Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Agregátor novinových zpráv

Plzeň, 2014 Michel Soběhart

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. května 2014

Michel Soběhart

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Michalu Konkolovi za jeho odborné vedení, trpělivost a cenné rady, poskytnuté při zpracování tématu bakalářské práce. Dále také Ing. Tomáši Brychcínovi za úpravu projektu HPS pro mé účely. Mé díky patří také rodině a přítelkyni, kteří mi dodali motivaci a plně podporovali.

Abstrakt

Hlavním cílem bakalářské práce je vytvořit program, který shlukuje podobné novinové zprávy. Dalším cílem je vytvořit program pro stahování novinových dat z internetových zdrojů, která budou sloužit jako testovací data. Je potřeba navrhnout posloupnost zpracování dat, určení podobnosti a také princip clusterování dat. Poslední fází je určení správného prahu, po kterém jsou dva soubory podobné. Aplikace byla programována v objektově orientovaném jazyce Java. Při testování navržená aplikace dosáhla 96,85% F-míry.

Abstract

The main goal of the bachelor thesis is to create a program, which clusters similiar RSS newspaper data. Next goal is to create a program for downloading newspaper data from internet sources, which will be used as testing data. Designing a sequence of data processing, determinating similiarities and also principle of the data clustering is required to design. The last phase is selecting the right threshold, after which two files are similar. The application was created using object-oriented Java language. After the testing proposed application has reached 96,85% F-measure.

Obsah

1 Úvod

Novinové agregátory jsou systémy, které sledováním internetových zdrojů novinových zpráv shrnují, co se ve světě internetu píše. Využívají se v databázích novinových článků, kde jsou založeny na shromažďování, analýze a rozdělení velkého množství dat. Používají metody pro předzpracování textu, kdy dále sledují a zpracovávají různé příznaky textu jako je jejich délka, obsažená slova nebo jména osob a organizací. Na základě informací poté určují podobnost mezi články a shlukují je do skupin, ve kterých jsou si dané články navzájem podobné.

V praxi se člověk často setká se situací, kdy si chce přečíst novinové zprávy na internetu. Na procházení všech serverů nemá čas a tak by bylo vhodné, aby data z jeho oblíbených serverů byla na jednom místě. Pořád je nutné hledat podobné zprávy v množství dat, proto je ideální data roztřídit podle tématu, o kterém píší. Mohu si tedy přečíst všechny zprávy týkající se specifického tématu z více serverů po sobě a nemusím je vyhledávat.

Cílem bakalářské práce je nejdříve prozkoumat technologie, které jsou použitelné pro agregátor novinových zpráv. Dále navrhnout systém pro agregátor a následně jej implementovat. V neposlední řadě také zhodnotit výsledky systému a otestovat jeho funkčnost. Pro otestování je nutné vytvořit program pro stahování novinových zpráv z internetových zdrojů.

2 Analýza

2.1 RSS

RSS je datový formát, jehož hlavní výhodou je jeho jednoduchost. Velké množství webových stránek, informačních nebo zpravodajských serverů a v neposlední řadě i blogů, mají možnost odebírat RSS kanály. RSS kanál obsahuje informace nebo novinky poskytované webovou stránkou. Kanál je strukturovaný a tak není složité zpracovat data, která se v něm nacházejí. RSS zdroj je formátován značkovacím jazykem XML. Obsahem RSS kanálu je popis serveru nebo webové stránky, jednoduché popisy společně s názvy zpráv a odkazy na stránku, kde je zpráva publikována na internetu. Dokument popisuje pouze formát a ne jeho vzhled. Ten lze nadefinovat např. pomocí kaskádového stylu CSS[1].

2.1.1 Jazyk XML

RSS vychází a je založen na principu otevřeného značkovacího formátu XML. Tento jazyk vyvinula a dále standartizuje organizace W3C. Původním plánem bylo nahradit formát HTML pro psaní WWW stránek. Informace jsou ohraničeny tzv. párovými značkami[2] a popisují význam dat mezi nimi. Tímto způsobem se zvyšuje hodnota informace v dokumentu a snižují nadbytečné informace. XML se nepoužívá pouze v RSS, ale je využitelné v mnoha jiných aplikacích a formátech. Příkladem může být formát SVG[3], jazyk pro psaní webových stránek XHTML[4] nebo ve formátu dat ATOM[5].

[1] CSS – kaskádový styl pro vzhled stránek – http://www.w3schools.com/css/

[2] Párová značka – je vytvořena na začátku i na konci textu

[3] SVG – formát obrázků ve vektorové grafice – http://www.w3schools.com/svg/

[4] XHTML – Jazyk pro psaní kódu webových stránek – http://www.w3schools.com/html/html_xhtml.asp

[5] ATOM – formát dat podobný RSS – http://atomenabled.org/

2.1.2 Využití RSS

RSS je využitelný v mnoha sférách internetu. Jako běžní uživatelé se nejčastěji setkáme s využitím na zpravodajkých serverech, stránkách s produkty, diskuzních fórech a blozích. Většinou RSS kanály ukazují novinky, které se na serveru publikovali. Na zpravodajském serveru nemusí být pouze jeden informační kanál, ale může se rozdělovat do více kanálů, které lze odebírat. Tím lze zajistit filtrování zpráv.

Uživatel se však s technologií nemusí setkat vědomě, ale nějakým způsobem ji využívá. Pomáhá v podcastu[6], sdílení dat na internetu a orientaci na webových stránkách. Vzhledem k faktu, že RSS se využívá k sdílení dat, slouží také k propojení internetových stránek. Dalším využitím je možnost zobrazení dat na jednom místě a tím úspora času v procházení stránek. K tomu slouží aplikace zvané RSS čtečky.

Mezi argumenty, proč používat RSS čtečky hovoří především časová úspora. Uživatel nemusí procházet desítky oblíbených webů (třeba několikrát za den), což mu zabere nemálo času, ale může se rychle podívat na souhrn toho, co ho zajímá prostřednictvím webové aplikace. Většina četby se navíc soustředí na zprávičky a perexy, takže není třeba čtečku často vůbec opustit. Nezobrazuje se reklama, což má příznivý vliv na spotřebu elektrické energie a výkon počítače a jde tedy o časově i ekologicky zajímavé řešení. Převzato z [2]

2.1.3 Srovnání a syntaxe RSS specifikací

Různé specifikace RSS vznikaly postupně a každou verzi vyvíjela jiná organizace. Právě proto ani neznáme přesný význam zkratky, která se pro tuto technologii využívá, ale máme tři různé významy:

- RDF Site Summary (RSS 0.90 a 1.0)
- Rich Site Summary (RSS 0.91)
- Really Simple Syndication (RSS 2.0)

[6]Podcast – přehrávání hudby

Existují různé verze, které se dnes již nevyužívají. Například verze 0.90 je nyní plně nahrazena 1.0 verzí. Postupem při vyvíjení dalších verzí RSS je většinou přidávání elementů jako parametrů do souboru. Všechny však obsahují deklaraci XML hlavičky s jeho verzí a deklaraci verze RSS dokumentu.

Nejvíce využívanou specifikací je RSS 2.0, na celém internetu ji využívá okolo 80% RSS kanálů.

RSS 0.91

Tato specifikace je nejjednodušší verzí RSS dokumentu a jejím obsahem jsou základní informace o dokumentu, kanálu a zprávě. Také je obsah dokumentu omezen, a to několika vlastnostmi. Prvním limitováním je využití HTML tagů, které v této verzi nejsou podporovány. Další omezení se týkají počtu a délky elementů. Počet zpráv, které lze publikovat, může být maximálně 15 a délky titulku a popisu nesmí překročit 100 a 500 znaků, stejně jako 500 znaků je maximální délka odkazu na soubor nebo kanál. Tuto specifikaci vyvíjela firma UserLand Software. Dále je také vyžadován element <!DOCTYPE> pro deklaraci DTD[7].

- **xml** - volitelný - verze jazyka XML
- **!DOCTYPE** - povinný - definice typu dokumentu
- **rss** - povinný - kořenový element s atributem version
- **channel** - povinný - element kanálu
- **title** - povinný - název kanálu
- **link** - povinný - odkaz na daný kanál, maximálně 500 znaků
- **description** - povinný - popis kanálu, maximálně 500 znaků
- **language** - povinný - jazyk, ve kterém je kanál napsán, vybraný z výčtu jazyků
- **image** - povinný - obrázek, obsahující další elementy
  - **url** - povinný v obrázku - odkaz na umístění obrázku
  - **title** - povinný v obrázku - titulek obrázku

[7] DTD – definice typu dokumentu

link - povinný v obrázku - odkaz na kanál

width - volitelný v obrázku - šířka obrázku, maximálně 144 px

height - volitelný v obrázku - výška obrázku, maximálně 400 px

- **copyright** - volitelný - autorská práva, maximálně 100 znaků
- **managingEditor** - volitelný - email na editora kanálu, maximálně 100 znaků
- **webMaster** - volitelný - emailová adresa webmastera kanálu, maximálně 100 znaků
- **rating** - volitelný - PICS[8] hodnocení stránky, maximálně 500 znaků
- **pubDate** - volitelný - datum publikace
- **lastBuildDate** - volitelný - poslední úprava souboru
- **docs** - volitelný - dokument s popsaným formátem v dokumentu, maximálně 500 znaků
- **textInput** - volitelný - text vložený do souboru
- **skipDays** - volitelný - dny, kdy kanál nepublikuje nové informace
- **skipHours** - volitelný - hodiny, kdy kanál nepublikuje nové informace
- **item** - volitelný - zprávy publikované v RSS, maximálně 15 zpráv

  title - povinný ve zprávě - titulek zprávy, maximálně 100 znaků

  link - povinný ve zprávě - odkaz na plnou verzi zprávy, maximálně 500 znaků

  description - povinný ve zprávě - zkrácený popis zprávy, maximálně 500 znaků [1]

RSS 1.0

Tato specifikace se liší od výrazně ostatních a není s předchozími verzemi zpětně kompatibilní. Není totiž založena na formátu RSS, ale na RDF. Jako kořenový element tedy není uveden <rss>, ale <rdf:RDF>. Element kanálu

[8]PICS – hodnocení stránek na internetu – http://www.w3.org/PICS/

<channel> je uzavřen již před seznamem položek <items>. Novinkou je seznam položek před samotnými položkami a také možnost vytvoření jmenného prostoru, kde si může uživatel vytvořit vlastní elementy nad rámec základní verze. V kořenovém elementu <rdf:RDF> je připojen odkaz na externí dokument parametrem „xmlns:“. Tato verze je vyvíjena skupinou RSS-DEV Working Group.

RSS 2.0

Specifikace 2.0 rozšiřuje verzi 0.91 a odstraňuje některá omezení. Ukázku takového souboru najdete ve výpisu 2.1. V dokumentu již nemusí být deklarován DTD. Počet položek již není nijak omezen oproti předchozí verzi. I jiná omezení z hlediska délky informací v elementu jsou odstraněny, a to v elementech <link> a <description>. Element <language> se z nutného stává pouze volitelným a lze pro zprávu uvést více odkazů v různých jazycích. Další změnou je také fakt, že v textu RSS se mohou vyskytovat HTML tagy. Posledním rozdílem je přidání nových elementů, které jsou všechny volitelné:

- **generator** - řetězec, který ukazuje program generování RSS kanálu
- **category** - kategorie, do které kanál patří
- **cloud** - umožňuje procesům zaregistrovat oblak a být informován o změnách kanálu
- **ttl** - počet minut, jak dlouho může být kanál v mezipaměti před osvěžením od zdroje
- **<item>**

  enclosure - popisuje přílohu, která je ke zprávě přiložena

  source - URL odkaz, odkud pochází kanál

  guid - jednoznačná identifikace URL

Výpis 2.1: Ukázka RSS souboru specifikace 2.0

```
<rss xmlns:szn="http://www.seznam.cz" version="2.0">
 <channel>
  <title>Novinky.cz - Vase zpravy</title>
  <link>http://www.novinky.cz/vase-zpravy</link>
  <description>Novinky.cz - zpravodajsky server</description>
  <language>cs</language>
  <image>
   <link>http://www.novinky.cz/vase-zpravy</link>
   <title>Novinky.cz - Vase zpravy</title>
   <url>http://www.novinky.cz/static/images/logo.gif</url>
   <width>144</width>
   <height>49</height>
  </image>
  <item>
   <title>Policejni pes odhalil v baru drogy</title>
   <description>
   <![CDATA[
      V nedeli priblizne ...
   </description>
   <pubDate/>
   <guid>http://www.novinky.cz/vase-zpravy-21636-.html</guid>
   <link>
     http://www.novinky.cz/vase-zpravy/praha/1297-21636-policejni
        -pes-odhalil-v-baru-drogy.html
   </link>
   <szn:image/>
   <szn:upDate>Sun, 08 Dec 2013 19:17:23 +0100</szn:upDate>
  </item>
 </channel>
</rss>
```

2.2 Novinový agregátor

Novinové (RSS) agregátory jsou servery, které sledováním RSS zdrojů, shrnují, co se kde na internetu píše. Tématicky řazené nadpisy článků s krátkou anotací pak uveřejňují na svých stránkách s odkazem na původní zdroj. Tyto servery vlastně provádí jakýsi monitoring tisku a v tom je jejich přidaná hodnota. [3]

2.3 Příznakový model

Při zpracování textových dokumentů se využívá tzv. příznakový model. Model je tvořen prostorem, který má n-dimenzí. Každá dimenze představuje příznak daného dokumentu.

2.3.1 Příznaky

Základní technikou používanou pro klasifikaci dokumentů jsou příznaky (angl. features). Jako příznak můžeme využít libovolnou informaci o objektu dokumentu. Objektem může být slovo, u kterého lze vypočítat hodnotu četnosti v dokumentu nebo frekvence slova ve všech dokumentech. Dále také slovní spojení (fráze) nebo pojmenované entity. Za pojmenované entity můžeme považovat slova a slovní spojení, které v textu představují geografické názvy, jména osob nebo produktů, názvy organizací nebo firem, časové údaje a další.

2.3.2 Reprezentace dokumentu

Každý dokument ke zpracování musí být rozdělen na jednotlivá slova. Základním principem může být tvoření příznakového modelu pouze z binárních hodnot (1, pokud slovo je v dokumentu obsaženo a 0, pokud není). Častěji se však využívá četnosti daného slova. To však nezohledňuje délku dokumentu, kde stejných slov může být mnohem více. Proto je možné využít metodu zvanou TF-IDF (více v kapitole 2.5). Příklad převzat z [5].

1. **Máma má mísu.**
2. **Máma mele maso.**
3. **Ema nese mísu.**
4. **Ema a máma solí maso.**

Nejprve vytvoříme unikátní slovník slov ve větách. Poté ho podle abecedy seřadíme vzestupně.

[1=„a“, 2=„ema“, 3=„maso“, 4=„mele“, 5=„má“, 6=„máma“, 7=„mísu“, 8=„nese“, 9=„solí“]

Jelikož počet slov ve slovníku je devět, bude model obsahovat stejný počet dimenzí. Obsahuje-li vektor slovo, bude hodnota na této pozici rovna jedné, pokud neobsahuje, je hodnota nulová.

1. **[0,0,0,0,1,1,1,0,0]**
2. **[0,0,1,1,0,1,0,0,0]**
3. **[0,1,0,0,0,0,1,1,0]**
4. **[1,1,1,0,0,1,0,0,1]**

Vzhledem k faktu, že každý jazyk obsahuje mnoho různorodých slov, pohybuje se počet dimenzí prostoru řádově ve statisících. Při takovémto počtu je vhodné slovník zredukovat.

Redukce by se měla týkat slov, která nemají velký vliv na ohodnocení dokumentu. Jsou to tedy slova vyskytující se ve všech dokumentech ve velmi nízkém počtu nebo naopak ta, která jsou zastoupena v každém dokumentu v hojném počtu. V českém jazyce se jedná převážně o zájmena, předložky a spojky.

Dalším faktorem při zpracování textových dokumentů do příznakového modelu je ohýbání slov, rozdíl mezi jednotnými a množnými čísly, skloňování a časování slov. Budeme-li mít zprávu, týkající se města Klatovy, můžeme zde najít slova jako „klatovský“ a „klatovská“. Obě slova mají podobný význam a v analýze by měla být považována za stejná. Proto je pro textovou analýzu důležité předzpracování textu, kde základními principy jsou lemmatizace a stemming.

2.3.3 Stemming

Stemming je operace pro úpravu slov. Pro všechna slova se snaží vrátit kmen slova[9]. Stemmer pro český jazyk je složitější vzhledem např. k anglickému jazyku. V českém jazyce využíváme často předpon a přípon (např. pro záporné slovo často přidáváme předponu ne-), ale hlavně také změny slova takové, že se změní i jeho kmen. Například u podstatného jména se můžeme často setkat se třemi tvary slova, kdy kmen je pokaždý jiný (viz Tabulka 2.1).

[9]kmen – tvar slova po odstranění koncovky nebo předpony

Pád	Tvar slova	Stemmer
1. pád jedn. čísla	kočka	kočk-
3. a 6. pád jedn. čísla	kočce	kočc-
2. pád množ. čísla	koček	koček-

Tabulka 2.1: Ohýbání českých slov

2.3.4 Lemmatizace

Lemmatizace je operace podobná stemmeru, která se s ním často mylně zaměňuje. Při této operaci totiž získáme tvar zvaný lemma[10]. Výsledné slovo by tedy mělo vypadat jako základní slovníkový tvar. Máme-li tedy slovo „kočce“, výsledkem by měla být lemma „kočka“.

Je viditelné, že lemmatizace vrací slovo, které lépe vyjadřuje význam původního slova.

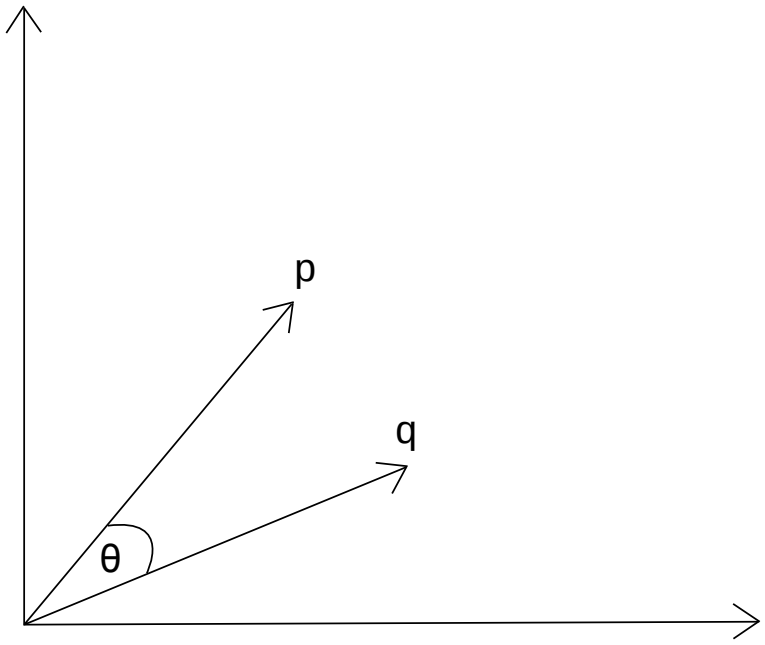
2.4 Podobnost dokumentů

Využití příznakového modelu je důležité pro vypočtení podobnosti dokumentů. Podobnost dokumentů se nachází v intervalu $\langle -1, 1 \rangle$. Zprávy jsou vyjádřeny vektory, mezi nimiž lze vypočítat podobnost pomocí matematických přístupů. Vektory p a q jsou dva dokumenty, mezi kterými chceme znát míru podobnosti. Očekává se, že zprávy, které budou popisovat stejnou událost, budou také využívat podobná slova. Základní podobnosti jsou popsány a převzaty z [5].

Pokud vypočítáme vzdálenost $d(p, q)$, je výsledný interval jiný než pro podobnost. Musíme tedy dále upravit vzorec, abychom získali hodnotu v intervalu $\langle -1, 1 \rangle$.

$$podobnost = \frac{1}{1 + d(p, q)}$$

[10]lemma – může být první pád jednotného čísla nebo v určitých situacích také kmen slova



Obrázek 2.1: Ukázka vektorů pro výpočet podobnosti

2.4.1 Eukleidovská vzdálenost

Využívá rozdílu dvou vektorů. Výsledná vzdálenost je reprezentována vzdáleností koncových bodů vektoru.

$$d\left(p,q\right) = \sqrt{\sum \left(p_i - q_i\right)^2}$$

2.4.2 Vážená suma

Jedna z aplikací vážené sumy (weighted sum) je vyhledávání pomocí klíčových slov v dokumentu. Z těchto slov je sestaven vektor q a dokument poté představuje vektor p. Výsledná podobnost je dána součtem četností slov v dokumentu, které se shodují se slovy v dotazu.

$$d\left(p,q\right) = \frac{\sum p_i q_i}{\sum p_i}$$

2.4.3 Kosinová podobnost

Kosinová podobnost (cosine similarity) představuje míru podobnosti dvou vektorů, která je získána výpočtem kosinu úhlu ležícím mezi těmito vektory.

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cdot \cos\theta$$

$$Podobnost\,(a,b) = \cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Obecně může kosinová podobnost nabývat hodnot z intervalu $\langle -1, 1\rangle$. Hodnota -1 poté značí dokumenty sobě opačné a hodnota +1 značí shodné dokumenty. Pokud jsou oba vektory všude kladné, výsledná hodnota kosinové podobnosti by byla v intervalu $\langle 0, 1\rangle$.

2.4.4 Manhattonská vzdálenost

Počítá se jako součet pravoúhlých kroků mezi body, které určují vektory p a q.

$$d\,(p,q) = \sum |p_i - q_i|$$

2.4.5 Cebyševova vzdálenost

Vyjadřuje maximální rozdíl složek dvou vektorů p a q.

$$d\,(p,q) = max\,|p_i - q_i|$$

Je možné využít další metriky pro určení podobnosti vektorů. Více způsobů můžete vidět v [6].

2.5 TF-IDF

TF-IDF je metoda pro určování váhy slova. Zkratka se skládá ze dvou základních termínů, a to Term Frequency (dále jen TF) a Inverse Document Frequency (dále jen IDF).

Vzhledem ke skutečnosti, že každý dokument je jinak dlouhý, nesmí docházet k nadhodnocování delších dokumentů, které by měly stejnou četnost daného slova. Vydělíme tedy četnost počtem slov v dokumentu a výsledkem je hodnota $TF(i) = \langle 0, 1 \rangle$. Obecný vzorec pro výpočet je:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

kde n_i je vybrané slovo a $\sum_k n_k$ je počet všech slov v daném dokumentu j.

IDF určuje důležitost slova. Po vypočtení TF je každé slovo stejně důležité a nedělají se mezi nimi rozdíly. Avšak pro tuto část výpočtu musíme také předejít nadhodnocení slov, jako jsou „a“, „se“, které najdeme skoro v každém dokumentu, od slova „Jágr“, která najdeme jen převážně ve specifických dokumentech o hokeji. Čím se tedy slovo častěji vyskytuje v dokumentech, tím je méně důležité a naopak.

$$IDF_{i,D} = \log \frac{|D|}{|\{j : t_i \in d_j\}|},$$

kde $|D|$ je počet všech dokumentů a $|\{j : t_i \in d_j\}|$ je počet dokumentů, ve kterých se vyskytuje dané slovo.

Výsledné TF-IDF se poté vypočítá vynásobením obou předchozích výsledků.

$$tfidf_{i,j} = TF_{i,j} \cdot IDF_{i,D}$$

Hodnota TF-IDF bude vysoká pro ta slova, která se často vyskytují ve zpracovávaném dokumentu, ale v ostatních dokumentech se vyskytují minimálně.

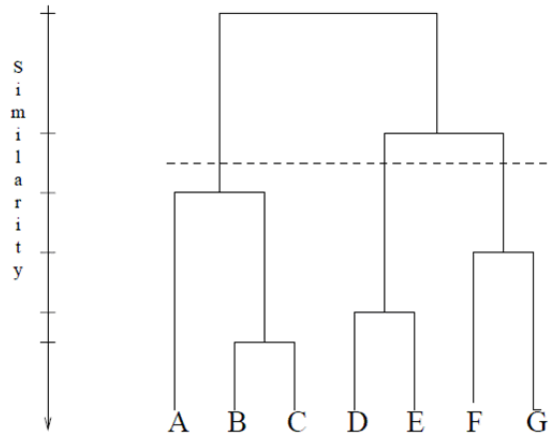
2.6 Shlukování dokumentů

Shlukování dokumentů (tzv. clustering) je ve své podstatě dělení dokumentů do podmnožin, které mají podobné téma nebo podobný obsah. Cíl je takový,

že dokumenty ve shluku (clusteru) se musí co nejvíce podobat, avšak dané shluky by se od sebe měli co nejvíce lišit. Při vytváření takovýchto skupin neznáme jejich počet, nemáme o nich žádnou představu.

2.6.1 Hierarchické clusterování

Hierarchické clusterování je založeno na principu rozkladu. Tato metoda shlukování lze znázornit binárním stromem, kde každý shluk je jeden uzel, tzv. dendrogramem (viz Obrázek 2.2). V našem případě je svislý směr čar vzdálenost mezi shluky a vodorovný směr čar stupeň rozkladu shluků. Podrobnější a více informací naleznete v [9] a [10].



Obrázek 2.2: Ukázka dendrogramu převzata z [8]

Aglomerativní přístup

Aglomerativní shlukování začíná nultým rozkladem, kdy máme n jednoprvkových objektů (n je počet prvků shlukování). Dalším postupem je sloučení shluků, které jsou nejpodobnější dle daného kritéria. Vznikne nový shluk o dvou prvcích a v každém dalším kroku z nejpodobnějších shluků nižšího stupně rozkladu další nový shluk. Každým krokem chceme zvyšovat vzdálenost mezi shluky a konec shlukování může být na základě vzdálenosti nebo počtu shluků. Aglomerativní shlukování se dělí na základě kritéria, kterým se vybírají nejpodobnější shluky.

Divizní přístup

Divizí přístup je opačný k aglomerativnímu přístupu. Nejdříve máme množinu objektů ke zpracování jako jeden shluk o n položkách. Dělíme tento shluk na dva menší, které jsou opět v daném čase optimální ke kritériu. Na konci této metody budou všechny shluky jednoprvkové. Má vysokou složitost (exponenciální), je tedy vhodný a proveditelný pouze na malé množství prvků.

2.6.2 Nehierarchické clusterování

Nehierarchický přístup rozkládá množinu prvků do podmnožin dle daného kritéria. Vybrané metody nejdříve vytvoří počáteční rozklad, který se dále již nedělí. Tento rozklad poté pouze optimalizujeme a upravujeme. Nevytváříme tedy stromovou strukturu, nýbrž K shluků.

K-Means clustering

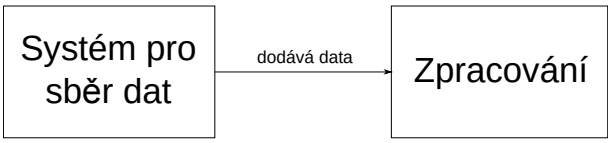
Tento algoritmus patří k jednoduchým způsobům přes zadaný počet shluků. Jedná se však pouze o lokálně optimální řešení, protože závisí na počátečních podmínkách. Výhodou oproti hierarchickým metodám je fakt, že dokáže pracovat i s vysokým počtem dat. Nevytváří totiž matici podobnosti, ale pracuje přímo s daty.

Fuzzy C-Means

Tato metoda přímo neříká do jakého shluku patří jaký objekt, ale pravděpodobnosti, s jakými do jednotlivých shluků patří. Pomocí této metody tedy můžeme poznat, zda-li je do shluku správně zařazen nebo případ opačný, zda-li shluk správně reprezentuje.

3 Návrh

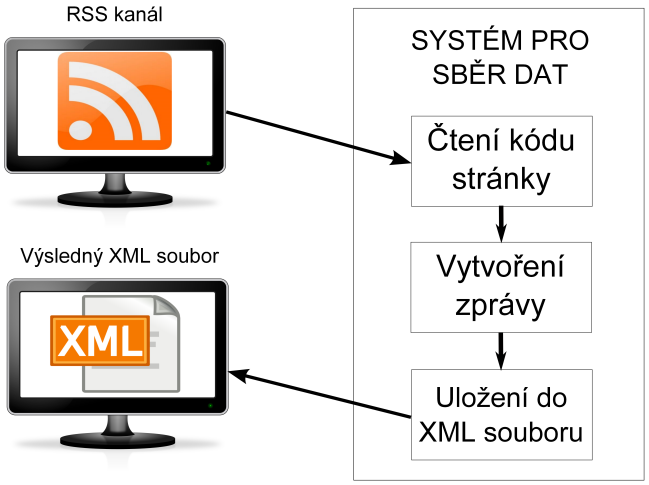
Celý program bude rozdělen do dvou hlavních částí. Nejdříve musím získat data pro agregátor, poté data zpracovat.



Obrázek 3.1: Základní návrh programu

3.1 Systém pro sběr dat

První částí celého projektu je sběr dat z internetových RSS kanálů. Cílem je uložit data ve vhodném a dobře čitelném formátu, aby se dále dala snadno zpracovat. Vzhledem k potřebám agregátoru není nutné využívat všechny vlastnosti dokumentu, které novinová zpráva nabízí. Důležitými vlastnostmi jsou nadpis, odkaz a popis zprávy. Další volitelné vlastnosti, jako jsou GUID, autor a jiné, potřeba nejsou. Pro úplnost je však dobré je také uložit. Výstupem tedy bude databáze zpráv ve formátu *.xml*.



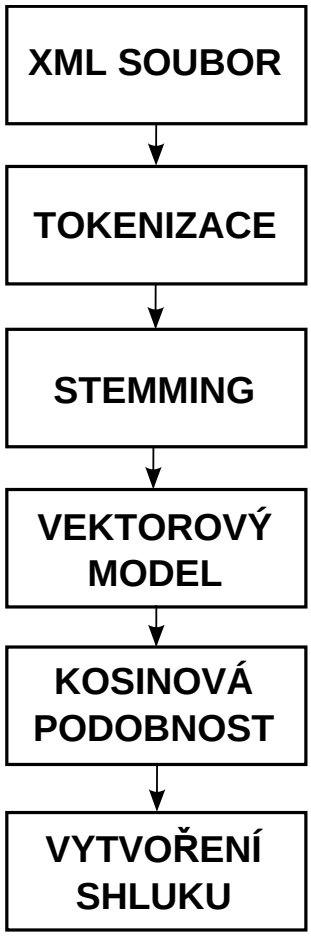
Obrázek 3.2: Návrh systému pro sběr dat

3.2 Zpracování

Druhou částí je zpracování souborů a vytvoření shluků (clusterů) podobných dokumentů. Data se nejdříve musí předzpracovat pro vyšší přesnost, vypočítat podobnost a vytvořit clustery. Pro lepší ovladatelnost bude vytvořeno grafické uživatelské rozhraní.

Předzpracování dokumentu probíhá v několika částech (viz Obrázek 3.3):

- **XML soubor** – soubor bude mít jednoznačně určený formát. Základním elementem je **zprava**, která obsahuje subelementy jednoznačně popisující zprávu. Subelementy jsou **datum, odkaz, guid, autor, titulek, popis.**
- **Tokenizace** – rozdělení souboru na jednotlivá slova (tzv. tokeny).
- **Stemming** – stemming je jednodušší na implementaci než lemmatizace. Na Katedře informatiky a výpočetní techniky ZČU je vyvíjen stemmer, který mohu využít v mé bakalářské práci. Více o tomto projektu naleznete na [7].
- **Příznakový model** – příznakový model bude obsahovat hodnoty TF-IDF slova. Je to způsob, který zhodnotí dokumenty lépe než jen jejich počet, protože je počítán přes všechna testovací data. Do trénovacích dat, kde budu počítat IDF slov, nezahrnuji nově stažená data, která se stanou daty testovacími.
- **Kosinová podobnost** – pro svůj účel jsem si vybral kosinovou podobnost, která vrátí číslo v intervalu $\langle 0, 1 \rangle$, kde vysoká hodnota znamená vyšší podobnost slov v dokumentech a naopak nízká hodnota nižší podobnost dokumentů.
- **Vytvoření shluku** – rozdělení dat do shluků dle podobnosti mezi nimi a hodnotě prahu. Při podobnosti přesahující zadaný prah budou dva soubory uznány za podobné a vloženy do jednoho clusteru.



Obrázek 3.3: Návrh posloupnosti zpracování dat do clusterů

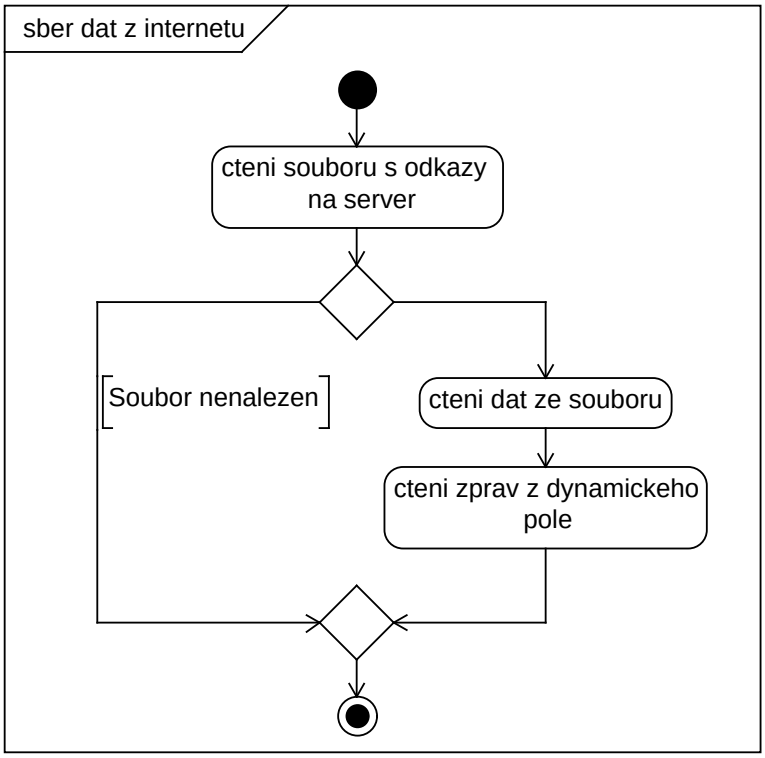
4 Implementace programu

Obě části implementuji v objektově orientovaném jazyce Java.

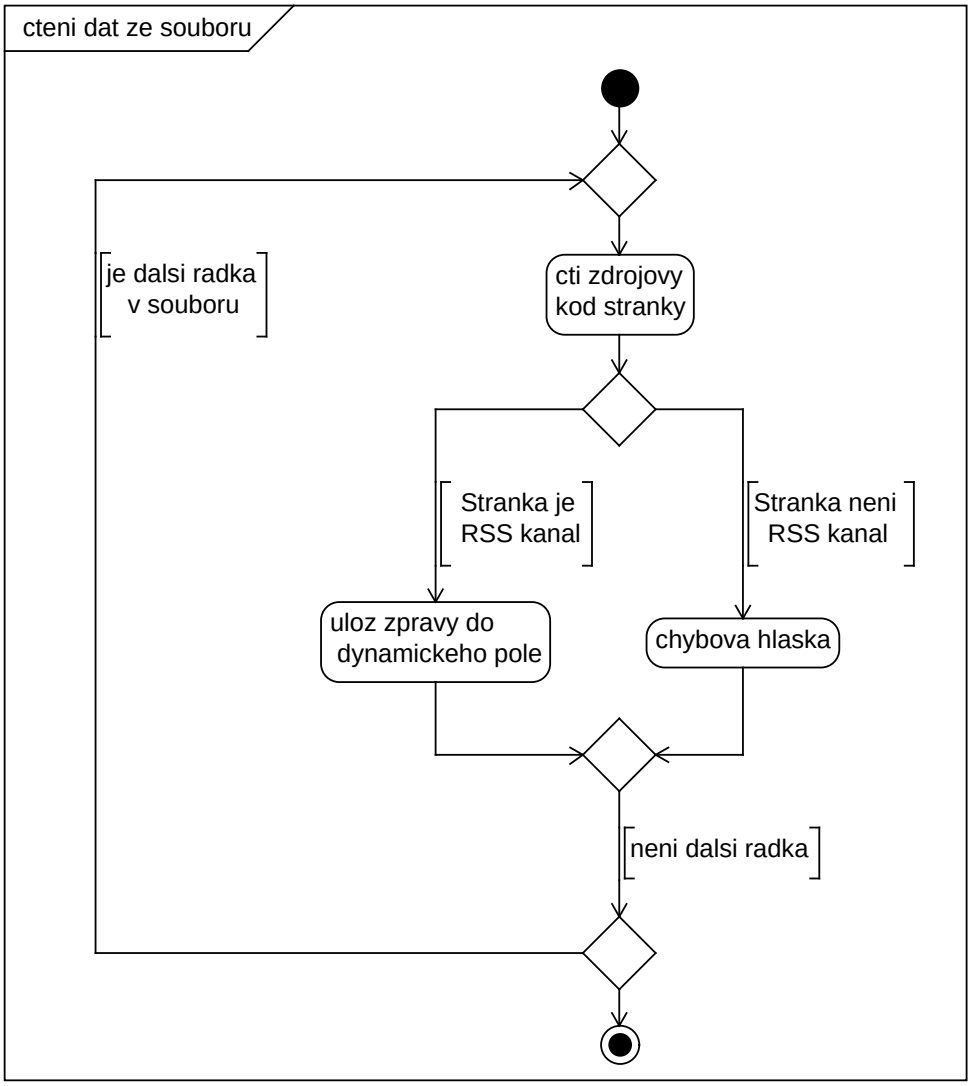
4.1 Systém pro sběr dat

4.1.1 Vývojové diagramy

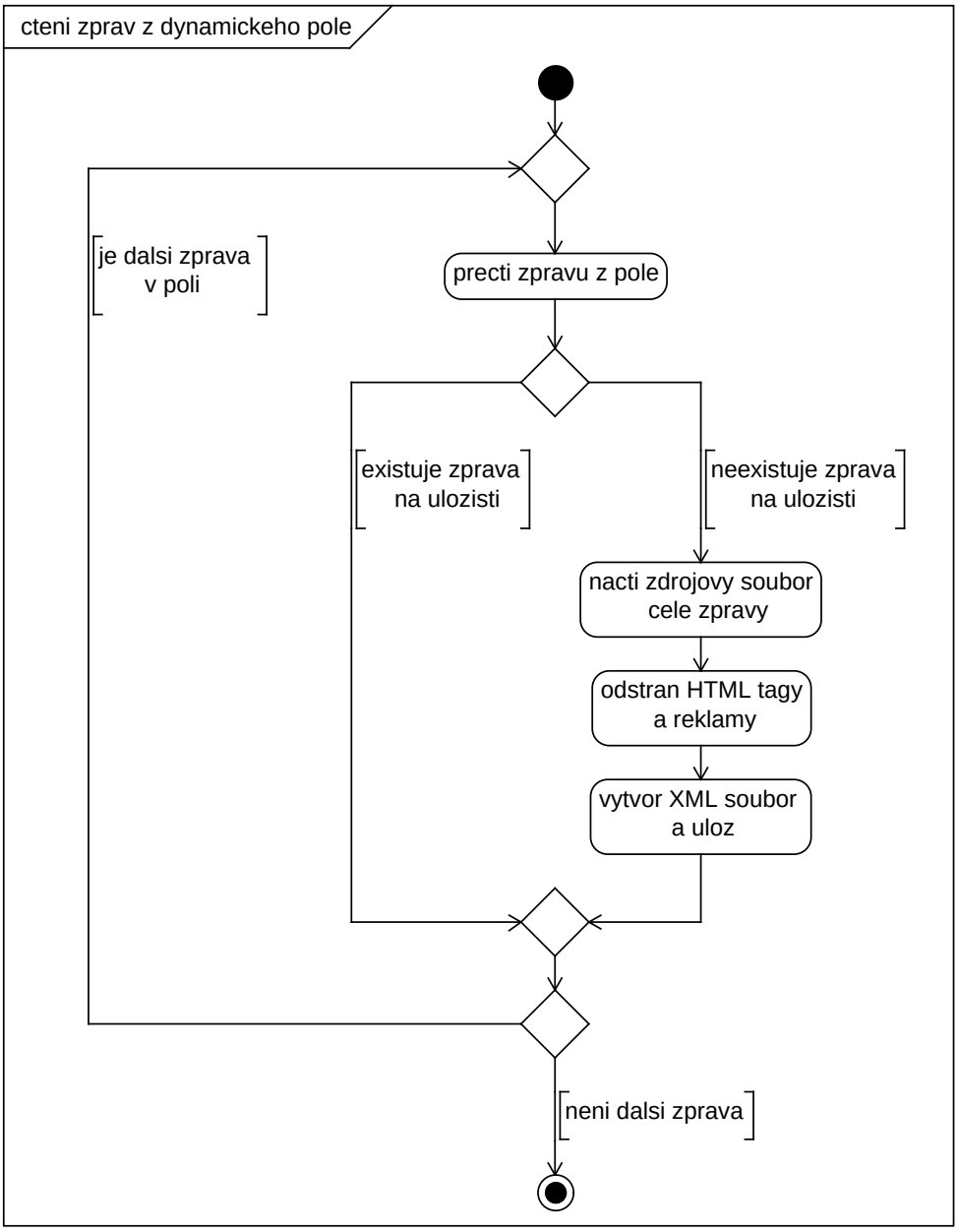
Vývojové diagramy využité při implementaci systému. První diagram je zobrazení celého diagramu (viz Obrázek 4.1), do kterého poté spadají dva další diagramy (viz Obrázky 4.2 a 4.3).



Obrázek 4.1: Vývojový diagram



Obrázek 4.2: Vývojový diagram prohledávání souboru



Obrázek 4.3: Vývojový diagram procházení dynamickým polem

4.1.2 Třída Feed.java

Přepravka pro uložení důležitých atributů o jednotlivých novinových zprávách. Atributy jsou titulek, popis, odkaz na celou zprávu, autor, datum publikace a GUID identifikační číslo zprávy. Tyto atributy jsou soukromé a přístup k nim je pomocí getterů a setterů. Dále obsahuje dynamické pole zpráv.

```
public static List <Feed> feeds = new ArrayList <Feed>();
```

Dynamické pole slouží k uložení všech načtených zpráv a následnému zpracování.

4.1.3 Třída Server.java

Přepravka pro uložení důležitých atributů jednotlivých RSS serverů. Atributy jsou titulek, popis, odkaz na RSS server, copyright, datum poslední úpravy a jazyk, ve kterém je napsán. Všechny atributy jsou privátní a přístup je pomocí getterů a setterů.

4.1.4 Třída HTML.java

Třída slouží k vyjmutí textu z kódu HTML stránek, resp. odstranění HTML značkovacích tagů ze zdrojového kódu stránky. Třída využívá projekt BoilerPipe (více v [4]) pro odstranění HTML tagů z textu a vychází z něj. Obsahuje tyto metody:

- `public HTML(String link)`
  Konstruktor pro vytvoření instance třídy. Instance vytvořena, pokud je url stránky správně zadané.
- `public String deleteTags()`
  Načte zdrojový kód z webové stránky, odstraní HTML značky a vrátí jej.

4.1.5 Třída DataRead.java

Třída zpracovává data ze zdrojového kódu RSS serveru do formátovaného XML souboru v počítači. Při zpracování plného textu zprávy využívá třídy `HTML.java`. Pro zápis do souboru používá třídu `XML.java`. Obsahuje tyto metody:

- `public DataRead (String serverURL)`
  Konstruktor pro vytvoření instance třídy. Vytvoří se, pokud je url serveru správně zadané.
- `private static String convertFileName(Feed feed)`
  Odstraní nepodporované znaky z názvu souboru.
- `private static String getValue(String tag, Element element)`
  Nalezne v daném elementu párovou značku a vrátí data mezi nimi, pokud párová značka není obsažena v elementu, vrací prázdný řetězec.
- `public void serverInformation(Document doc)`
  Výpis informací o serveru, ze kterého stahujeme data.
- `public void feedInformation(Document doc)`
  Uložení zprávy publikované na serveru do dynamické struktury.
- `public void readFeed()`
  Získání publikované zprávy ze serveru. Pokud nejsme připojeni k internetu, program vypíše chybovou hlášku a stahování ukončí.
- `public static void dataToFile() throws IOException`
  Vytvoření XML dokumentů s pevně danou strukturou (viz 4.2) ze zpráv, které ještě neexistují na disku ve složce pro stahování.

4.1.6 Třída ServerRead.java

Třída zpracovává textový soubor s odkazy na RSS servery, využívá třídy `DataRead.java`. Obsahuje tyto metody:

- `public ServerRead(String filename)`
  Konstruktor pro vytvoření instance třídy. Vstupem je odkaz na soubor odkazující na RSS servery.

- `private void readServerFeed(String link)`
  Načtení zpráv z jednoho RSS serveru.
- `public void readServers()`
  Metoda postupně načítá odkazy na RSS servery, ukládá do dynamické struktury a ukládá data na úložiště. Pokud nelze zapsat na disk nový soubor XML.

4.1.7 Třída Download.java

Hlavní třída projektu. Obsahuje tyto metody:

- `public static void main(String [] args)`
  . Hlavní metoda projektu.

4.1.8 Třída XML.java

Třída pro formátování XML dokumentu. Využívá se se standartními metodami pro zápis o souboru. Obsahuje tyto metody:

- `public String header()`
  Sestavení formátovaného řetězce pro hlavičku XML dokumentu.
- `public String startElement()`
  Sestavení formátovaného řetězce pro začátek hlavního elementu XML dokumentu.
- `public String endElement()`
  Sestavení formátovaného řetězce pro konec hlavního elementu XML dokumentu.
- `public String element(String name, String content)`
  Sestavení formátovaného řetězce vnitřního elementu XML dokumentu.

4.1.9 Formáty vstupu a výstupu

Vstupem do systému je soubor s odkazy na servery, které jsou ve formátu RSS (viz Výpis 4.1). Každý odkaz musí být na nové řádce.

Výpis 4.1: Ukázka vstupního souboru

```
http://aktualne.centrum.cz/feeds/rss/domaci/regiony/pardubicky/?
    photo=1
http://www.novinky.cz/rss2/vase-zpravy/jihocesky-kraj/ceske-
    budejovice/
```

Výstupem programu jsou soubory se zprávami strukturované dle určitých pravidel. Soubor se bude nadále snadno zpracovávat, jelikož má pevně danou strukturu (viz Výpis 4.2).

Výpis 4.2: Ukázka výstupního strukturovaného souboru se zprávou

```
<zprava>
 <datum>15.12.2013</datum>
 <odkaz>
  http://www.rozhlas.cz/_zprava/znakarka-baumrtova-si-z-me-v-
      dansku-odvazi-ctvrtou-medaili--1292768
 </odkaz>
 <guid></guid>
 <autor></autor>
 <titulek>
  Znakařka Baumrtová si z ME v Dánsku odváží čtvrtou medaily
 </titulek>
 <popis>
  Česká znakařka Baumrtová si na ME v Dánsku vybojovala čtvrtou
      medaili...
 </popis>
</zprava>
```

4.1.10 Chybová a informativní hlášení programu

[**ERROR**] **Feed URL is broken** – odkaz na webovou stránku se zprávou není korektně napsán nebo daná stránka neexistuje

[**ERROR**] **Server URL is broken** – odkaz na webovou stránku serveru není korektně napsán nebo daná stránka neexistuje

[**ERROR**] **BoilerPipe inner error** – vnitřní chyba projektu BoilerPipe

[**ERROR**] **Can not create HTML document** – chyba při vytváření HTML dokumentu pomocí `HTMLFetcher`

[**ERROR**] **Can not create Text document** – chyba při vytváření textového dokumentu pomocí `BoilerpipeSAXInput`

[**ERROR**] **File with RSS servers not found** – textový soubor s RSS servery nebyl nalezen

[**ERROR**] **File with RSS servers not found** – textový soubor s RSS servery nebyl nalezen

[**ERROR**] **Data could not be written to XML file** – XML soubor s novou zprávou nelze zapsat na disk

[**ERROR**] **Internet connection is disabled** – nelze se připojit k internetu

[**ERROR**] **Content of website is incorrect, document can not be create** – XML formát webové stránky není kompatibilní s RSS, nebude vytvořen žádný nový dokument z daného serveru

[**ERROR**] **Not enough parameters. First is path of RSS file and second is directory for downloading.** – málo parametrů programu

[**ERROR**] **RSS file is broken.** – RSS soubor neobsahuje správné informace

[**INFO**] **Program starts reading file and feeds from RSS servers** – program čte soubor s RSS servery a stahuje z nich novinové zprávy

[**INFO**] **Start program. Reading servers** – začátek programu

[**INFO**] **All files downloaded to *path*.** – konec programu

[INFO] Program starts to process feeds – program zpracovává nové novinové zprávy

[INFO] Program will be stopped now – stahování je zastaveno kvůli chybě

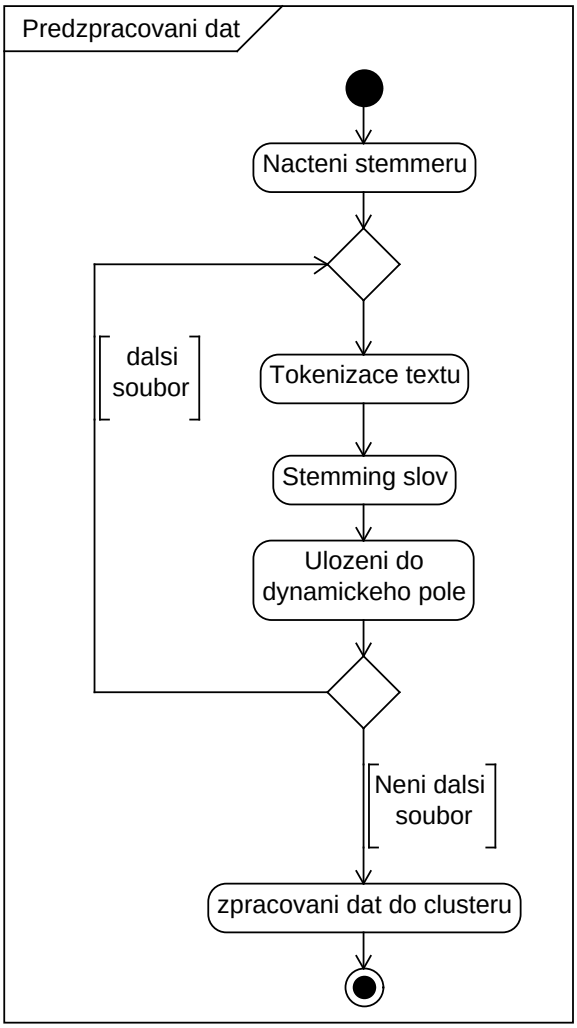
[INFO] RSS Feed (TITLE= titulek, DESCRIPTION= popis, LINK= odkaz, AUTHOR= autor, GUID= guid) – informace o nově stahované zprávě

[INFO] Server (TITLE= titulek, COPYRIGHT= práva, DESCRIPTION= popis, LANGUAGE= jazyk, LINK= odkaz, PUBLICATION DATE= datum publikace) – informace o RSS serveru

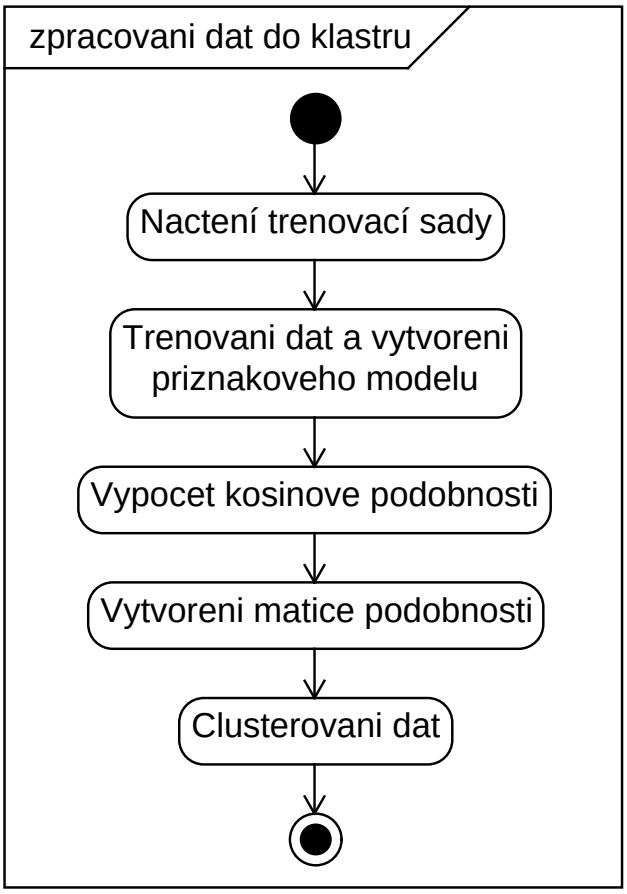
4.2 Zpracování zpráv

4.2.1 Vývojové diagramy

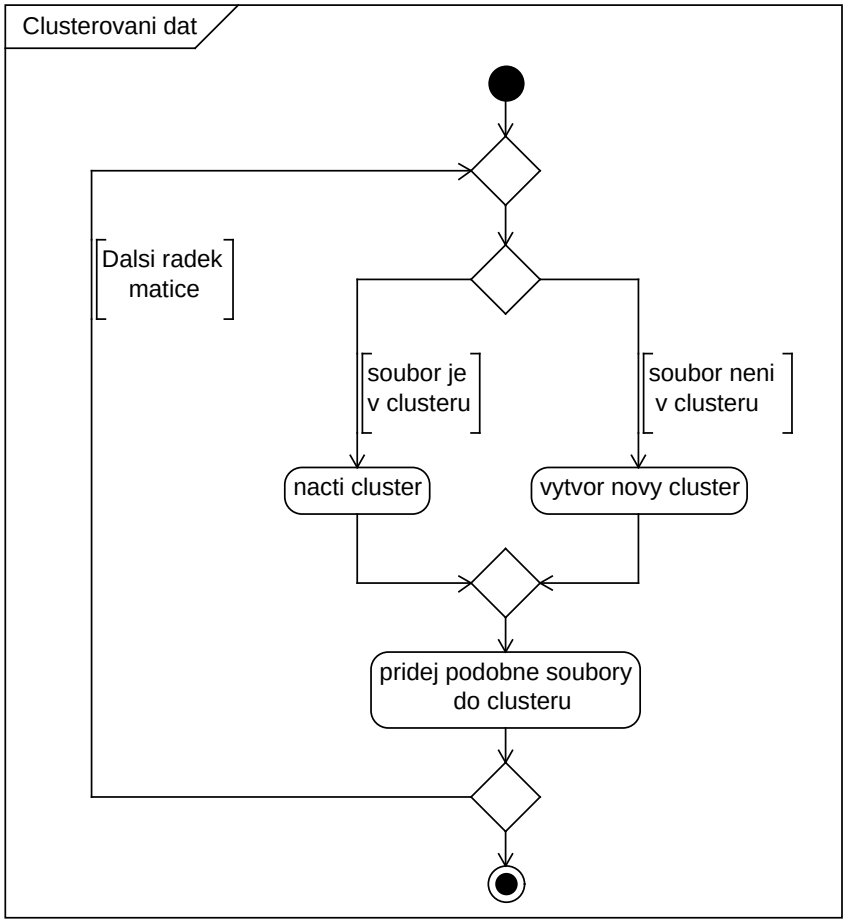
Vývoj programu je rozdělen do tří částí. Jako první je předzpracování dat (viz Obrázek 4.4), poté samotné zpracování dat do příznakového modelu (viz Obrázek 4.5) a nakonec clusterování dat (viz Obrázek 4.6).



Obrázek 4.4: Vývojový diagram předzpracování



Obrázek 4.5: Vývojový diagram zpracování dat



Obrázek 4.6: Vývojový diagram clusterování dat

4.2.2 Třída Cluster.java

Třída slouží jako přepravka pro uložení clusterů dat. Obsahuje jako privátní atribut dynamické pole s čísly souborů. Také gettery a settery. Dále obsahuje metodu:

- `public static void clusterSimilarity(DenseDoubleMatrix simMatrix)`
  Vytvoření clusterů z matice podobnosti.

4.2.3 Třída Graphical.java

Třída slouží k vykreslení grafických výsledků clusterování.

- `public void paint(Graphics g)`
  Vykreslení grafických výsledků clusterování na plátno.

4.2.4 Třída GUI.java

Třída slouží jako grafické uživatelské rozhraní aplikace. Obsahuje atributy představující jednotlivé komponenty rozhraní.

- `private void initComponents()`
  Inicializace komponentů pro uživatelské rozhraní.
- `private void initPanels()`
  Inicializace panelů a rozložení uživatelského rozhraní.
- `private void addListeners()`
  Přidání posluchačů na komponenty.
- `public void actionPerformed(ActionEvent e)`
  Metoda pro posluchače tlačítek.
- `private void controlFiles(File file)`
  Kontrola souborů ve vybraném adresáři, který musí obsahovat alespoň dva soubory s koncovkou *.xml*.

- `private void writeResults(String [] soubory)`
  Textový výpis výsledků do uživatelského rozhraní.
- `public static void writeText(String text)`
  Výpis textu do logu.
- `private static void createGUI()`
  Vytvoření `JFrame` instance s grafickým uživatelským rozhraním.
- `public static void main(String[] args)`
  Hlavní metoda projektu.

4.2.5 Třída Processing.java

Třída slouží k předzpracování, zpracování a shlukování dat.

- `public static void start(String directory, double threshold)`
  Cyklus předzpracování, zpracování a shlukování dat.
- `private static void xmlFiles(String [] files)`
  Vybrání dat s koncovkou *.xml* do dynamického pole **names**.

4.2.6 Třída Stemming.java

Třída slouží ke stemmingu dat. Využívám zde projekt HPS (High Precision Stemmer) [7].

- `public String[] fileStemmer(String[] tokens)`
  Z jednotlivých tokenů vytvoří jejich kmeny.

4.2.7 Třída TextAreaObserver.java

Třída slouží jako pozorovatel ke vydavateli `TextObservable` a mění text v textovém poli. Dědí od třídy `Observer`. Obsahuje tuto metodu:

- `public void update(Observable obs, Object object)`
  Aktualizace textu v textovém poli.

4.2.8 Třída TextObservable.java

Třída slouží jako vydavatel, upozorňuje pozorovatele na svojí změnu. Implementuje rozhraní `Observable`. Obsahuje tyto metody:

- `public void setText(String textInput)`
  Přidá vložený text do textového pole a upozorní posluchače.
- `public String getText()`
  Vrátí text z textového pole.

4.2.9 Třída TfIdf.java

Třída slouží k vytvoření příznakového modelu. Příznaky jsou TF-IDF slov.

- `public void extractFeature(ListIterator<String[]> iterator, FeatureVectorGenerator generator)`
  Vytvoření příznakového modelu.
- `public int getNumberOfFeatures()`
  Počet příznaků modelu.
- `public void train(TrainingInstanceList<String[]> trainData)`
  Metoda pro trénovací data, která získá ze všech vstupních dokumentů jejich příznaky.
- `private double inverseDocumentFrequency(int allDocuments, int documentsWithTerm)`
  Vypočítá hodnotu inverzní četnosti slova ve všech dokumentech dle vzorce 2.5.
- `private double termFrequency(int totalTerms, int countTerm)`
  Vypočítá hodnotu frekvence termu v dokumentu pomocí vzorce 2.5.

- `private double countTfIdf(double tf, double idf)`
  Vypočítá hodnotu TF-IDF dle vzorce 2.5.

4.2.10 Třída Tokenizer.java

Třída pro tvorbu tokenů z textu. Obsahuje privátní proměnnou `tokens`, do kterého se jednotlivé tokeny ukládají.

```
private ArrayList <String> tokens;
```

- `public String getText(String filename)`
  Získání popisu zprávy z XML dokumentu.
- `private String getValue(String tag, Element element)`
  Získání obsahu elementu v XML souboru.
- `public void tokenizeText(String description)`
  Samotná tokenizace textu. Jednotlivé tokeny uloží do dynamického pole *tokens*.

4.2.11 Třída TrainData.java

- `public static DoubleMatrix train(String [][] dataTrain, String [][] dataExtract, String featureSetFile, String trainFile)`
  Načte data z trénovacího souboru **trainFile**, konfigurační data ze souboru **featureSetFile** a podle příznakového modelu vypočítá podobnost.
- `public static DoubleMatrix train(String [][] dataTrain, String [][] dataExtract, int threshold, String trainFile)`
  Načte data z trénovacího souboru **trainFile**, prah nejnižší počet daného slova ve všech dokumentech z parametru **threshold** a podle příznakového modelu vypočítá podobnost.
- `private static DoubleMatrix similarityCount(DoubleMatrix dataMatrix)`
  Výpočet podobnosti mezi všemi soubory z příznakového modelu a uložení do matice podobnosti.

- `private static FeatureSet<String[]> readObject(String trainFile)`
  Zpracování objektu ze souboru pomocí **ObjectInputStream**.
- `public static void writeObject(FeatureSet <String[]> set, String filename)`
  Zápis objektu do souboru pomocí **ObjectOutputStream**.

4.2.12 Konfigurace programu

Konfigurační soubor `config.xml` obsahuje parametr **threshold**, implicitně nastavený na hodnotu 2. Jedná se o minimální počet slova ve všech dokumentech. Bude-li slovo mít nižší počet výskytů než je zadaný práh, nebude započítáno do příznakového modelu.

Výpis 4.3: Konfigurace souboru *config.xml*

```
<featureSet>
 <feature>
  <class>cz.zcu.fav.bc.TfIdf</class>
  <params>
   <param type="java.lang.String">
    <name>threshold</name>
    <value>2</value>
   </param>
  </params>
 </feature>
</featureSet>
```

Soubor `train.bin` obsahuje binární reprezentaci objektu příznakového modelu natrénovaného na 18437 souborech, které nezahrnují data, na kterých byl systém testován.

Soubor `cz.bin` je model v binárním formátu pro **HPS**.

4.2.13 Chybové a informativní hlášení programu

[**ERROR**] **Not enough XML files in directory** – ve vybrané složce je příliš málo *.xml* dokumentů

[**ERROR**] **XML file is empty** – soubor ke zpracování nemá žádný popis

[**INFO**] **Processing *filename*** – zpracování daného souboru

[**INFO**] **Program started. Loading stemmer** – začátek běhu programu a načítání stemmeru

[**INFO**] **Start processing files** – začátek zpracování dat

[**INFO**] **Start creating matrices** – vytváření matice podobnosti

[**INFO**] **Start clustering files** – clusterování dat

[**INFO**] **Start writing results** – zápis výsledků

[**INFO**] **Data Matrix: COLUMNS: columns number , ROWS: rows number** – počet řádek a sloupců příznakového modelu

[**INFO**] **Similarity Matrix: COLUMNS: columns number , ROWS: rows number** – počet řádek a sloupců matice podobnosti

5 Testování

5.1 TF-IDF

Testování části TF-IDF je zajištěno pomocí automatických JUnit testů. Vytvořil jsem si tři testovací entity, které by měli pokrýt spektrum možností, a vypočítal hodnoty. Výsledné porovnání proběhne na příznakovém modelu.

1. *a, už, já, v, Karel, a, a, myš, pes, už,*
2. *už, on, a, ke, pes, já, a, ke, už, už, já, v, já, Karel, ona,*
3. *a, Pepa, v, ke, v, Fík, ona, už, Fík, v*

Počet dokumentů, ve kterých se musí nacházet dané slovo, je nastaven na dva. Neměly by se tedy započítat slova *Fík*, *myš*, *on* a *Pepa*. Ostatní slova by se měla do příznakového modelu zahrnout. Jejich počet je devět, stejný jako počet řádek matice. Jelikož soubory jsou tři, tak počet dimenzí matice je také shodný. Výsledná hodnota je vypočítána dle vzorců 2.5.

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.01173941727037875 & 0.017609125905568124 \\ 0.0 & 0.0234788345407575 & 0.017609125905568124 \\ 0.017609125905568124 & 0.01173941727037875 & 0.0 \\ 0.017609125905568124 & 0.03521825181113625 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.09542425094393249 \\ 0.017609125905568124 & 0.01173941727037875 & 0.0 \end{bmatrix}$$

5.2 Shlukování

Testování shluků se dá rozdělit do dvou částí. První částí je hledání optimálního prahu podobnosti, který učí, zda-li jsou data podobná či nikoliv. Druhou částí je kontrola výsledků agregátoru se shluky ručně vytvořenými.

Pro správné otestování jsem si vytvořil ručně shluky dat z pěti dnů. Data za každý den jsem rozdělil do skupin dle podobnosti a vytvořil soubor reprezentující skupiny. Samotné testování probíhalo nejdříve ručně, kdy jsem se snažil o nalezení optimálního prahu podobnosti.

Již během této fáze jsem našel neshody se mnou shlukovanými daty. Shlukovala se data, která dle názvu byla odlišná. V prvním případě jsem po opětovném otevření zjistil, že z několika druhů stránek nebyl odstraněn webový rozcestník. V druhém případě byla data pouze videa s krátkým popiskem.

Hodnocení výsledků shluků se určuje pomocí metody precision, recall a F-measure (přesnost, pokrytí a F-míra. Přesnost a pokrytí se hodnotí mezi ručně vytvořenými shluky a výslednými shluky novinového agregátoru.

$$precision = \frac{S}{O},$$

kde S je počet dokumentů, které jsou zároveň v clusteru vytvořeném aplikací a v clusteru vytvořeném člověkem a O počet všech dokumentů v clusteru vytvořeném aplikací.

$$recall = \frac{S}{N},$$

kde N počet všech dokumentů v clusteru vytvořeném člověkem. Výsledná F-míra se poté vypočítá pomocí vzorce (více v [11]):

$$Fmeasure = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Výsledná hodnota je poté průměrem hodnot F-míry všech souborů ve shlucích. Výsledná hodnota shodnosti při testování programu byla 96,85% F-míry. Testovací data obsahovala 1162 dokumentů a bylo v nich 129 clusterů.

6 Závěr

Tato bakalářská práce se zabývala agregátory novinových zpráv. Hlavním cílem bylo vytvořit nový novinový agregátor, který používá vhodné metody.

Prvním cílem bylo prozkoumání technologií použitelných pro agregátor novinových zpráv. Splnění tohoto cíle spočívalo v analýze metod pro předzpracování dat, druhy příznaků v datech, určení podobnosti a metod shlukování dat. Zpracování cíle mi přinesl nový rozhled v těchto oblastech.

Druhým z těchto cílů bylo navržení agregátoru novinových zpráv, ale také návrh systému pro stahování testovacích dat z internetových zdrojů a jejich formát. Při zpracování tohoto cíle byl vybrán formát souboru s novinkami, dále metody předzpracování dat jako je tokenizace a stemming. Byl vybrán příznak TF-IDF jako vhodný pro obsah dat v novinkách a také metoda shlukování.

Do třetího cíle spadala implementace nového systému pro novinový agregátor. Pro implementaci byly využity různé knihovny, technologie a projekty, které byly vhodné. Bylo také vytvořeno grafické uživatelské rozhraní pro jednodušší a přehlednější ovládání.

Posledním cílem bylo testování výsledků a jejich zhodnocení. Testování správného výpočtu příznaku TF-IDF je provedeno pomocí automatických JUnit testů na vlastních datech. Při testování shluků bylo dosaženo shodnosti 96,85% F- míry s testovacími daty.

Po splnění všech dílčích cílů je také splněn hlavní cíl. Výstupem práce je nový systém pro agregátor novinových zpráv. Systém je možné dále rozšířit o možné příznaky, jako je délka dokumentu nebo jména osob a organizací a také o možnosti uživatelského rozhraní.

Seznam použitých zkratek

ATOM – formát dat podobný RSS

CSS – kaskádový styl pro vzhled stránek

GUID – identifikátor novinových zpráv

RSS – datový formát pro sdílení a publikování obsahu internetových stránek

SVG – vektorový grafický formát dat

TF-IDF – metoda pro určování váhy slova

URL – určuje adresu serveru a protokol, přes který lze přistoupit

HTML, XHTML – značkovací programovací jazyk pro tvorbu webových stránek

XML – strukturovaný rozšiřitelný značkovací jazyk

WWW – webová stránka na internetu

Literatura

[1] *RSS 0.91 Specification (UserLand)*. RSS Advisory Board [online]. 2000 [cit. 2013-12-07]. Dostupné z: `http://www.rssboard.org/rss-0-9-1#textInput`

[2] *RSS: Technologie a online čtečky*. Metodický portál RVP [online]. 2013 [cit. 2013-12-07]. Dostupné z: `http://clanky.rvp.cz/clanek/o/g/17325/RSS-TECHNOLOGIE-A-ONLINE-CTECKY.html/`

[3] *Jak se neztratit v záplavě zpráv a informací*. Živě.cz - O počítačích, IT a internetu [online]. 2006 [cit. 2013-12-08]. Dostupné z: `http://www.zive.cz/clanky/jak-se-neztratit-v-zaplave-zprav-a-informaci/sc-3-a-129715/default.aspx`

[4] *BoilerPlate Removal and Fulltext Extraction from HTML pages*. Google Code [online]. 2011 [cit. 2013-12-17]. Dostupné z: `https://code.google.com/p/boilerpipe/`

[5] MARTINEC, Petr. *Detekce témat článků* [online]. Praha, 2013 [cit. 2014-04-15]. Dostupné z: `https://dspace.cvut.cz/handle/10467/16086`. Diplomová práce. České vysoké učení technické v Praze.

[6] SUNG-HYUK, Cha. *Comprehensive survey on distance/similarity measures between probability density functions*. International Journal of Mathematical Models and Methods in Applied Sciences, 1(4):300–307, 2007.

[7] KONOPÍK, Miloslav a BRYCHCÍN, Tomáš. *HPS: High Precision Stemmer* [online]. 2013 [cit. 2014-04-15]. Dostupné z: `http://liks.fav.zcu.cz/HPS/`

[8] *Data Mining Algorithms In R/Clustering/Hybrid Hierarchical Clustering*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2014-04-29]. Do-

stupné z: http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/Hybrid_Hierarchical_Clustering

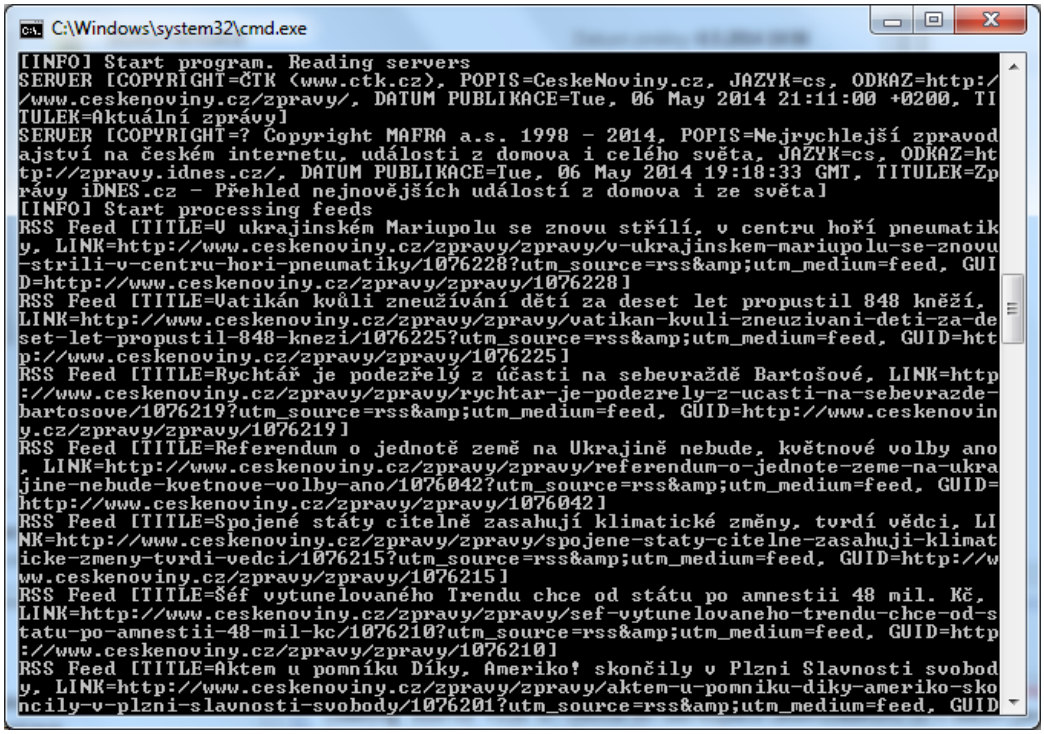
[9] RYCHTAŘÍKOVÁ, Lenka. *Shluková analýza* [online]. 2012 [cit. 2014-04-29]. Diplomová práce. Masarykova univerzita, Přírodovědecká fakulta. Vedoucí práce Marie Budíková. Dostupné z: http://is.muni.cz/th/175677/prif_m/

[10] KUČERA, Jiří. *Metody kategorizace dat* [online]. 2008 [cit. 2014-04-29]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Matěj Štefaník. Dostupné z: http://is.muni.cz/th/172767/fi_b/

[11] MELAMED, I. DAN a GREEN, Ryan a Turian, JOSEPH, P. *Precision and Recall of Machine Translation.* [online]. 2013 [cit. 2014-05-07]Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2. [] Dostupné z: http://dx.doi.org/10.3115/1073483.1073504.

A Uživatelská dokumentace sběru RSS dat

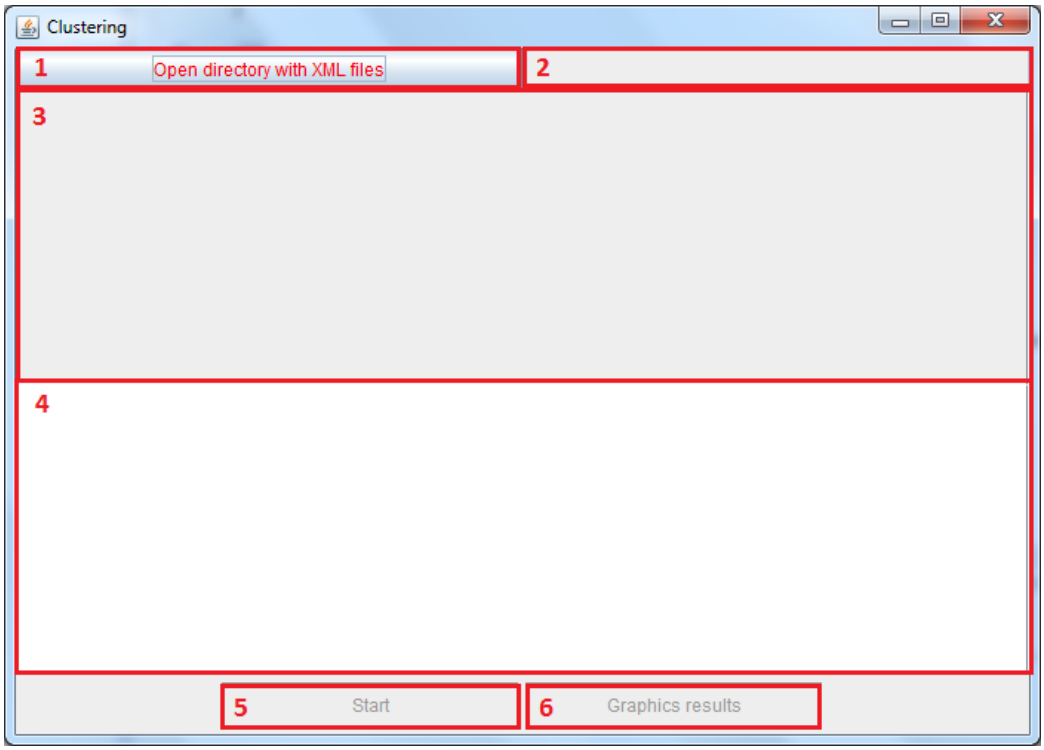
Program spustíme pomocí dávkového souboru *sber_rss_dat.bat* na disku v kořenovém adresáři, pokud máme operační systém Windows. Máme-li operační systém Linux, spustíme pomocí shellového skriptu *sber_rss_dat.sh*. Jako konfigurační soubor s odkazy na RSS kanál je implicitně nastaven soubor *RSS.txt* a složka pro uložení novinových zpráv je *RSS*. Obojí najdeme ve složce *sber_rss_dat*. Program je konzolová aplikace (viz Obrázek A.1).



Obrázek A.1: Konzolová aplikace sběru RSS dat

B Uživatelská dokumentace novinového agregátoru

Program spustíme pomocí dávkového souboru *zpracovani.bat* na disku v kořenovém adresáři, pokud máme operační systém Windows. Máme-li operační systém Linux, spustíme pomocí shellového skriptu *zpracovani.sh*. Po spuštění se objeví hlavní okno (Obrázek B.1).

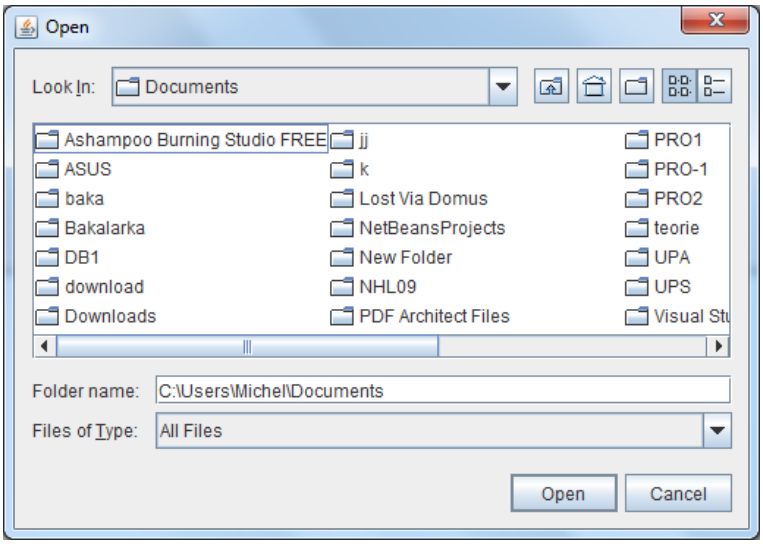


Obrázek B.1: Základní rozvržení agregátoru

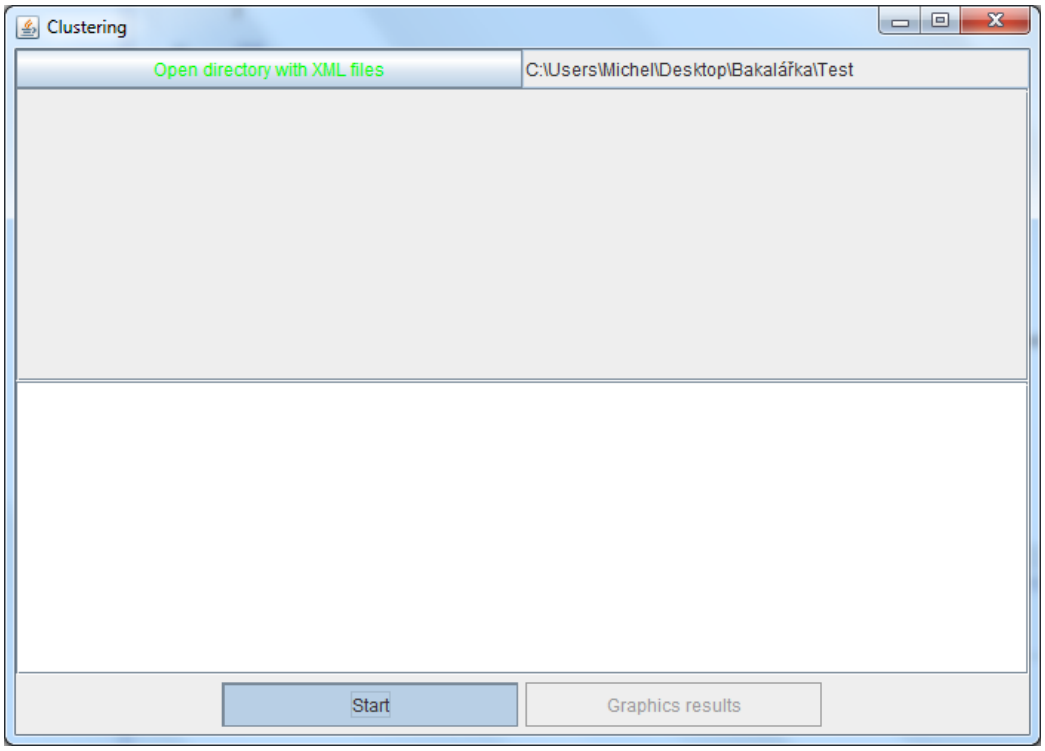
1. Tlačítko pro otevření složky s novinkami
2. Pole obsahující vybranou složku
3. Výsledky programu
4. Textové pole pro informace o stavu a běhu programu
5. Tlačítko pro spuštění clusterování dat

6. Tlačítko pro grafický výstup programu

První věcí, kterou musíme provést je výběr složky, ve které máme data pro novinový agregátor. Složku vybereme kliknutím na tlačítko „**Open directory with XML files**“. Po kliknutí na tlačítko se objeví dialogové okno a vybereme danou složku (viz Obrázek B.2). Pokud vybereme správnou složku, tlačítko změní barvu na zelenou (viz Obrázek B.3).

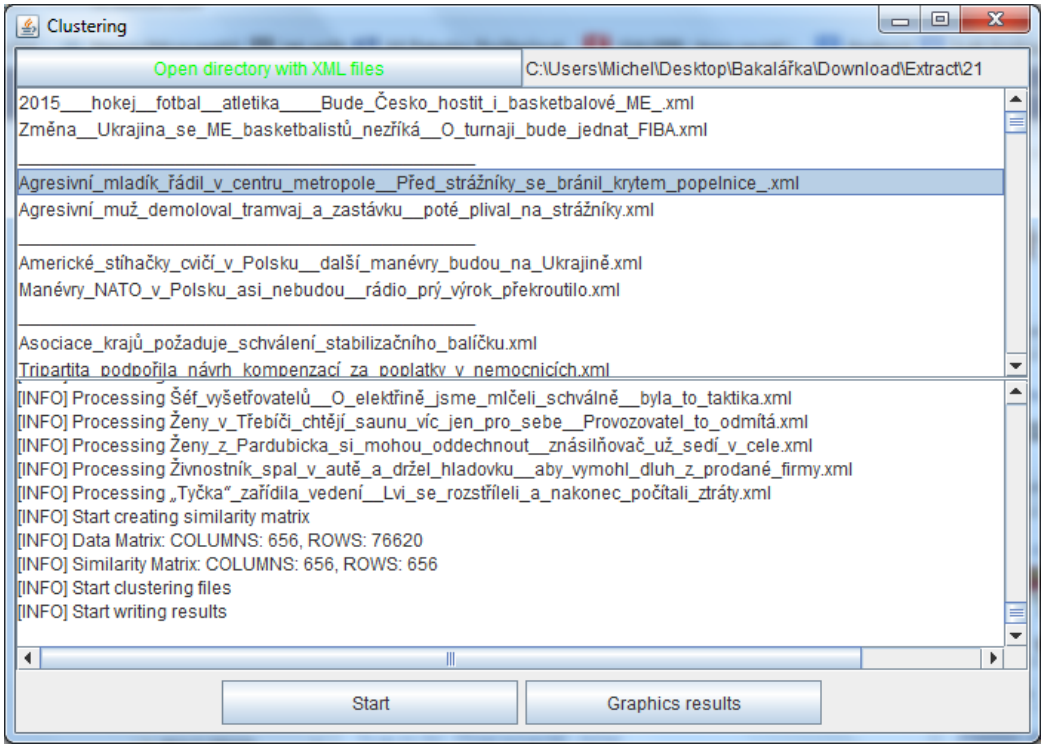


Obrázek B.2: Dialogové okno pro výběr složky s XML soubory

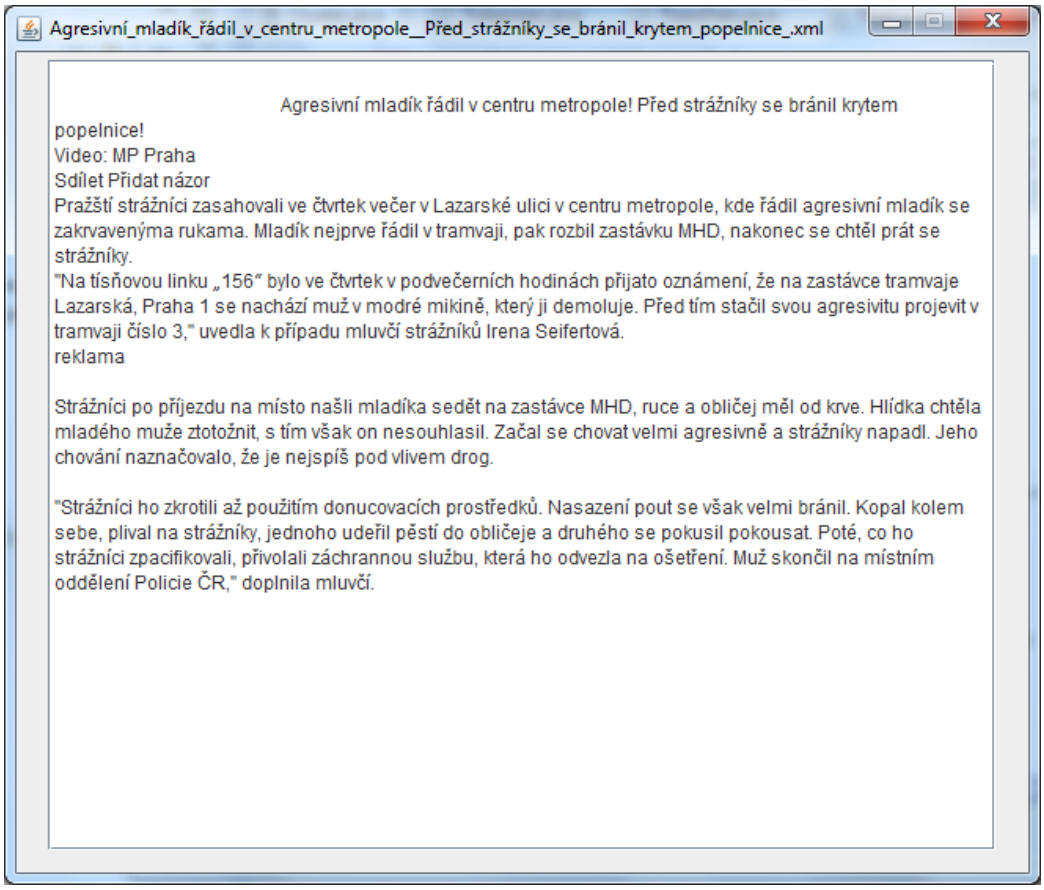


Obrázek B.3: Změna vzhledu po vybrání správné složky

Pro běh programu stačí znát složku se soubory, můžeme tedy program spustit tlačítkem „**Start**“. Výpisy programu se vypisují do logovacího okna. Výsledky se zapíší do výsledkového okna (viz Obrázek B.4). Dvojitým kliknutím na název zprávy ve výsledcích otevřeme popis zprávy (viz Obrázek B.5)

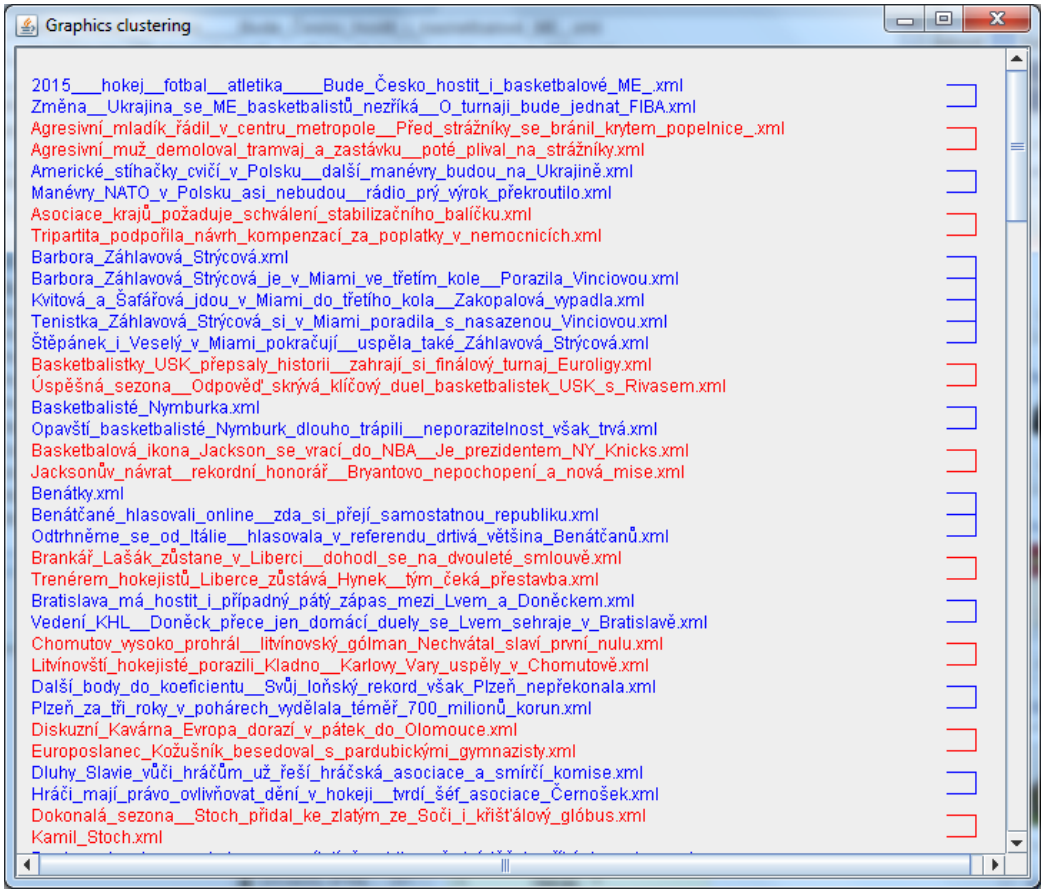


Obrázek B.4: Vzhled agregátoru po clusterování



Obrázek B.5: Popis souboru

Kliknutím na tlačítko „**Graphical results**“ otevřeme nové okno s grafickými výsledky (viz Obrázek B.6). Soubory v jednom clusteru jsou spojeny čárami mezi názvy.



Obrázek B.6: Grafické výsledky

C Obsah CD

sber_rss_dat – soubory potřebné ke sběru RSS dat

- **src** – zdrojové soubory
- **target** – zkompilované soubory, testy, spustitelné JAR soubory
  - **classes** – zkompilované soubory
  - **test-classes** – testovací soubory a výsledky
  - **SBER_RSS_DAT-1.0.jar** – spustitelný JAR soubor
- **RSS** – imlicitní složka pro ukládání novinových zpráv
- **pom.xml** – soubor s informacemi o projektu a překladu
- **RSS.txt** – konfigurační soubor s odkazy na RSS kanály

zpracovani – soubory potřebné pro agregátor

- **src** – zdrojové soubory
- **target** – zkompilované soubory, testy, spustitelné JAR soubory
  - **classes** – zkompilované soubory
  - **test-classes** – testovací soubory a výsledky
  - **ZPRACOVANI-1.0.jar** – spustitelný JAR soubor
- **config.xml** – nastavení programu
- **cz.bin** – binární soubor českých slov pro stemmer
- **pom.xml** – soubor s informacemi o projektu a překladu
- **train.bin** – binární soubor s natrénovanými daty

dokumentace – složka obsahující dokumentace k bakalářské práci

- **javadoc_sber_rss_dat** – javadoc dokumentace ke sběru RSS dat
  - **index.html** – zobrazení dokumentace v internetovém prohlížeči
- **javadoc_zpracovani** – javadoc dokumentace ke zpracování dat
  - **index.html** – zobrazení dokumentace v internetovém prohlížeči
- **BP_Sobehart_agregator.pdf** – dokumentace k bakalářské práci

README.txt – soubor se základními informacemi o programu

sber_rss_dat.bat – spuštení programu pro sběr dat na systému Windows

sber_rss_dat.sh – spuštení programu pro sběr dat na systému Linux

zpracovani.bat – spuštení programu pro zpracování na systému Windows

zpracovani.sh – spuštení programu pro zpracování na systému Linux