informaci. Proto se provádí shlukování pro různé počty shluků a ze získaných výsledků se zjistí, jaký je jejich optimální počet. Jako příklad postupů, kterými lze získat jednoznačné přiřazení, lze uvést metodu *k*-průměrů a její modifikace (*k*-medoidů, *k*-modů, *k*-histogramů). Metody *k*-průměrů a *k*-medoidů jsou určeny pro objekty s vlastnostmi tvořenými kvantitativními (numerickými) proměnnými. Metody *k*-modů a *k*-histogramů jsou zase určeny pro objekty s vlastnostmi tvořenými nominálními proměnnými (lze pouze určit, zda jsou stejné nebo různé).

2.3.1 Metoda *k*-průměrů

Tato metoda vychází z počátečního rozdělení do *k* shluků (*k* je předem zadané). Nejdříve se zvolí k centroidů (k tomu jsou různé metody, jako například použít k prvních objektů). Poté je pomocí výpočtu euklidovské vzdálenosti každý objekt přiřazen k nejbližšímu centroidu. Pro každý shluk je vypočítán nový centroid, kterým je *m*-rozměrný vektor průměrných hodnot vlastností jednotlivých objektů v daném shluku (*m* reprezentuje počet vlastností objektů). Opět se zkoumají vzdálenosti objektů od jednotlivých centroidů. Pokud je objekt blíže jinému centroidu, přesune se do shluku bližšího centroidu. Celý proces se opakuje, dokud dochází k přesunům.

2.3.2 Metoda *k*-medoidů

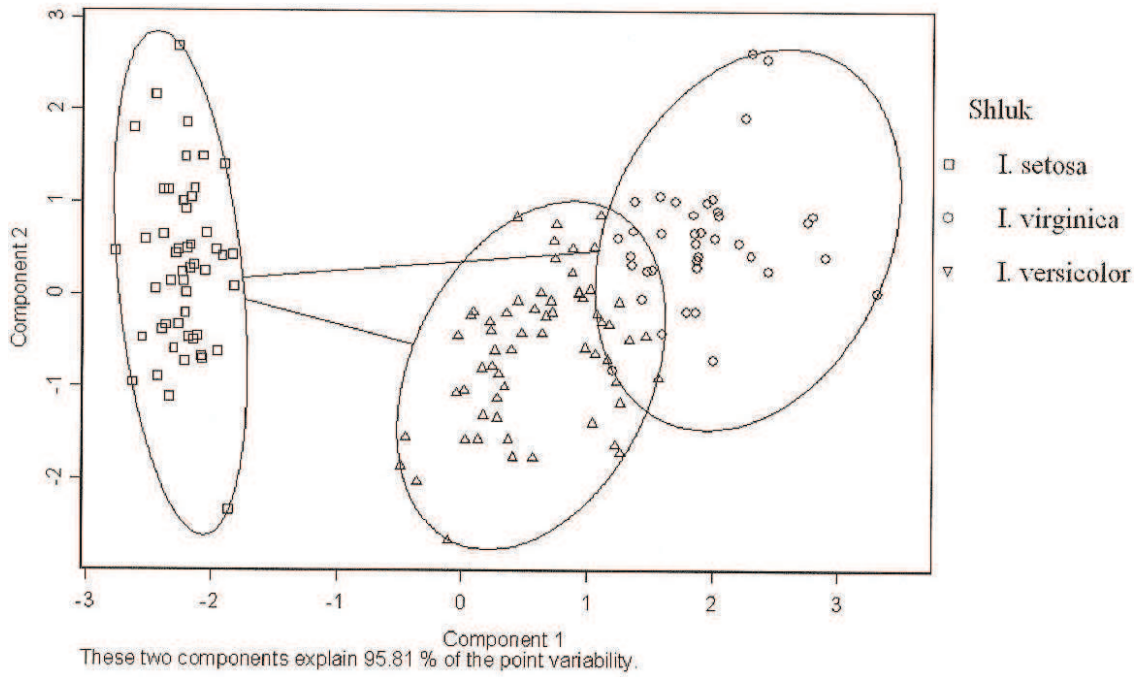
V této metodě je pro každý shluk zjištěn medoid, což je konkrétní objekt ze shluku. Je určen tak, aby součet vzdáleností jednotlivých objektů ve shluku byl minimální. Poté se zkoumají jednotlivé objekty, ke kterému medoidu jsou nejblíže. Pokud jsou blíže k medoidu z jiného shluku než v kterém se objekt nachází, přesunou se do shluku bližšího medoidu.

V dalších iteracích jsou medoidy stanoveny minimalizací funkce, která je součtem vzdáleností jednotlivých objektů od medoidů stejného shluku. Pokud označíme *i*-tý objekt jako x_i a medoid v *g*-tém shluku, k němuž je přiřazen *i*-tý objekt, jako $m_{g,i}$, pak jsou medoidy určeny tak, aby bylo dosaženo minimum funkce v Rovnice 1. Celý postup je opakován tak dlouho, dokud klesá hodnota této funkce.

$$f = \sum_{i=1}^{n} D(x_i, m_{g,i})$$

Rovnice 1: Výpočet funkce pro metodu k-medoidů

Obrázek 2 znázorňuje příklad z [11]. Jedná se o 150 vzorků tří druhů kosatce (konkrténě *setosa*, *versicolor*, *virginica*). U každého druhu bylo provedeno 50 měření délek kališního a okvětního lístku (v cm). Na to byla následně použita metoda *k*-medoidů. Jak je na obrázku vidět, u dvou shluků (vyjadřující druhy *versicolor* a *virginica*) došlo k překrytí a tím i k chybnému zařazení některých vzorků k příslušnému shluku.



Obrázek 2: Vytvoření shluků metodou *k*-medoidů (Zdroj: [11])

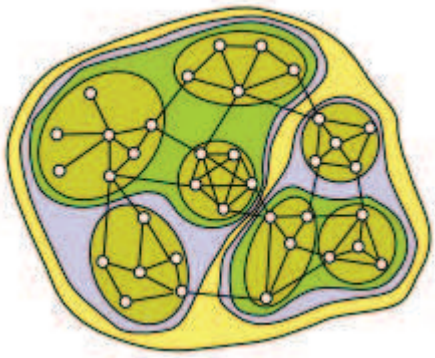
2.4 Analýza sítí

Základní strukturou pro reprezentaci sítí je orientovaný graf. Síť je definována jako uspořádaná čtveřice S = *(V, E, P, W)*, kde *V* je množina vrcholů a *E* je množina orientovaných hran, *P* je ohodnocení vrcholů a *W* je kladné ohodnocení hran grafu (viz [2]).

V závislosti na tom, co daná síť reprezentuje, se mohou aplikovat různé algoritmy na analýzu sítě. Může se jednat například o výpočet skóre grafu, maximálního toku sítí, hledání cest a kružnic různých i požadovaných délek. Při velikosti dnešního výpočetního výkonu není problém tyto údaje zjistit u malých grafů. Problém však nastane, když je potřeba zpracovat velké sítě, které mají tisíce nebo i miliony vrcholů. Hlavními důvody jsou časová náročnost a fakt, že se daná síť celá nevejde do operační paměti. Vizualizace velkých sítí je pak také náročná a hodně nepřehledná. Z těchto důvodů se provádí dekompozice velkých sítí následované dalším zpracováním za účelem zjednodušení.

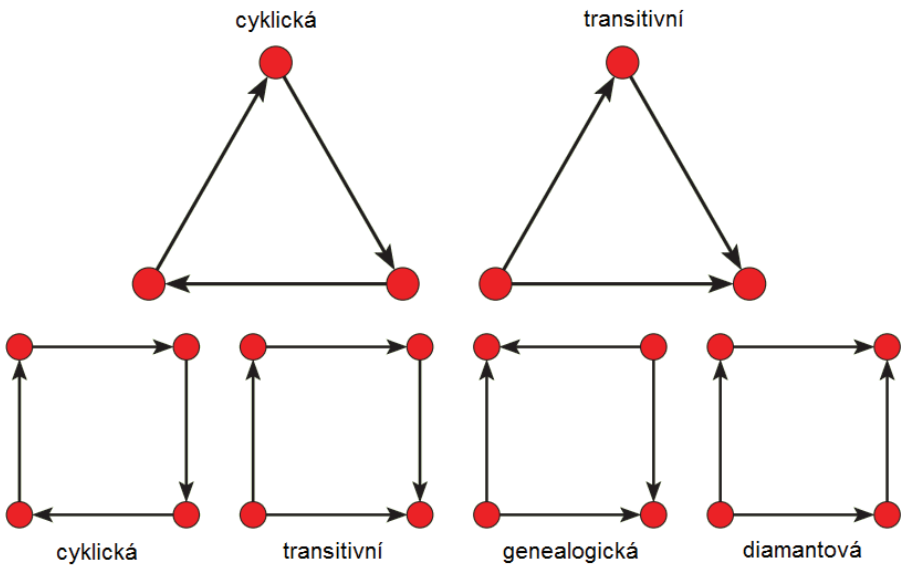
2.4.1 Zjednodušení velkých sítí

Dekompozice sítě se provádí vytvořením shluků v síti například na základě nalezení okolí důležitých vrcholů. Důležitými vrcholy mohou být například vrcholy s velkým stupněm (počtem hran, které do vrcholu vstupují či z něj vycházejí). Do jednoho shluku pak mohou být přiřazeny důležité body a jeho bezprostřední sousedi. Z těchto shluků lze pak vytvořit další síť, kde jednotlivé vrcholy reprezentují jednotlivé shluky. Dekompozice pak lze aplikovat i na takto vzniklé sítě. Obrázek 3 znázorňuje vícenásobnou dekompozici neorientovaného grafu.



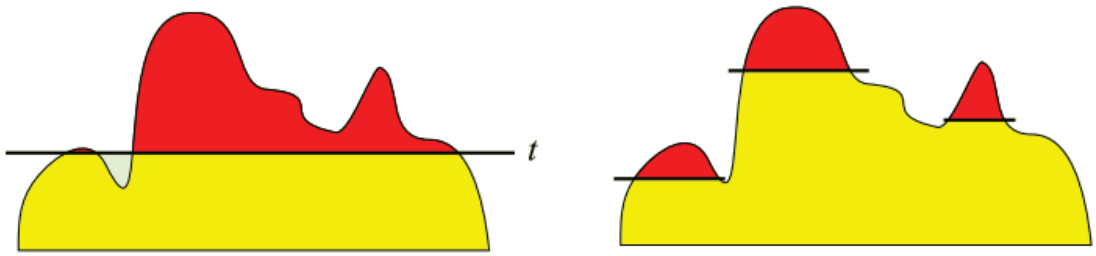
Obrázek 3: Rozklad grafu pomocí shluků (Zdroj: [2])

Velice účinná metoda zjednodušení sítě je ořezání hran a vrcholů. Pomocí nastavení parametru prahu pro hodnoty hran a prahu pro hodnoty vrcholů lze pak odstranit vrcholy a hrany, které se nacházejí pod úrovní prahu. Pokud nejsou ohodnocení hran a vrcholů zadaná, je potřeba je před použitím ořezu spočítat. To se dá například spočítat metodou *k*-kružnic (anglicky *k*-rings), kde *k* definuje délku kružnice. V síti se pak naleznou všechny kružnice délky *k* a jednotlivé hrany a vrcholy pak získají ohodnocení podle toho, v kolika kružnicích se vyskytují. Zde je nutné pamatovat na to, že v orientovaném grafu existuje více typů kružnic dané délky. Typy kružnic délky 3 a 4 znázorňuje Obrázek 4.



Obrázek 4: typy kružnic v orientovaném grafu délky 3 a 4 (Zdroj: [2])

Pokud zobrazíme předem dané nebo vypočítané ohodnocení hran a vrcholů do diagramového grafu, kde na ose *x* jsou jednotlivé objekty (vrcholy nebo hrany) a na ose *y* je ohodnocení daného objektu, můžeme použít další metodu k ořezání sítě, která se nazývá ostrovy. Existuje více způsobů, jak síť rozřezat na ostrovy. Dva z nich znázorňuje Obrázek 5. Jedná se o velmi obecný a přitom efektivní způsob, jak najít důležité podsítě v dané síti (viz [2]).



Obrázek 5: Rozřezání ohodnocených hran/vrcholů na ostrovy (Zdroj: [2])

3 Technické prostředky

Pro dosažení cílů této práce a splnění zadání je nutné si nejprve definovat dostupné technické prostředky pro tuto práci.

3.1 Programovací jazyk

Jako první bylo nutné si zvolit programovací jazyk. Z toho se poté odvíjela základní struktura programu. Java například umožňuje tvorbu a implementaci rozhranní, kdežto C++ umožňuje vícenásobnou dědičnost. Já jsem pro tuto práci zvolil programovací jazyk Java, jako vývojové prostředí jsem zvolil Netbeans.

3.2 Zdroje dat

Existuje mnoho zdrojů digitálních dat v různých formátech. Může se jednat o čistý text, formátovanou html stránku, soubor videa či binární soubor. Tato práce se zaměřuje na html dokumenty z internetu, které si lze prohlížet v internetovém prohlížeči. Tyto zdroje se rozdělují do několika skupin, některé z nich jsou zmíněny níže.

3.2.1 Internetové vyhledávače

Z obecných internetových vyhledávačů můžeme jednoduchým parserem získat webové adresy webových stránek, které podle daného vyhledávače nejlépe odpovídají hledanému textovému řetězci. V České republice patří mezi nejpoužívanější internetové vyhledávače Google a Seznam.

Vyhledávač Seznam lze snadno použít díky nízké úrovni detekce bota, tedy robotického vyhledávání. Stačí mu pouze do hlavičky dotazu podstrčit pár údajů, jaké odesílá nějaký známý webový prohlížeč. Na jeden dotaz jsme schopni získat až 20 odkazů, které podle něj vyhovují námi hledanému textovému řetězci.

Google naproti Seznamu má mnohem lepší detekci robotického vyhledávání. Na druhou stranu Google pamatuje na programy, které potřebují jeho služby internetového vyhledávání. K dispozici je API od Googlu napsáno v Javě, která umožňuje snadné vyhledávání na internetu. Jediný problém tohoto API je ten, že zdarma je maximálně 100 vyhledání za den, jinak se musí platit.

Nad získanými webovými adresami se následně může spustit crawler, který stáhne příslušný html dokument a k tomu všechny další html dokumenty, na které ten původní

odkazuje a které jsou na stejné doméně. Získané html dokumenty pak mohou být zpracovány obecným parserem založeným například na četnosti slov či hledání klíčových slov ať už podle html tagů (nadpisy H1 apod.), nebo podle předzpracovaného CSS (kontrola, zda je text viditelný, získání textu s největším písmem apod.).

3.2.2 Sociální sítě

V dnešní době se hodně rozmáhají sociální sítě, na nichž lze díky tomu získat spoustu informací. Některé sociální sítě jsou přístupné i bez nutné registrace účtu. Cesta k získání dat z těchto serverů pak bývá celkem snadná. Jedná se zejména o twitter, myspace a lide.cz.

Na druhé straně jsou zde i sociální sítě s povinnou registrací, kde je nutné být přihlášen pro prohlížení obsahu. Tyto servery pak mají zpravidla nějaký systém pro správu viditelnosti osobních údajů, kde lze omezit množinu registrovaných lidí, kteří mohou prohlížet vaše údaje. Do této skupiny patří například facebook a google+.

3.2.3 Citační databáze

Citační databáze jsou databáze článků publikovaných v časopisech a příspěvků v konferenčních sbornících. Mezi nejčastěji používané citační databáze patří v České republice SCOPUS a WOS (Web of Science). Tato práce se bude zabývat právě touto skupinou zdrojů dat. Jejich velkou nevýhodou je však fakt, že přístup do nich je omezen. Některé jsou přístupné pouze s univerzitní IP adresou, nebo vyžadují přihlášení na registrovaný účet se speciálními právy přidělovanými univerzitní knihovnou.

Další nevýhodou je, že některé citační zdroje neumějí pracovat s diakritikou. Proto je lepší nepoužívat diakritiku ani v těch serverech, kde je možné ji použít. Vyhneme se tak problémům s porovnáváním výsledků mezi jednotlivými servery.

3.3 Uložiště dat

Získaná data, ať už jsou jakákoliv, by se měla někam ukládat, aby se nemusela pokaždé znovu získávat (stahovat a parsovat). Mohou se ukládat například do databáze či formátovaného souboru.

3.3.1 Databáze

Existuje velké množství druhů databází, kde každá je vhodná k jinému účelu. Asi nejvíce používané jsou dnes relační databáze, je zde ale i široká škála dalších databází, v poslední době je například velký rozmach NoSQL databází.

Základem každé relační databáze je tabulka (relace), která je složena z předem definovaných sloupců (atributů) a jednotlivých řádků identifikovatelných primárním klíčem, který je unikátní v rámci celé tabulky. Databázi pak tvoří soubor různých tabulek, které jsou mezi sebou různě provázány. Tyto databáze jsou vhodné pro aplikace kancelářského typu, nejsou však již tolik vhodné pro práci s velkým množstvím dat. Důvodem je zejména nedostatečná rychlost čtení a zápisu.

Pro velké a rychlé databáze jsou vhodnější NoSQL databáze. Pojem NoSQL, neboli Not Only SQL, označuje velké množství nerelačních databází s různým zaměřením. Oproti relačním databázím jsou sice méně robustní, pro některé účely jsou mnohem rychlejší. NoSQL databáze neřeší věci jako je referenční integrita či uživatelská oprávnění, výsledkem je pak vysoký výkon při zápisu a čtení. Jsou tak vhodné pro velké objemy dat, takže jsou vhodné i pro tuto práci, která je mimo jiné na získávání velkého množství dat zaměřená. Podrobnější rozebrání a volba NoSQL databáze pro tuto práci viz kapitola 4.2.

Existuje ještě mnoho druhů databází, jako například objektové, které vycházejí z objektově orientovaného programování, deduktivní databáze vycházející z logického programování, či temporální databáze, které jsou časově závislé, takže lze od sebe data odlišit například i podle toho, kdy byla přidána. To může být vhodné například pro burzovní aplikace. Pro tuto práci však nejsou vhodné, proto se jimi podrobněji zabývat nebudu.

3.3.2 Další možnosti ukládání dat

Data mohou být ukládána i přímo do binárních souborů či textových souborů založených například na xml. To ale vyžaduje přístup na nižší úrovni než při použití databáze, navíc se data většinou uloží jen na lokální uložiště (např. pevný disk). Pomocí bitové serializace a deserializace, které slouží k převedení obsahu paměti aplikace na proud bytů a zpět (tento proud lze uložit do binárního souboru), lze například uložit stav aplikace před jejím ukončením a následně tento stav obnovit po jejím opětovném spuštění.

4 Realizace

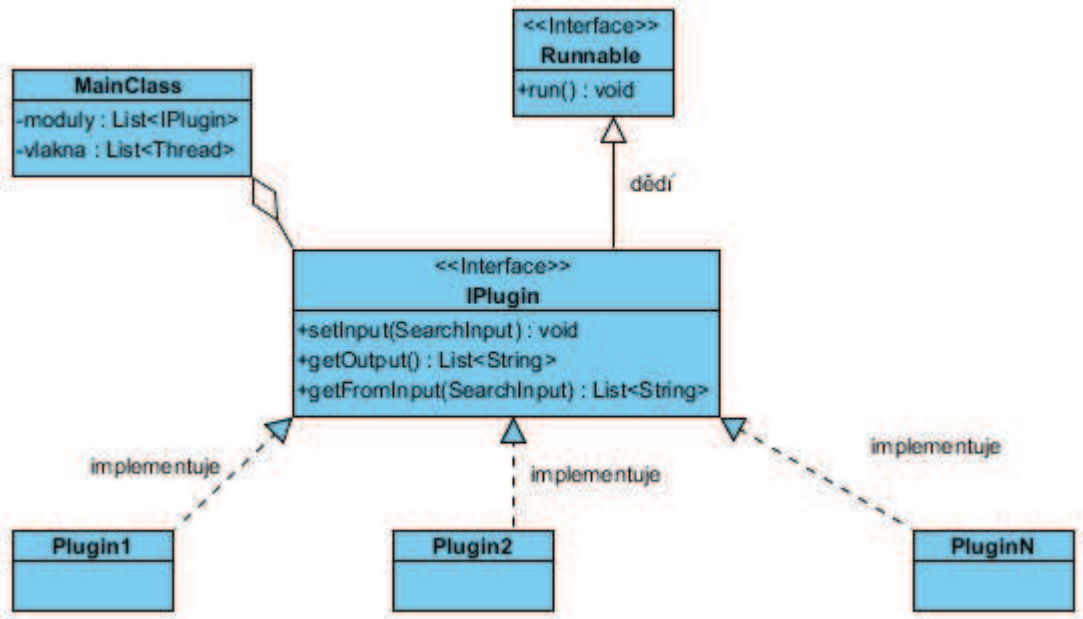
Před samotnou realizací aplikace je třeba si určit přesné požadavky a cíle, které musí aplikace splňovat. To je zejména požadavek na snadnou rozšiřitelnost o další zdroje dat, která bude aplikace získávat, a aby data byla přístupná z více míst, ne jen lokálně. Bude tedy nutno vybrat databázi, do které se budou data ukládat, a také najít server, na kterém databáze poběží.

4.1 Moduly pro sběr dat

Před získáním dat je nutné si určit zdroje, ze kterých se budou data získávat, a přesná data, která se z těchto zdrojů vyberou. Pak je třeba určit postup, jak se k požadovaným datům na daném zdroji dostat. Tato práce je zaměřená na data z citačních serverů *Web of Knowledge* a *Scopus*, konkrétní data jsou jména osob, které osoba zadaná uživatelem aplikace citovala. Jsou ale předpřipraveny i moduly na jiné zdroje.

Poznámka: Jednotlivé moduly jsem realizoval jako třídu se slovem *plugin* v názvu, která může mít i několik parserů. Plugin a parser jsou dohromady jedním modulem, jeden plugin však nemusí mít žádný parser. Tyto parsery jsou navíc čistě záležitostí pluginů, proto pro následující text budou považována slova *modul* a *plugin* za ekvivalentní.

Obecným úkolem jednotlivých modulů je získat data, která budou následně uložena do databáze. Obecně platí, že ke každému zdroji může existovat i více modulů. Každý modul má pak právě jeden zdroj, se kterým pracuje a ze kterého se snaží dostat právě jednu množinu dat stejného typu (například jména herců u zadaného filmu ve filmové databázi, ale na jména scénáristů už je potřeba druhý modul). Pro snadné přidávání nových modelů v budoucnu je tak vhodné využít polymorfismu, kdy nadřazená třída bude obsahovat dynamické pole či spojový seznam takového datového typu, který dědí (či v případě rozhraní implementují) všechny moduly. Jednotlivé moduly jsou však na sobě nezávislé a tak může být každý modul spuštěn ve vlastním vlákně. To pak urychlí proces získávání dat díky tomu, že aplikace poběží paralelně, resp. pseudoparalelně. Obrázek 6 pak znázorňuje pomocí diagramu tříd UML společnou strukturu jednotlivých modelů (v obrázku označených jako *pluginy*) a jejich použití v nadřazené třídě (v obrázku jako *MainClass*). Nadřazená třída pak obsahuje spojový seznam typu *IPlugin*, což je předek všech pluginů. V tomto spojovém seznamu jsou pak instance konkrétních pluginů (*Plugin1* až *PluginN*). Ze struktury rozhranní *IPlugin* lze také vyčíst, že jednotlivé pluginy mohou být spouštěny v jednom vlákně (sekvenčně) i ve vlastním vlákně (pseudoparalelně).



Obrázek 6: Diagram tříd UML struktury použití polymorfismu pro různé pluginy

S vedoucím práce jsme vybrali zdroje, ze kterých bude výsledná aplikace stahovat určitá data, a zdroje, pro které budou jednotlivé moduly pouze předpřipraveny pro získání dat, která zatím nejsou konkrétní. Budou tak připravené do budoucna, ale v rámci této práce nebudou dále využité. Jednotlivé zdroje jsou popsány v následujících podkapitolách.

4.1.1 Citační servery

Zdrojem dat, na který je tato práce zaměřená, jsou citační servery *Web of Knowledge* a *Scopus*. Jako požadovaná data byl určen seznam osob, které ve svých publikacích citovala osoba zadaná na vstupu. Základní postup je tedy pro oba servery stejný. Nejdříve se musí vyhledat publikace zadané osoby, k těmto publikacím je třeba najít seznam referencí a u jednotlivých referencí zjistit jejich autory. To vše však musí proběhnout v co nejmenším počtu kroků.

Vzhledem k tomu, že se jedná o dva různé zdroje, je nutné zjistit, jaký je vyžadovaný formát vstupu a v jakém formátu jsou výstupy. Web of knowledge má problémy s diakritikou, takže jednotlivá jména na vstupu je vhodné zbavit háčků a čárek, popřípadě i jiných diakritických znamének. Písmeno *é* se pak přepíše na *e*, písmeno *ř* na *r* a podobně. Scopus má zase většinu jmen ve formátu, kde křestní jméno je reprezentováno pouze prvním písmenem následovaným tečkou, mezerou a příjmením. Pokud má osoba více jmen, jsou tato další jména zadána různě i v rámci jednoho serveru. Někdy je zde pouze první písmeno dalšího jména, jindy je napsáno celé a někdy není napsáno vůbec. Proto jsem pro vstupy a výstupy zavedl následující omezení:

- Všechny znaky jména jsou malými písmeny
- Žádný znak jména nemá diakritické znaménko
- Jméno obsahuje pouze první znak křestního jména následovaný tečkou, mezerou a celým příjmením

Jméno *Lev Nikolajevič Tolstoj* pak bude po přeformátování dle výše zmíněných omezení psáno jako *l. tolstoj*, jméno *Roman Špánek* je pak přeformátováno na *r. spanek*. Dojde tak sice k částečné ztrátě informace a možnému nalezení špatných výsledků, je to však nutné pro zavedení jednotnosti mezi oběma servery.

Web of Knowledge používá pro svou činnost javascript, takže je nutné pro přístup k datům na tomto serveru zvolit nástroj, který je schopen pracovat s javascriptem, nejlépe nějaký webový prohlížeč ovládaný přímo kódem aplikace a pracující v neviditelném režimu, tedy že nebude mít fyzicky otevřeno žádné okno, ve kterém se bude daná stránka zobrazovat. Pro tento byl použit nástroj *HtmlUnit*. Jedná se o sadu knihoven v Javě, které emulují webový prohlížeč. Podle specifikací pak jako engine pro javascript používá *Rhino* od Mozilla Foundation. Ten je zde však trochu omezen, například nezná funkce pro zjišťování rozměru okna (protože žádné okno nemá). Samotný postup k nalezení požadovaných výsledků je pak celkem jednoduchý, na serveru *Web of Knowledge* ale zdlouhavý. Nejdříve je nutno vyplnit jméno hledaného autora. Po nalezení výsledků (publikace autora) pak musí *HtmlUnit* postupně *kliknout* na každý nalezený výsledek, čímž se zobrazí podrobnosti o nalezené publikaci. Na této stránce se pak nalézá odkaz na seznam citací. V tomto seznamu pak lze rovnou vyčíst autory publikací, které byly citovány ve zvolené publikaci hledané osoby. Tato jména pak získá statický parser.

Druhým citačním serverem, který jsem použil jako zdroj dat, je *Scopus*. Podobně jako *Web of Knowledge* používá ke svým funkcím javascript, avšak nástroj *HtmlUnit* si s ním nedokáže poradit, tedy tento nástroj nelze použít. Vzhledem k tomu, že se mi již nepodařilo najít žádný jiný emulátor webového prohlížeče, který je napsaný v Javě, musel jsem použít nástroj *Selenium*, který slouží k ovládání webových prohlížečů jako je například *Firefox* či *Chrome*. Při samotném prohledávání tohoto zdroje tak bude vidět fyzicky okno zvoleného prohlížeče. Já jsem pro tyto účely zvolil *firefox* od Mozilla Foundation. Z toho tak plyne omezení, že při použití tohoto modulu je nutné mít nainstalovaný *firefox*.

Samotné získání požadovaných dat na serveru *Scopus* je relativně snadné. Po vyplnění jména hledané osoby se zobrazí seznam nalezených publikací. Dále stačí označit všechny

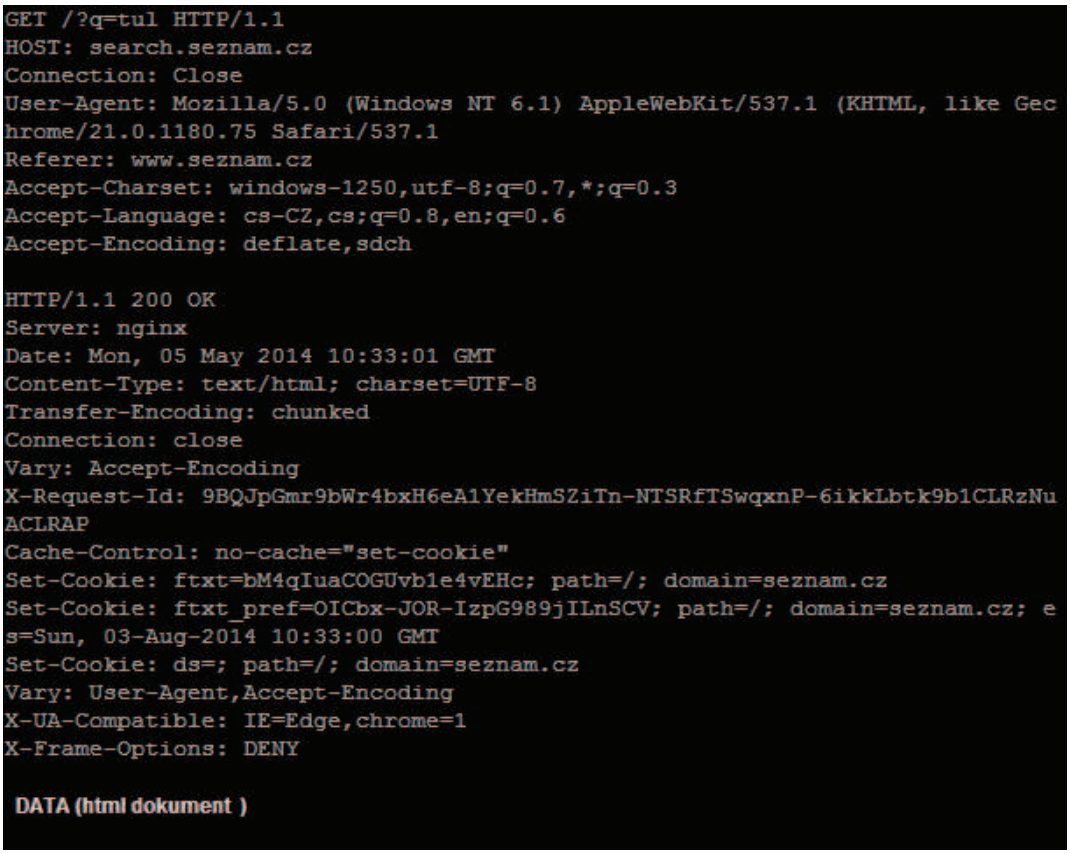
publikace na stránce (20 publikací na stránku) a kliknout na odkaz *View References*. Na několika stránkách se tak zobrazí citace k několika publikacím najednou. Jména autorů těchto citací opět získá příslušný statický parser. Tento postup se provede pro všechny stránky nalezených publikací. *Scopus* však zobrazí maximálně 2000 záznamů seřazených od nejnovějších. Pokud je k někomu nalezeno více než 2000 záznamů, pak tento modul nedokáže získat data z nejstarších záznamů.

Vzhledem k tomu, že oba moduly jsou založeny na statickém přístupu ke zdroji, může dojít k tomu, že i při nepatrné změně struktury daných zdrojů dojde k tomu, že modul přestane fungovat. V takovém případě je nutné upravit postup modulu tak, aby vyhovoval upravené struktuře zdroje. Během tvorby této práce k tomu došlo několikrát.

4.1.2 Internetové vyhledávače

Ve své práci jsem se zaměřil i na internetové vyhledávače, konkrétně *Google* a *Seznam*, které patří v ČR k nejpoužívanějším. Požadavkem na tyto zdroje je získání odkazů, které jsou nalezeny daným vyhledávačem na zadaný výraz. Tyto odkazy pak mohou být v budoucnu použity k dalšímu zpracování, v rámci této práce se jimi však více nezabývám.

Seznam je český internetový vyhledávač s velice nízkou detekcí robotů při vyhledávání. Stačí mu podstrčit URL s hledaným výrazem a tvářit se při tom jako známý webový prohlížeč. Pro hledaný výraz *tul* pak je kompletní URL *search.seznam.cz/?q=tul*. Samotné vyhledávání nepoužívá zabezpečený protokol (https), takže stačí otevřít obyčejný raw socket na adresu *search.seznam.cz* a port 80, pomocí požadavku GET zažádat o */?q=tul* a přidat pár dalších informací, jako je *User-Agent*, *Accept-Charset* a podobně. Některé nedůležité informace je nutné vyplnit, jinak dojde k detekci robotického vyhledávání. Celý dotaz pro vyhledání textového řetězce ‚tul' včetně odpovědi je na Obrázek 7. V datové části odpovědi je pak html kód dokumentu s nalezenými výsledky. Nad tím se pustí příslušný parser, který získá odkazy na nalezené výsledky, které se pak mohou dále zpracovávat.



Obrázek 7: žádost o html dokument s vyhledaným výrazem 'tul' na server search.seznam.cz, port 80

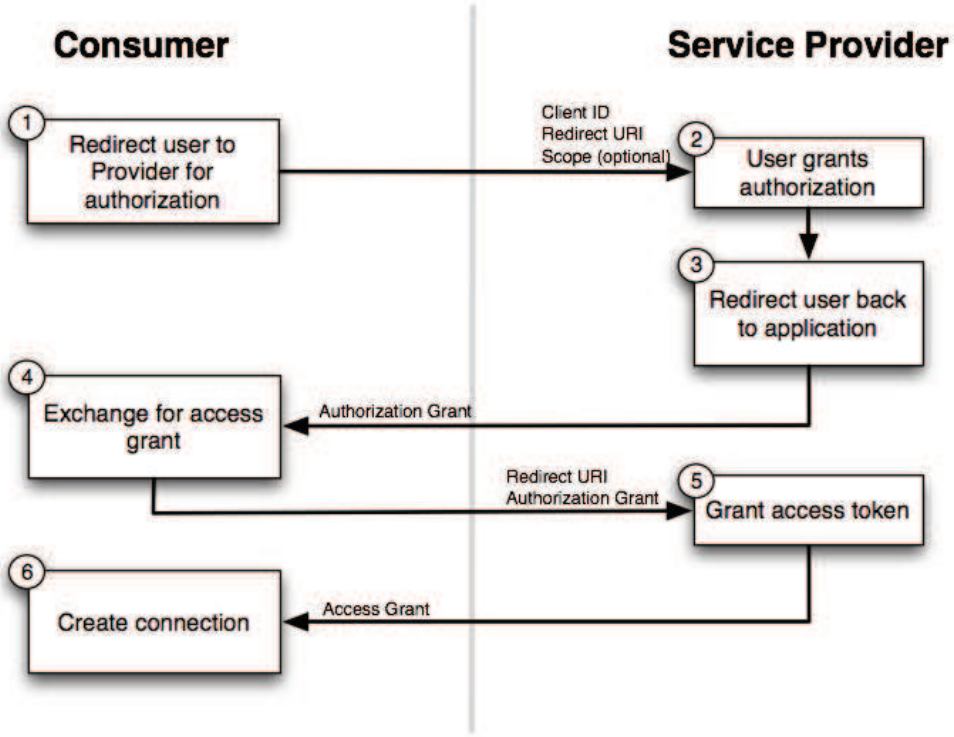
Google má oproti *Seznamu* mnohem silnější detekci robotického vyhledávání. Také má ale vlastní vyhledávací API (jednoduše nazvané *custom search*), které umožňuje získat výsledky vyhledávání pomocí několika knihoven. Zdarma je však pouze 100 vyhledání za den. Je také nutné si nejprve nechat vygenerovat klíč pro aplikaci, kde má být vyhledávání použito. Tento klíč je pak pro jednu aplikaci stále stejný (jeden klíč lze použít ve více aplikacích). Na získání výsledků (nalezené odkazy) je pak třeba ještě velice jednoduchý parser, který výsledky převede například do textového pole.

4.1.3 Sociální sítě

V této práci jsem se zaměřil pouze na jednu sociální síť, a to *facebook*. Pro tuto sociální síť existuje více API přímo v Javě, konkrétně *RestFB* a *Spring Social*. Modul pro tento zdroj pak má za úkol jen získat obecný přístup k datům, neboli vyřešit přihlášení uživatele. Pro tyto účely je však nejprve nutné vytvořit uživatelský účet, přes který se bude na sociální síť aplikace přihlašovat. Pro samotné přihlášení používá facebook protokol OAuth2, což je protokol pro přihlašování pro aplikace třetích stran. Jeho jednotlivé kroky znázorňuje Obrázek 8.

V prvním kroku přihlášení je nutné vygenerovat a odeslat požadavek na přihlášení. Odpověď na tento požadavek je URL na doméně facebooku. Tuto URL by měl uživatel, který se přes aplikaci chce přihlásit, navštívit a vyplnit své přihlašovací údaje. V tomto kroku je problém v tom, že modul pro facebook má být plně automatický, tedy že během jeho běhu nesmí docházet k další komunikaci s uživatelem aplikace. Tento problém jsem po delší době vyřešil použitím nástroje HtmlUnit, což je sada knihoven pro emulaci webového prohlížeče. Lze tak strojově načíst danou URL adresu, vyplnit správné přihlašovací údaje a potvrdit odeslání přihlašovacího formuláře, přičemž se vše provede pomocí kódu modulu bez otevření jakéhokoliv fyzického okna prohlížeče.

Po zadání přihlašovacích údajů dojde k přesměrování na URL předem definovanou v požadavku o přihlášení. V datové části této URL se pak nachází kód, který je potřeba k dokončení přihlašovacího procesu. V programu se vygeneruje požadavek na získání přístupového tokenu ze získaného kódu, odpovědí je pak samotný přístupový token, pokud všechny předchozí body autentizace byly správně provedeny. Tento token je pak použit pro veškerou činnost API na sociální síti. V rámci této práce jsem předpřivavil moduly se *Spring social* i *RestFB* API.



Obrázek 8: Vizualizace kroků protokolu OAuth2 (Zdroj: [13])

4.2 NoSQL databáze

Cílem této práce je mimo jiné i ukládání velkého množství dat a jejich čtení v co nejkratším čase. Největší požadavek je tedy na rychlost čtení a zápisu. K tomuto účelu jsou vedeny NoSQL databáze. Je však mnoho druhů NoSQL databází, kde každý druh je vhodný k jinému účelu. V této podkapitole bude popsáno základní dělení NoSQL databází a výběr databáze, která bude v této práci použita.

Základními charakteristikami, které jsou společné pro většinu NoSQL databází, jsou následující:

- Nepoužívají relační model
- Jsou open-source (většinou)
- Jsou navrhnuté pro práci v clusterech
- Jsou horizontálně škálovatelné
- Nemají pevnou strukturu (nemají schéma)

NoSQL databáze pak lze rozdělit například podle datového modelu, a to na model klíč-hodnota (key-value), model rodiny sloupců (column-family), dokumentový model, grafový model a další.

4.2.1 Model klíč-hodnota

Databáze modelu klíč-hodnota jsou nejvíce podobné relačním databázím. Každá tabulka obsahuje primární klíč (jeden nebo více sloupců) a dále další sloupce s hodnotami závislými na primárním klíči. Každý řádek ale může mít jiné schéma. K jednotlivým hodnotám v databázi se pak dá přistupovat pouze pomocí klíče.

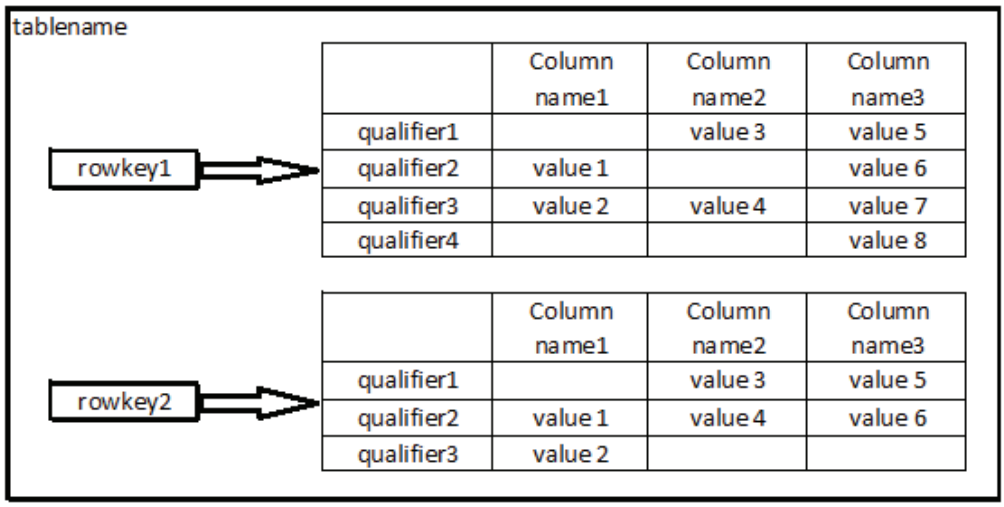
NoSQL databáze tohoto typu jsou například DynamoDB, LevelDB či Dynomite.

4.2.2 Model rodiny sloupců

Tento typ NoSQL databází má data uložená jako vícerozměrné mapy. Každý řádek má klíč, podle kterého se jednotlivé řádky identifikují. Ke každému klíči pak existuje mapa obsahující jednotlivé hodnoty. Každá hodnota v mapě je pak určená názvem sloupce (column name) a kvalifikátoru (qualifier). Názvy sloupců jsou definovány při vytváření tabulky (později lze přidávat a odebírat další sloupce). Při vkládání záznamu tak je kromě samotné hodnoty zadána hodnota klíče, název existujícího sloupce a libovolný kvalifikátor. Každá

hodnota pak obsahuje ještě automaticky vkládané časové razítko značící, kdy byl záznam vložen. Obrázek 9 znázorňuje strukturu dat v tomto typu NoSQL databází.

Do této kategorie NoSQL databází patří například BigTable, Cassandra a Hbase, která byla vybrána a použita pro tuto práci.



	Column name1	Column name2	Column name3
qualifier1		value 3	value 5
qualifier2	value 1		value 6
qualifier3	value 2	value 4	value 7
qualifier4			value 8

	Column name1	Column name2	Column name3
qualifier1		value 3	value 5
qualifier2	value 1	value 4	value 6
qualifier3	value 2		

Obrázek 9: Struktura dat v modelu rodin sloupců

4.2.3 Dokumentový model

Cílem dokumentových NoSQL databází je zakódovat data do nějakého standardního formátu, ať už textového (XML, JSON atd.), či binárního (BSON, PDF atd.). Databáze je pak většinou velice podobná modelu klíč-hodnota, kde hodnota je zde příslušný dokument. Každý dokument pak může mít různé schéma a může obsahovat i reference na další dokument.

Do této skupiny NoSQL databází patří například MongoDB a CouchDB.

4.2.4 Grafový model

Dalším typem NoSQL databází jsou grafové databáze. Ty jsou vhodné zejména pro ukládání vztahů (relací) mezi jednotlivými entitami. Výborně se tak hodí pro sociální sítě. Základními prvky jsou uzly a hrany. Jejich princip se podobá spojování tabulek v relačních databázích.

Do této skupiny patří například GraphBase, InfoGrid a Neo4J.

4.2.5 HBase

Pro ukládání dat v rámci této práce jsme s vedoucím práce zvolili NoSQL databázi HBase od Apache Software Foundation. Je to open-source NoSQL databáze založená na modelu rodiny sloupců (column families) a vychází z databáze BigTable (od Google). Je

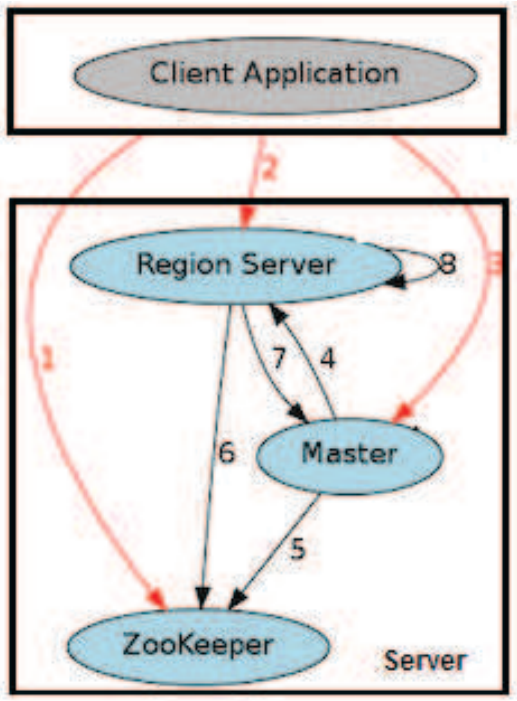
napsána v programovacím jazyce Java a využívá HDFS (*Hadoop Distributed FileSystem*), který umožňuje uložení velkých objemů dat na několika počítačích, optimálně na nějakém clusteru.

Poslední verze Hbase, která byla označena jako stabilní, je verze 0.94, která je určena pro UNIXové operační systémy. Tato verze byla umístěna na školní server *kaja.ntil.tul.cz*, na němž bylo pro Hbase vyhrazeno místo po domluvě se správcem tohoto serveru (dr. Jiří Hnídek).

Hbase se konfiguruje pomocí XML souboru. Všechny parametry, které lze v tomto souboru nakonfigurovat, mají danou defaultní hodnotu, na jejíž hodnotu jsou nastaveny, pokud nejsou přenastaveny v konfiguračním souboru. Konfigurační soubor tak nemusí obsahovat všechny parametry, ale pouze ty, které je nutno nastavit. Stejný soubor je vhodné použít i pro načtení konfigurace v klientské aplikaci, která se má k dané databázi připojit.

Hbase se spouští v distribuovaném (resp. pseudodistribuovaném) režimu, kdy jsou spuštěny celkem 3 procesy. První proces je *Hbase master*, který řídí celou databázi. Dalším procesem je *ZooKeeper*, který je zde pro komunikaci s klienty. Posledním procesem je *Region Server*, který se stará o ukládání dat. Pokud je databáze spuštěna na clusteru, *Region Server* pak musí být spuštěn na každém uzlu clusteru a seznam IP adres jednotlivých uzlů pak musí být zaznamenán v souboru *regionservers* ve stejné složce, ve které se nachází konfigurační soubor. Jednotlivé procesy pak spolu komunikují po síti pomocí protokolu TCP, je tedy nutné na serveru otevřít příslušné porty. Obrázek 10 pak znázorňuje komunikaci mezi klientem a HBase databází, včetně interní komunikace mezi částmi databáze.

Pro připojení k databázi a práci s databází je k dispozici API, které jsem použil v rámci této práce.



Obrázek 10: Komunkace mezi klientem a Hbase databází (Zdroj: [16])

5 Implementace

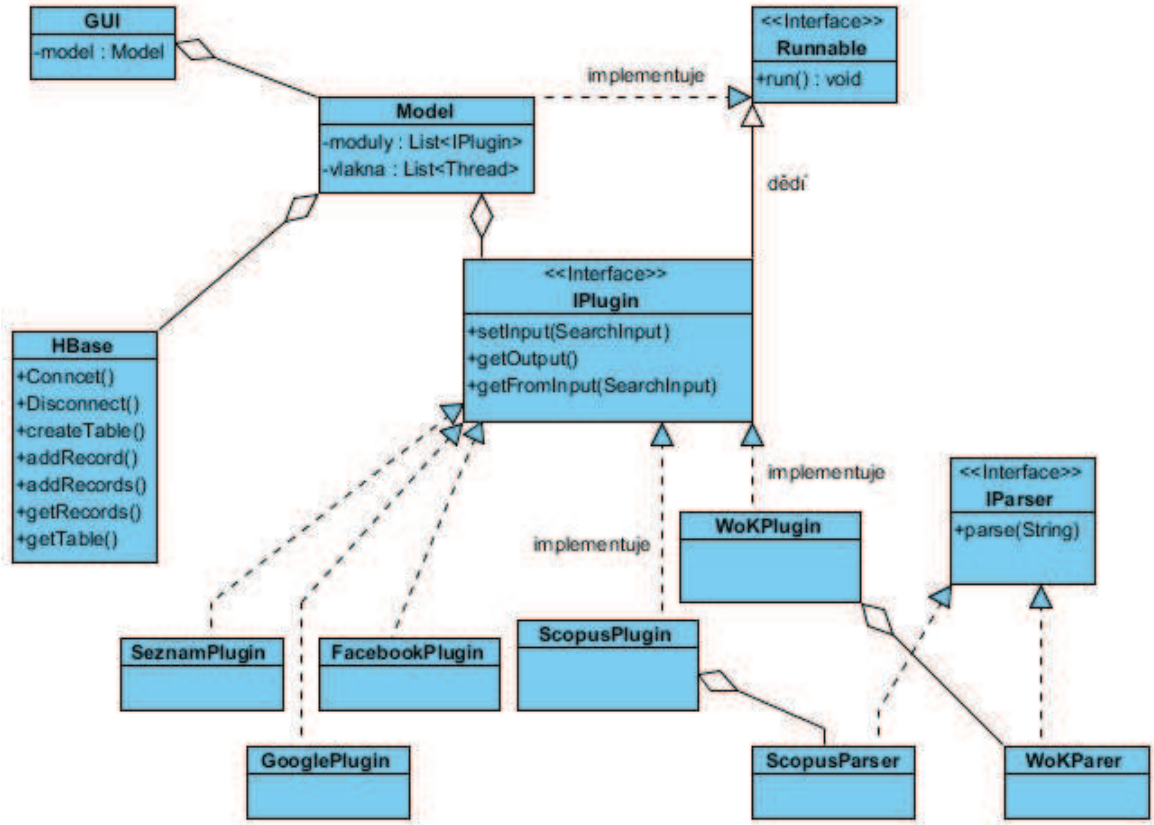
Výsledná aplikace byla rozdělena na dvě části. První je určená ke stahování samotných dat do databáze pomocí připravených modulů, která bude spuštěna delší čas v kuse. Důvodem je dlouhá doba získání většího množství dat. Druhá část pak bude sloužit k načtení dat z databáze a jejich zpracování (např. ořez hran a export grafu do souboru určitého formátu).

Před začátkem implementace těchto dvou aplikací je však nutné vytvořit si tabulku v databázi, do které se data budou ukládat. Vytvořil jsem tak tabulku se dvěma *rodinami slouců* (*column family)*, jeden pro každý modul pro stahování dat. První je tedy pojmenován *scopus* a druhý *wok* (na pořadí nezáleží). V případě přidání dalšího modulu v budoucnu pak bude třeba přidat další *rodinu sloupců*. Daty na každém řádku je pak seznam jmen, které citovala osoba, jejíž jméno je klíčem řádku. Tato data jsou rozdělena podle příslušných zdrojů, odkud byla data sebrána. Při zadání jednoho citovaného jména je pak nutné zadat název tabulky (ten je stejný pro celou aplikaci), klíč řádku (jméno osoby, která vkládanou osobu citovala), rodinu sloupce (zdroj, ze kterého jméno pochází) a kvalifikátor, který musí být unikátní pro každý záznam v rodině sloupců (jinak dojde k přepsání jiného záznamu). Jako kvalifikátor jsem tedy zvolil celočíselnou hodnotu, která se inkrementuje pro každý vložený záznam. Toto číslo spravuje vytvořená knihovna pro práci s HBase databází, když je volána funkce pro vložení množiny dat do databáze.

5.1 Aplikace pro stahování dat

Pro aplikaci, která má za úkol stahovat data, jsem zvolil architekturu MVC (Model-View-Controller). View, neboli pohled, je vlastní GUI (grafické uživatelské rozhraní) dané aplikace. V tomto případě se jedná o velice jednoduché GUI, protože aplikace je spíše určena pro samostatnou práci bez zásahu uživatele. Controller, neboli řadič, je pak v podstatě součástí vlastního GUI. Jedná se o callback funkce volanými pohledem např. při stisknutí tlačítka, nebo volanými modelem při změně jeho stavu. Model je pak hlavní částí aplikace, protože provádí většinu práce této aplikace, totiž získává data.

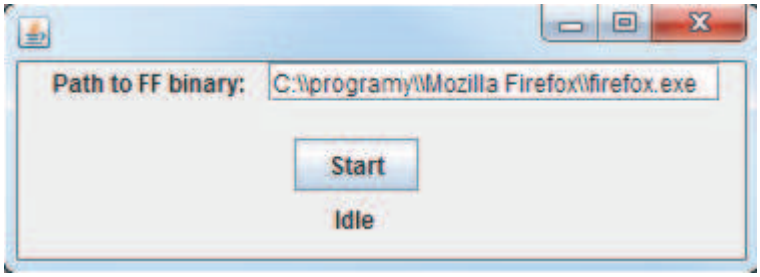
Obrázek 11 znázorňuje zjednodušený diagram tříd v UML pro tuto aplikaci. Model obsahuje několik pluginů typu IPlugin, jednotlivé instance jsou však konkrétní pluginy. Model také obstarává uložení dat do databáze pomocí třídy HBase. Model a jednotlivé pluginy implementují rozhranní Runnable, lze je tak spustit ve vlastních vlíknech



Obrázek 11: Zjednodušený diagram tříd v UML pro aplikaci pro stahování dat

5.1.1 Grafické uživatelské rozhraní

Jak bylo zmíněno výše, GUI této aplikace je velice jednoduché. Obsahuje pouze tlačítko pro spuštění činnosti aplikace, výpis statusu aplikace (zda stahování běží či nikoliv) a dále ještě obsahuje vstupy potřebné pro jednotlivé moduly, čímž je v současnosti jen cesta k webovému prohlížeči *Mozilla firefox*. Obrázek 12 znázorňuje výsledné GUI této aplikace.



Obrázek 12: GUI aplikace pro stahování dat

Po kliknutí na tlačítko *Start* načte aplikace textový soubor *names-to-download.txt*, který se nalézá ve stejném adresáři jako tato aplikace. Obsahem tohoto textového souboru jsou celá jména osob, která mají sloužit jako vstup pro jednotlivé moduly. Jednotlivá jména jsou v souboru oddělena zalomením řádku.

Pod tlačítkem *Start* se nalézá značení stavu stahování dat. V současnosti se jedná pouze o nápisy *Idle*, *Running* a *Idle after Exception*. *Idle* značí, že aplikace nic nedělá. To nastane po startu aplikace a po dokončení získávání dat. Nápis *Running* značí, že běží proces získávání dat. Nápis *Idle after Exception* pak značí, že program přestal stahovat data kvůli neočekávané výjimce. Ta je pak většinou zobrazena jako okno s chybovou hláškou, zároveň je vypsána i do konzole aplikace (pokud je aplikace spuštěna přes příkazový řádek).

5.1.2 Model

Model této aplikace je spuštěn ve vlastním vlákně, protože jeho činnost je časově náročná a do jeho ukončení by došlo k zamrznutí GUI, kde je model naplněn vstupními daty a spouštěn. Jeho vstupem je fronta jmen, která je nutno přivést na vstupy jednotlivých modulů a následně výstupy těchto modulů uložit do databáze. Každé jméno na vstupu i na výstupu je upraveno do formátu společného pro všechny zdroje dat, jak bylo napsáno v kapitole 4.1.1.

Model umožňuje vyhledávání do různé hloubky. Pokud je nastaven na hloubku 1, pak pomocí modulů získá pouze jména osob, která byla citována osobou na vstupu. Pro hloubku 2 pak nalezne jména osob, které citovaly osoby, které citovala osoba na vstupu. Velikost hloubky není příliš omezená, měla by se však používat s rozvahou, jelikož počet dat pak roste exponenciálně. Aplikace je v současnosti nastavena na hledání do hloubky velikosti 2.

Funkce modelu pro jednotlivé vstupy by se dala popsat několika jednoduchými kroky. Nejdříve se zjistí, zda daný vstup již není uložen v databázi. Pokud ne, pošle se na vstup jednotlivých modulů, které model následně spustí (každý modul ve vlastním vlákně) a dále čeká na ukončení činnosti všech spuštěných modulů. Poté posbírá výsledky a uloží je do databáze. Nakonec zkontroluje hloubku zadanou u vstupu. Pokud je větší než 1, tak nalezené výsledky zařadí do své fronty vstupů. Pokud je na začátku zjištěno, že daný vstup již byl zpracován v minulosti (ze záznamu v databázi), pak se pouze zkontroluje zadaná hloubka pro tento vstup, na základě hloubky může dojít k přidání citovaných jmen (která jsou v databázi) do vstupní fronty modelu, a následně začne zpracování dalšího vstupu z fronty. Při přerušení činnosti modelu pak při jeho znovuspuštění dojde k pokračování v místě, kde při přerušení přestal (nemusejí se tak stahovat data, která již byla stažena a uložena). Také se tím zjednoduší problém, kdy se skupina osob cituje v cyklu (kružnici). Nevýhodou však je obtížná aktualizace již získaných záznamů v budoucnu.

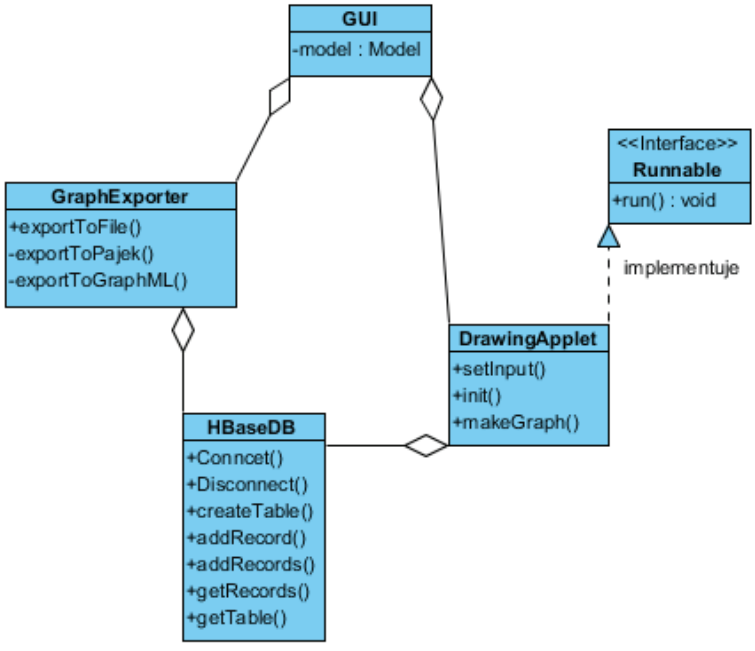
5.2 Aplikace pro zpracování dat

Jakmile jsou alespoň částečná data k dispozici v databázi a jejich získávání stále probíhá, je vhodné se zaměřit na zpracování těchto dat. Tato část se soustředí na zpracování konkrétních dat, aplikace pro tyto účely tedy nebude tak snadno rozšiřitelná o zpracování dalších dat.

V rámci této práce se do databáze ukládají jména osob, které se podle citačních serverů citovaly. Lze tak sestavit orientovaný graf s ohodnocenými hranami, kde jednotlivé vrcholy jsou reprezentovány jmény osob a tyto vrcholy jsou propojené orientovanými hranami právě tehdy, když osoba tvořící zdrojový vrchol citovala osobu tvořící cílový vrchol. Ohodnocení hrany pak udává číslo, kolikrát byla cílová osoba citována zdrojovou osobou.

Na trhu existuje mnoho aplikací či knihoven pro zpracování a vizualizaci sítí. Patří mezi ně například volně dostupný program Pajek, který je vhodný pro rychlé zpracování a vizualizaci velkých sítí. Proto byla tato aplikace vytvořena s cílem vizualizovat graf načtený z databáze s možností jednoduchého zjednodušení ořezem hran podle jejich ohodnocení a pro složitější práci se sítí se provede export do souboru, který se dále importuje do pokročilého programu pro práci s touto sítí.

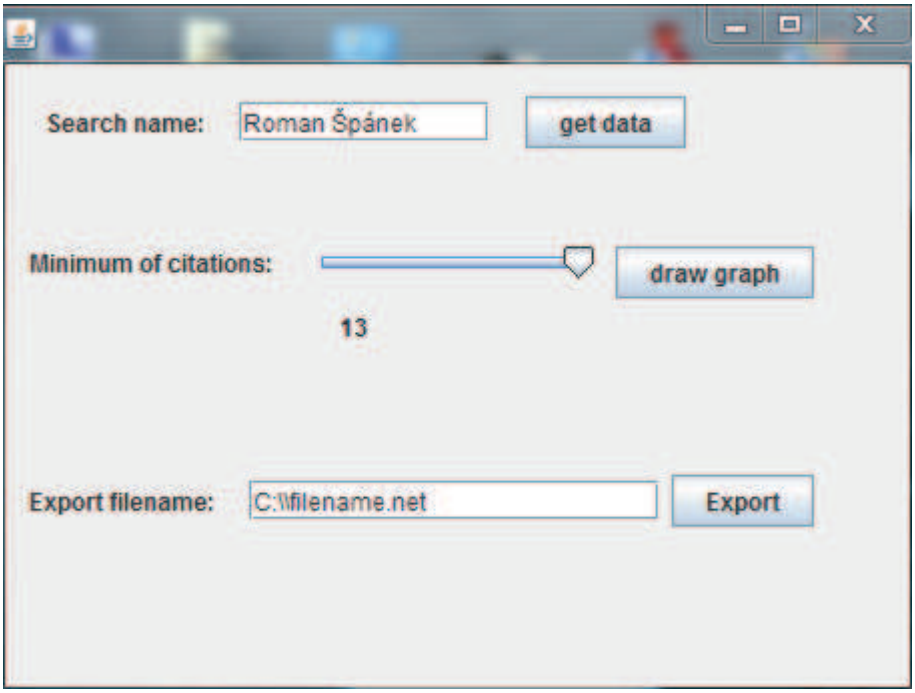
Zjednodušený diagram tříd pro tuto aplikaci pak zobrazuje Obrázek 13. GUI pak ovládá vykreslování grafu pomocí třídy DrawingApplet, která si veškerá data pro vykreslení stáhne z databáze. Stejně tak činí třída pro export do souboru.



Obrázek 13: diagram tříd v UML aplikace pro zpracování dat

5.2.1 GUI aplikace

Vlastní grafické uživatelské rozhranní aplikace je opět jednoduché, jak můžete vidět na Obrázek 14. Nachází se zde vstupní pole pro zadání jména, které pro které se mají v databázi najít data, a vlastní tlačítko pro spuštění získávání dat z databáze. Zbylé komponenty se objeví až po získání a předzpracování dat. Je pak tedy možno vykreslit graf s nastaveným ořezem hran či exportovat graf do souboru.

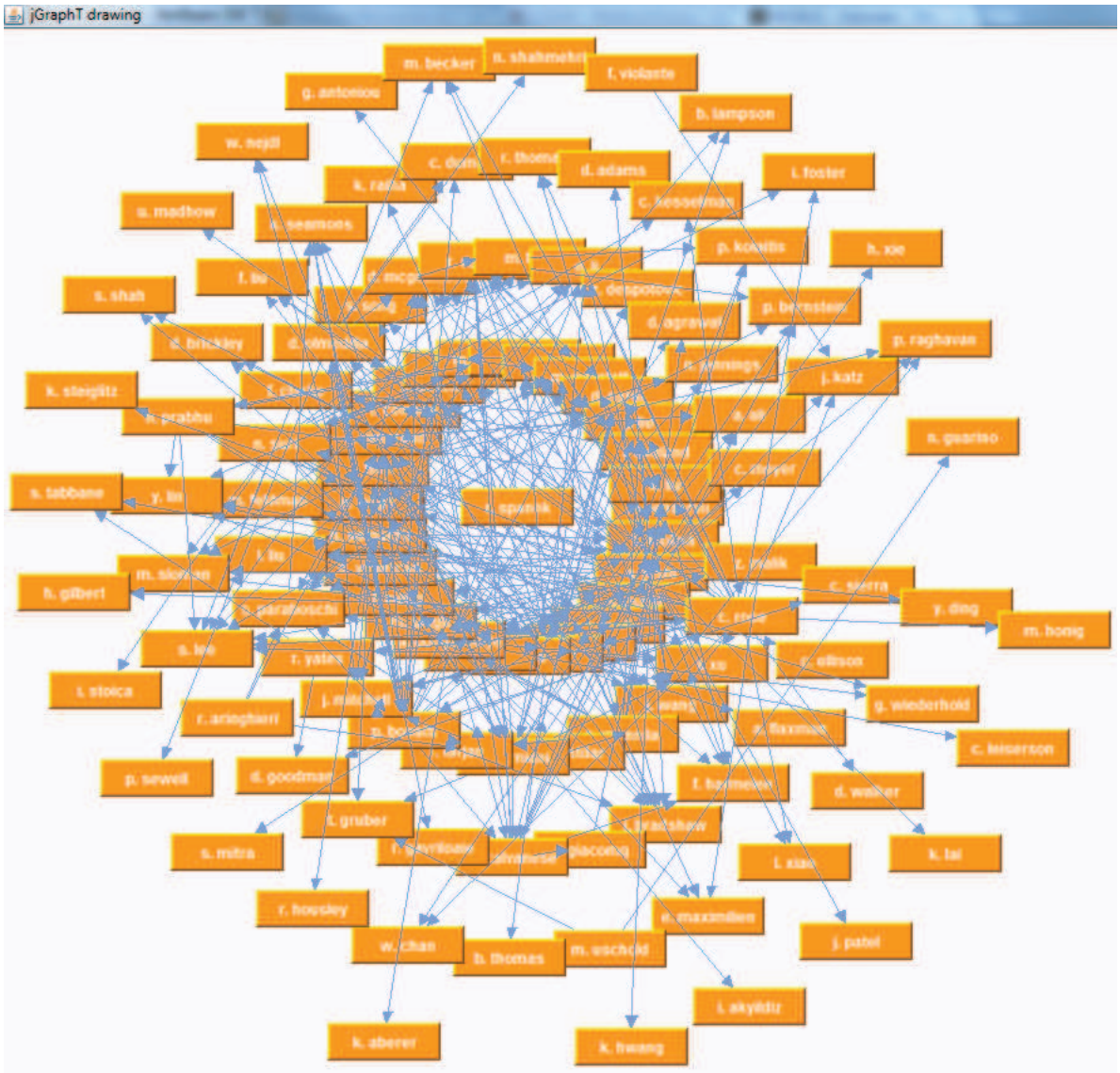


Obrázek 14: GUI aplikace pro zpracování dat

5.2.2 Vykreslení grafu

Po předzpracování dat, v kterém se data uloží do datových struktur, ze kterých se následně může vytvořit a vykreslit síť, je možno zvolit minimální ohodnocení hrany v grafu. Všechny hrany, které mají nižší ohodnocení než zde zadané, jsou pak smazány. Stejně tak jsou následně smazány vrcholy, které nemají žádnou hranu (vstupní ani výstupní). Maximální hodnota pro tento ořez je rovna nejvyššímu ohodnocení hrany vycházející z vrcholu, který je zadán ve vstupním poli. Nedojde tak ke smazání tohoto vrcholu díky tomu, že by neměl žádnou hranu (vždy má alespoň jednu výstupní hranu). Tento ořez také může mít za následek vytvoření nesouvislého grafu.

Obrázek 15 znázorňuje vykreslení grafu pro vstupní osobu Roman Špánek s minimálním ohodnocením hran rovno 13. Pro vykreslení bylo použito knihovny jGraphT, jednotlivé vrcholy byly vykresleny v kruhu kolem hlavního (vstupního) vrcholu.



Obrázek 15: Vykreslený graf po ořezu hran

5.2.3 Export grafu do souboru

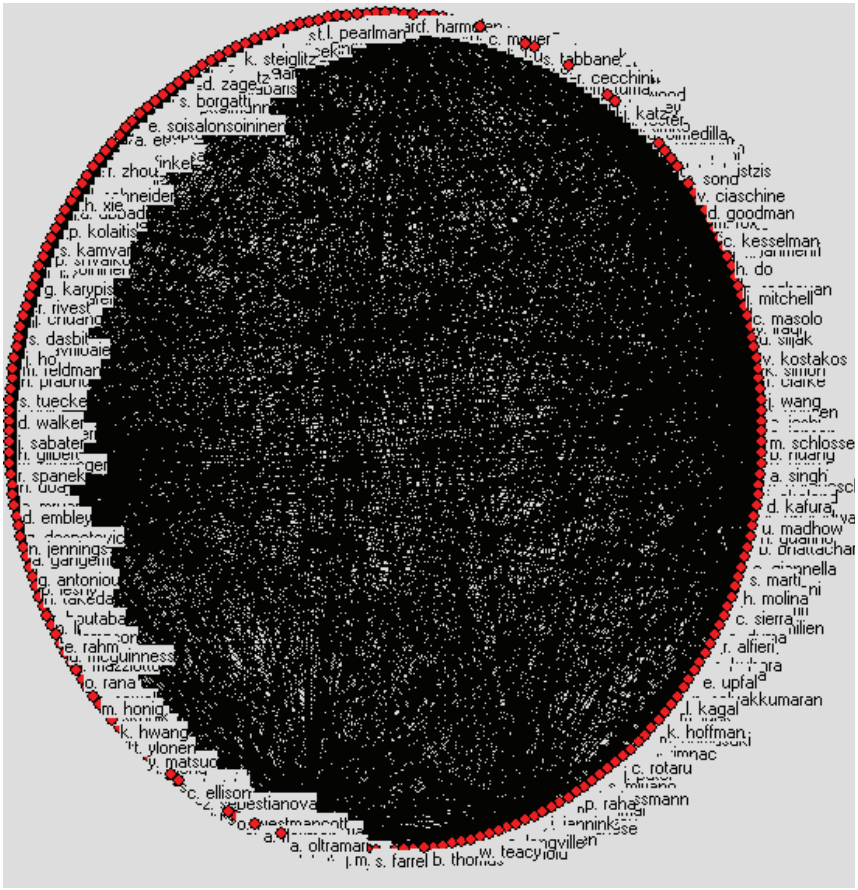
Pro pokročilejší zpracování grafu je vhodné použít již hotové nástroje. K tomu je však nutné přenést data o grafu z aplikace realizované v této práci do externího programu. To je realizováno pomocí exportu dat do souboru, který je pak načten jiným programem. Pro tyto účely jsem vytvořil export grafu do souboru pro program Pajek a do souboru ve formátu GraphML.

GraphML je formát souboru pro grafy. Je založen na XML a je tedy i celkem přehledný. Navíc je volně použitelný jak pro vědecké účely, tak v komerční sféře. Samotný export pak jen načte data z databáze, vytvoří soubor s příponou *.graphml*, uloží do něj požadované hlavičky, a pak jen přidává jednotlivé vrcholy a poté jednotlivé hrany Nakonec je přidáno uzavření dosud otevřených elementů, ve kterých se graf nalézá a soubor je hotov.

Export do programu Pajek je o něco složitější. Vrcholy nejsou identifikovány pomocí svého názvu (jména osob), ale pomocí číselného identifikátoru. Hrany jsou pak tvořeny třemi čísly, tedy zdrojovým identifikátorem vrcholu, cílovým identifikátorem a číslem pro ohodnocení hrany. Samotné ohodnocení hrany sice není povinné, v rámci této práce je to však nutné. Kvůli těmto identifikátorům je tak nutné (na rozdíl od formátu GraphML) nejdříve vytvořit graf v určitých datových strukturách a až poté tyto struktury projít prvek po prvku a ukládat je do souboru.

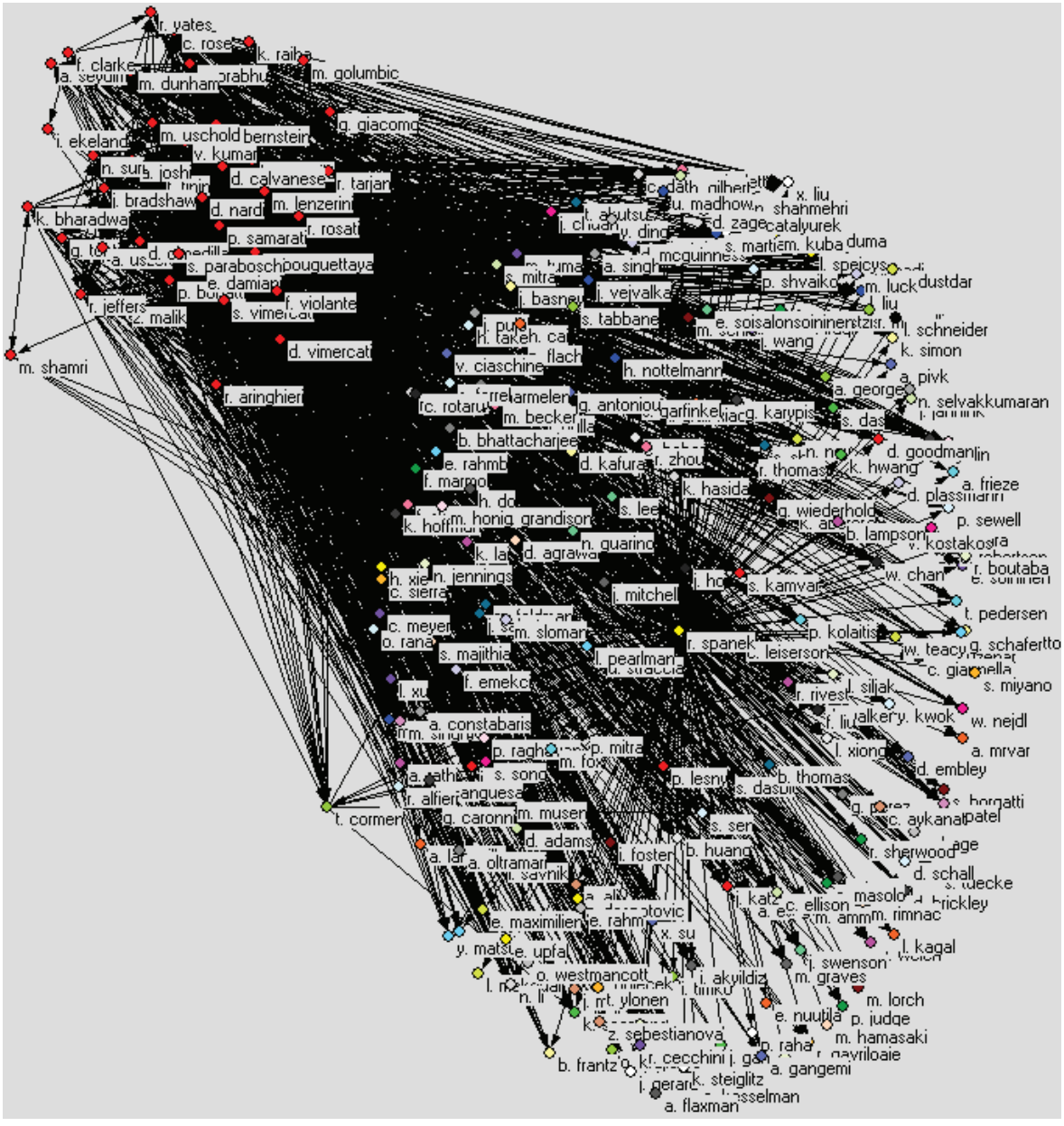
5.2.4 Program Pajek

Pajek je program pro analýzu a vizualizaci velkých sítí. Za tímto účelem byl použit v rámci této práce. Program načte soubor získaný exportováním dat, jak bylo popsáno v předchozí podkapitole. Tento graf se může rovnou vykreslit, což vyjadřuje Obrázek 16. Graf byl sestaven ze stažených dat pro osobu Roman Špánek (pro vedoucího této práce). Červené body jsou jednotlivé vrcholy, hrany jsou pak zobrazeny černou barvou. Takto vykreslená síť bez dalších úprav je dosti nepřehledná.



Obrázek 16: obecné vykreslení sítě v programu Pajek

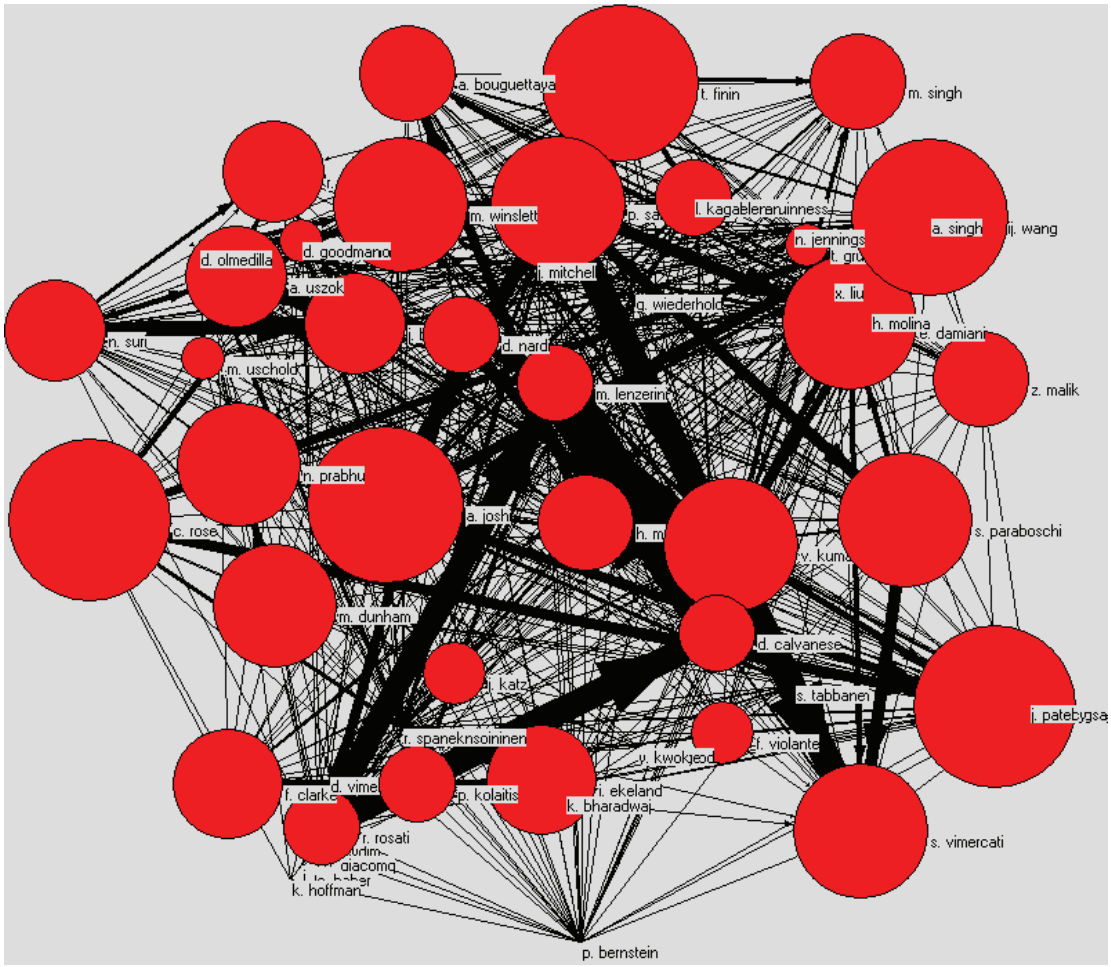
Ve vzniklém grafu můžeme pak například hledat skupiny autorů, kteří se mezi sebou navzájem často citují. Toho se docílí vyhledáním silně souvislých komponent. Silně souvislá komponenta v souvislém orientovaném grafu je podgraf, u kterého platí, že pro libovolné dva vrcholy u a v existuje cesta z u do v a také z v do *u*. Obrázek 17 znázorňuje nalezení silně souvislých komponent v dříve zmíněném grafu. Program v tomto grafu nelezl pouze jednu silně souvislou komponentu, která má více než jeden vrchol. Vrcholy v této komponentě mají na obrázku červenou barvu.



Obrázek 17: nalezení silně souvislých komponent v programu Pajek

Další možností je využití toho, že jednotlivé hrany jsou ohodnoceny. To je vhodné například pro vytvoření ostrovů. Velikost ostrovu zde závisí na množství vstupních

a výstupních hran a na ohodnocení těchto hran. Obrázek 18 znázorňuje vytvoření ostrovů v daném grafu, kde velikost vrcholu značí velikost ostrovu a šířka hran značí jejich ohodnocení.



Obrázek 18: vytvoření ostrovů v grafu v programu Pajek

Program Pajek nabízí mnoho dalších algoritmů pro práci s grafy, vždy záleží hlavně na tom, jaké informace chce uživatel z grafu vyčíst.

6 Závěr

V rámci této práce byla vytvořena aplikace pro získání dat z online zdrojů. V současné verzi jsou připraveny moduly pro stahování dat z citačních serverů, dále byly předpřipraveny moduly na získávání dat z webových vyhledávačů *Google* a *Seznam* a sociální sítě *facebook*. Aplikace byla navržena tak, aby bylo v budoucnu možno snadno přidávat další moduly. Pro ukládání stahovaných dat byla po důkladném prozkoumání možností zvolena NoSQL databáze HBase, která byla spuštěna na školním serveru *kaja.nti.tul.cz*.

Dále došlo v této práci k nastudování technik pro zpracování získaných dat a vytvoření aplikace, která dokáže vizualizovat orientovaný graf vytvořený ze stažených dat po jednoduchém ořezu hran podle jejich ohodnocení. Pro pokročilejší zpracování dat byl v této aplikaci vytvořen export grafu do souboru formátu *graphml* a *NET*. Druhý z nich je používán programem Pajek, v kterém došlo ke zpracování a vizualizaci pokročilejších technik pro zpracování dat (vytvoření ostrovů, nalezení silně souvislých komponent) za účelem získání některých informací z dostupných dat.

Všechny body zadání této práce byly splněny.

7 Seznam použité literatury

[1] Batagelj, V.: Networks [online], NICTA, Sidney 2005, [cit. 2014-05-07]. URL: (vlado.fmf.uni-lj.si/pub/networks/Doc/Seminar/nicta01.pdf)

[2] Batagelj, V., Mrvar, A.: Analysis of Large Networks with Pajek [online], Senbelt XXVII, Vancouver, British Columbia 2006, [cit. 2014-05-02]. URL: (vlado.fmf.uni-lj.si/pub/networks/Doc/Sunbelt/sunbeltXXVI.pdf)

[3] Citační databáze [online], ČVUT, Praha, [cit. 2014-04-24]. URL: (knihovna.cvut.cz/veda/citacni-databaze/)

[4] Černý, J.: Silně souvislé komponenty [online]. URL: (algoritmy.eu/zga/pruchod-grafu/silne-souvisle-komponenty/)

[5] Data mining [online], [cit. 2014-04-27]. URL: (www.unc.edu/~xluan/258/datamining.html#history)

[6] Dúbravčík, M.:NoSQL uložiště pro data v podobě časových řad [online], Fakulta informatiky, Masarykova univerzita, Brno 2013, [cit. 2014-05-03]. URL: (is.muni.cz/th/325004/fi_m/diplomova-praca.pdf)

[7] Firestone, J.M.: Data mining and KDD: A Shifting Mosaic [online], 1997, [cit. 2014-04-27], URL: (dc314.4shared.com/doc/x4l-xxpX/preview.html)

[8] Franklin, C.: How Internet Search Engines Work [online], [cit. 2014-05-10]. URL: (computer.howstuffworks.com/internet/basics/search-engine1.htm)

[9] Hliněný, P.: Toky v sítích [online], Fakulta informatiky, Masarykova univerzita, Brno, [cit. 2014-05-07]. URL: (//www.fi.muni.cz/~hlineny/Teaching/OU/OU-lect--2.pdf)

[10] Procházka, M.: Data mining: jiný pohled na problém [online], [cit. 2014-04-27]. URL: (vtm.e15.cz/aktuality/data-mining-jiny-pohled-na-problem)

[11] Řezanková, H., Húsek, D., Snášel, V.: Shluková analýza dat, Professional Publishing 2007, ISBN 978-80-86946-26-9

[12] SAS Enterprise Miner: SEMMA [online], [cit. 2014-04-28]. URL: (www.sas.com/offices/europe/uk/technologies/analytics/datamining/miner/semma.html)

[13] Spring Framework: OAuth2 service providers [online], [cit. 2014-04-25]. URL: (www.springframework.net/social/refdoc/serviceproviders.html)

[14] Skřivan, J.: Databáze nejsou jen MySQL [online], [cit. 2014-05-03]. URL: (interval.cz/clanky/databaze-nejsou-jen-mysql/)

[15] The Search Engine list [online], [cit. 2014-05-10], URL: (www.thesearchenginelist.com/)

[16] Xiang, J.: Guide to Using Apache HBase Ports [online], 2013, [cit. 2014-05-03]. URL: (blog.cloudera.com/blog/2013/07/guide-to-using-apache-hbase-ports/)

[17] Wilson, J.: Understanding Hbase and BigTable [online], [cit. 2014-05-03]. URL: (http://jimbojw.com/wiki/index.php?title=Understanding_Hbase_and_BigTable)

8 Přílohy

8.1 Zdrojové kódy aplikace pro stahování dat

Zdrojové kódy pro tuto aplikaci jsou na přiloženém CD v adresáři *DataMinerDownloader*.

8.2 Zdrojové kódy aplikace pro zpracování dat

Zdrojové kódy pro tuto aplikaci jsou na přiloženém CD v adresáři *DataMinerGraph*.

8.3 Graf pro Pajek

Soubor s exportovaným programem do formátu pro program Pajek je na přiloženém CD s názvem *graf.net*.