VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

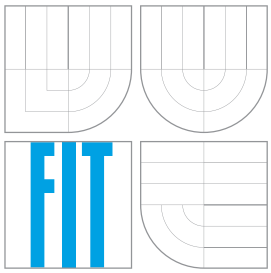
ADAPTIVNÍ RSS ČTEČKA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JINDŘICH LUŽA
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ADAPTIVNÍ RSS ČTEČKA

ADAPTIV RSS READER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JINDŘICH LUŽA
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. SMRŽ PAVEL, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

Práce se zabývá možností obohacením běžné RSS čtečky o rozšíření umožnující uživateli jednodušeji filtrovat RSS záznamy na základě jejich zařazení do skupin podle obsahu jejich textu. Jsou zde probrány problémy které vznikají při obecné klasifikaci a při klasifikaci textů. Dále je zde poukázáno na nutné teoretické aspekty formátu RSS, které je potřeba zvažovat při implementaci modulu RSS čtečky a možná podoba návrhnu modulu. Jako poslední je zde uvedeno testování vhodnosti použité klasifikátoru.

Abstract

Purpose of this balcheor thesis is posibility to enhance common RSS reader by extension, which allowing user filter RSS feed depends on that's classification by content to groups. There is discussed problems in common classification and in text classification. Forth, there is reveal teoretical aspect of RSS format, which is needed to be considered in implementation of RSS reader module and prototype of module. At last, testing of used classifier is stated here.

Klíčová slova

RSS čtečka, Adaptivní RSS čtečka, Klasifikace textu

Keywords

RSS reader, adaptive RSS reader, text classification

Citace

Jindřich Luža: Adaptivní RSS čtečka, bakalářská práce, Brno, FIT VUT v Brně, 2011

Adaptivní RSS čtečka

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže Ph.d.

.....
Jindřich Luža
18. května 2011

Obsah

Seznam obrázků

Úvod

„Čas je mnohem cennější než peníze. Peněz můžete získat více, ale času ne“
Jim Rohn

Trend moderní doby je rychlost, jednoduchost a efektivita. Tomuto trendu podléhají bezmála všechny oblasti lidského sociálního bytí. A právě tento trend dal podmět ke vzniku technologie RSS, která umožňuje uživatelům internetu odebírat nově publikované články ze svých oblíbených portálů, bez nutnosti otevřít webový prohlížeč a manuálně procházet každý server zvlášť. Z důvodů masivně se rozšiřující reklamy, datové náročnosti obsahu stránek a ne vždy dostačující technické kapacity servru pro rychlé zpracování požadavku, je úspora času při použití technologie RSS citelná.

Bohužel masivní rozvoj internetu, neustále jednodužší přístupnost a možnost snadno se podílet na utváření jeho obsahu, zapříčiňuje zahlcení internetu nepřesnými nebo nepříliš kvalitními informacemi mnohdy doprovazenými notnou dávkou reklam. Vrcholem zamořování internetu reklamou jsou webové plagiáty vylepšené spoustou reklamních ploch, stránky s nulovou informativní hodnotou zaměřující se z převážné většiny na propagaci reklamny a vyhledavací systémy využívající nekalé SEO praktiky, které směřují uživatele na jejich stránky podle obsahu, který zadal do vyhledávače. Je tak stále složitější a časově náročnější dopracovat se na internetu k informaci, kterou uživatel opravdu potřebuje. Z tohoto důvodu zavadí většina vyhledávačů a katalogů hodnotící algormitny snažící se eliminovat nekvalitní stránky od kvalitních a zkrátit uživateli čas k nalezení požadovaného výsledku.

Cílem této práce prozkoumat možnost vytvoření RSS čtečky disponující podobným algoritmem, usnaďnujícím uživateli výběr mezi články dostupnými přes RSS zdroje. Adaptivnost RSS čtečky spočívá v kategorizování článků podle jejich obsahu. Uživateli pak bude stačit zadat pouze klíčová slova nebo vybrat kategorii z niž chce články zobrazit. Pro efektivní zařazení do kategorií bude potřeba použít nějaký z algoritmů pro klasifikaci textů. Budou zde zmíněny teoretické možnosti vybraných algoritmů a jejich vhodnost pro potřeby implementovaného algoritmu. Z praktického hlediska zde bude nahlíženo na běžné problémy, které se mohou vyskytnout při nesprávném používání čtečky, např. nevalidita webové stránky nebo příliš krátký obsah článku. Jako poslední část práce bude uvedeno testování použitých metod.

Kapitola 1

Teoretická část

1.1 Webová syndikace

Internet je z převážné většiny tvořen aktivním obsahem, tj. nejnovějšími zprávami, hudbou, filmy. Tvorba obsahu je ale finančně nákladná záležitost a manuální vyhledávání aktuálního obsahu je časově nákladné. Pokud je tvůrce obsahu placen jedním vydavatelem zůstává často nedoceněn, protože vydavatel si nemůže dovolit mu zaplatit částku odpovídající jeho úsilí, kvůli tomu, že většinou platí více než jednoho autora. Celkově se to projeví u uživatele, který požaduje kvalitní obsah, ale nedostává ho. požaduje kvalitní obsah, ale nedostává ho.

Řešení tohoto problému přináší webová syndikace. Je to obchodní struktůra spočívající v distribuci obsahu mezi více stranami, kdy autor poskytuje svůj obsah více vydavatelům za menší poplatek. Vydavatel poskytuje kvalitní obsah čtenáři za členský příspěvěk nebo za příspěvek formou reklamy. Čtenář již nevyhledává obsah manuálně na webu, ale získává informace o nejnovějším obsahu přes informační kanál, kterým může být například RSS nebo Atom.[6]

1.1.1 Jazyk XML

XML (eXtebsible Markup Language) je značkovací jazyk vyvinutý skupinou XML Working Group pod zášťitou konsorcia W3C. Jazyk SGML, který je jeho předchůdcem, je velmi obecný a poskytyje mnoho možností k popisu dat. Právě komplexnost jazyka SGML vedla ke zniku jazyka XML. XML je otevřený formát, který má jednoduchou syntaxi a není aplikačně ani platformě závislý. To umožnilo jeho rychlé rozšíření po celém světě. Striktní syntaxe XML umožnuje levnější a rychlejší vývoj aplikací, které ho používají[7]. XML není omezen na použití konkrétním jazykem a umožnuje uchovávat strukturu dat, což z něho dělá jazyk vhodný k přenášení dokumentů i konfiguračních souborů či souborů popisující databázi.

1.1.2 Formát RSS

Před vznikem formátu RSS existovala celá řada formátů pro syndikaci obsahu, jmenovitě například formát CDF od Microsoftu nebo Scripting News, který je založen na značkovacím jazyku XML. První verzi formátu RSS vyvinul Dan Libby ze společnosti NetScape roku 1999. Tato verze byla označována jako verze 0.9. Téhož roku byla vydána verze 0.91, která převzala některé části jazyka Scripting News založeného na XML. Když společnost NetScape vývoj jazyka RSS opustila, její místo převzala skupina RSS-DEV. Současně na vývoji

jazyka RSS pracoval Dave Winer, tvůrce jazyka Scripting News. Kvůli nejednotnému vývoji v tuto dobu vznikají dvě nezávislé verze jazyka RSS. Verze RSS 1.0 vydaná skupinou RSS-DEV a RSS 0.92 vyvinutá Winerem. RSS 1.0 byla od přechozích verzí RSS syntakticky velmi odlišná a pro mnoho lidí obtížná k používání. Winer následně vyvinul minoritní verze 0.93 a 0.94 a v roce 2002 představil RSS v 2.0 [9].

1.1.3 Popis formáru RSS

Pro ozřejmění možností poskytnutých služeb webové syndikace obsahu zde bude popsán formát RSS v 2.0. Následující informace jsou čerpány z [9].

Jak již bylo zmíněno, formát RSS je formátem založeným na jazyku XML. Podle specifikace definuje dvojí typ informací, které bude kanál obsahovat: povinné informace, které musí každý rss zdroj obsahovat, aby byla zaručena jeho základní funčknost a volitelné informace.

Soubor RSS zdroje začíná jako každý XML soubor hlavičkou, která obsahuje informace a verzi XML a případně volitelně o použitém kodování. Samotný formát RSS začíná elementem *rss*, který obaluje veškerý obsah týkající se kanálu. Element musí obsahovat atribut *version*, který udává informace o verzi RSS. Element *rss* má podelement *channel*, který již obsahuje informace o rss kanálu.

Poviné prvky elementu *channel* jsou následující:

title – element obsahující titulek kanálu

link – element obsahující odkaz na html stránku, ze které kanál poskytuje informace

description – element obsahující textový popis obsahu kanálu.

Volitelné prvky podávající rozšiřující informace o rss kanálu jsou pak tyto:

language – element udavající přirozený jazyk kanálu

copyright – informace o autorských právech o zdroji

managingEditor – email na editora odpovědného za obsah kanálu

webMaster – email na osobu odpovědnou za techniký provoz kanálu

pubDate – datum publikace obsahu kanálu, tozn. datum kdy byl zveřejněn aktuální obsah kanálu.

lastBuildDate – datum poslední změny kanálu.

category – jedna nebo více kategorií, do kterých kanál patří

generator – název programu, který RSS kanál vygeneroval

docs – odkaz na dokumentaci verze RSS formátu, použitého pro daný rss zdroj.

cloud – u RSS kanálů se předpokládalo, že budou aktualizovány nejdříve jednou za hodinu. Pro potřeby rychlejší aktualizace byl do verze 2.0 zaveden element *cloud* obsahující informace o webové službě podporující rozhrají *rssCloud* a umožnit tak klientské aplikaci se připojit k této službě. Element obsahuje atributy domain, port, path, registerProcedure, protocol udavající parametry webové služby.

ttl – udává dobu platnosti aktuálnosti kanálu. V kombinaci s elementy pubDate nebo lastBuildDate lze určit, zda je potřeba zaslat na server požadavek na aktuální verzi kanálu nebo nikoliv.

image – udává informaci o obrázku vztahujícímu se ke kanálu. Má tři povinné podelementy:

title – obsahuje titulek obrázku

url – odkaz na url obsahující soubor s obrázkem.

link – odkaz na HTML stránky nesoucí informace o rss kanálu, po kliknutí na obrázek

a tři volitelné atributy:

width – šířka obrázku

height – výška obrázku

description – popis obrázku

rating – obsahuje hodnocení RSS kanálu ve formátu PICS

textInput – umožnuje čtenáři jistou interakci s kanálem formou textového pole s odesílacím tlačíkem, což může sloužit například k vyhledávání na webových stránkách kanálu nebo odeslání reakcí a připomínek. Obsahuje čtyři povinné podelementy:

title – popisek odesílacího tlačítka

description – popisek textového pole

name – jméno objektu textového pole

link – odkaz na skript zpracující odeslání textu

skipHours – prvek sdělující čtečce, ve které hodiny nemusí kontrolovat aktuálnost RSS kanálu(například proto, že v noci se kanál neaktualizuje). Tento element obsahuje podelementy *hour*, které obsahují hodnoty 0-23 značící dannou hodinu v časovém pásmu GMT.

skipDays – prvek sdělující čtečce, ve které dny nemusí kontrolovat aktuálnost rss zdroje. Obsahuje podelementy *day*, které mohou nabývat hodnot anglických pojmenování dnů v týdnu: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

Element *channel* dále obsahuje podelement *items*, který poskytuje informace o jednotlivých příspěvcích kanálu. Informace o příspěvku jsou zahrnuty v elementu *item*, který obsahuje elementy:

title – Název příspěvku

link – Odkaz na webovou stránku obsahující příspěvek

description – popis příspěvku

autor – e-mailová adresa na autora článku

category – jedna nebo více kategorií do niž článek patří

comments – odkaz na stránku s komentáři vztahujícími se k článku

enclosure – element určující obsah multimediálního typu přidružený k příspěvku. Element má tři povinné atributy

url – odkaz na soubor obsahující multimediální data.

length – velikost odkazovaného souboru v bytech

type – určuje typ multimediálního obsahu ve formátu MIME

guid – umožnuje přiřadit článku jednoznačný identifikátor. Toho lze využít při rozlišení dvou různých příspěvků v RSS se stejným nadpisem.

pubDate – datum kdy byl příspěvek publikován

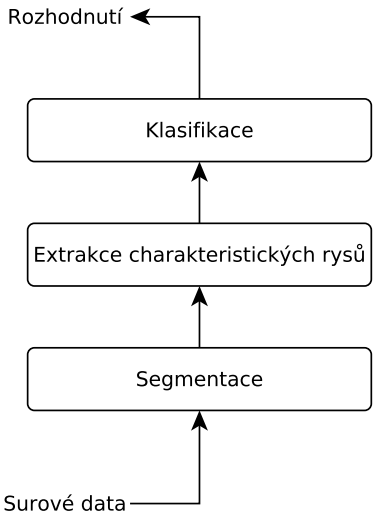
source – RSS kanál odkud příspěvek pochází

1.2 Obecná klasifikace

S potřebou rozdělit objekty do určitých skupin se setkáváme každoděnně ve svém životě. Mnohdy je schopnost rychle a správně přiřadit subjekt do určité kategorie nutnost potřebná k přežití. Tísíce let evoluce nás naučily rozpoznávat a přiřazovat věci do kategorií na základě předchozích zkušeností. Pro zjednodušený příklad jak funguje lidská klasifikace, uvažujme nalezení neznámé houby. Na základě předchozích zkušeností víme, že pozření houby, která má klobouček v červené barvě, způsobuje nevolnost. Dále víme, že houba která má štiplavou chuť je pravděpodobně jedovatá. Pro zjištění zda je houba jedlá nebo pohledem zjistíme barvu jejího klouboučku a ochutnáním malého kousku zjistíme chuť. Pokud bychom uvažovali chuť v hodnotách štiplavá,neštiplavá a barvu v hodnotách odstín červené,odstín žluté (který je tvořen jak červenou tak zelenou barvou) a odstín zelené, dostali bychom celkem šest možných kombinací hub. Po zjištění barvy a chuťi se podle nějakého kritéria rozhodneme zda houbu sníme nebo ne. Jako kritérium bychom v tomto případě mohli brát zvědavost, odvážnost nebo pud sebezáchovy. Celý proces přiřazení houby do kategorie představuje obrovské množství operací vykonaných naším mozkem.[8]

Klasifikace subjektu strojem pracuje na velmi podobném principu. Z pohledu stroje se ovšem nejedná o triviální záležitost. Stroj chápe surová data jako nuly a jedničky nebo na ještě větší specializaci jako velikosti elektrického napětí. Proto je potřeba udělit datům nejprve nějakou sémantiku. V praxi často potřebujeme vyčlenit ze shluku dat jednotlivé elementy. Tento proces se nazyva segmentace. Segmentace komplexních zdrojů dat jako například mluvená řeč nebo video může činit značný problém, protože jednotlivé elementy se například mohou překrývat. Po segmentaci a semantice získáme jednotlivé objekty a jejich vlastnosti získané z dat. Při velkém počtu získaných informací o objektu by byla klasifikace objektu podle všech těchto vlastností časově i pamětově náročná a neefektivní. Sousta vlastností má pro potřebu klasifikace nulovou informativní hodnotu a při jejich zahrnutí do porovnávaných vlastnotí při klasifikaci nedosáhneme žádného zlepšení přesnosti. Kvůli výše zmiňovaným důvodům abstraujeme od komplexního popisu objektu k zjednodušenému modelu. Proces abstrakce se nazývá extrakce charakteristických rysů (feature extraction) a spočívá v odstranění vlastností s nulovou a velmi malou informační hodnotou a vybrání těch vlastností, které jsou charakteristické pro danou kategorii do které objekt patří. Tato operace přináší možnost zjednodušení výsledného hodnotícího systému. Systém nebude muset znát všechny možné kombinace a vlastnosti klasifikovaných objektů, nicméně nebude schopný dosahovat stoprocentní přesnosti kvůli zanedbání některých vlastností objektu. Při výběru klíčových charakteristických rysů objektu musíme počítat s možností, že

ne všechny rysy objektu, námi zvolené jako stěžejní rysy pro klasifikaci, budou u každého objektu dostupné. Tato možnost nastává zejména při špatně získaných surových datech o objektu, například v důsledku šumu nebo překrývajících se objektů. Průběh klasifikace strojem je znázorněn na obrázku 1.1



Obrázek 1.1: Průběh klasifikace strojem

1.3 Klasifikace textu

Při klasifikaci textu je surovým datům udělena sémantika posloupnosti znaků tvořící slova,které dále tvoří věty. Segmentace v tomto případě odpadá, protože obvykle již máme vstupní data získaná po jednotlivých segmentech. Klasifikovaný text se pro dosažení lepších výsledků klasifikace předzpracovává. Mezi nejčastejší úpravy patří lematizace, stemming a odstranění stop-slov.

1.3.1 Stemming

V běžném textu se slova vyskytují v různých tvarech. Tato skutečnost vnáší složitost do algoritmů zabývající se zpracováním textu na informace, mezi které patří i klasifikace textu. Algoritmy získávající informace z textu se obvykle rozhodují hlavně na základě počtu výskytů a hustotě výskytu slova nebo slovního spojení v textu[4]. Při výskytu různých tvarů téhož slova v textu, získá algoritmus nesprávné informace. Jedním ze způdobů jak docílit přesnějšího zpracování textu je stemming, který extrauje základ slova. Vezmeme-li v úvahu počet existujících přirozených jazyků a nejednoduché gramatické pravidla, vážící se ke každému znich, dojdeme k závěru,že proces pro získání základu slova není jednoduchou a už vůbec ne obecnou záležitostí. Pro stemming textu existuje značné množství metod, například: odtržení koncovek, zarovnání slov, segmentace slov aj. Mezi nejznámější stemmery patří Porterův stemmer a Lovinsův stemmer, pracující na principu odtržení koncovek.[4]

1.3.2 Lematizace

Lematizace je dalším způsobem jak docílit transformace slova do základního tvaru. Narozdíl od stemmingu, který produkuje základ slova, lematizace získává základní tvar slova pomocí morfologický pravidel daného jazyka.

1.3.3 Stop-slova

Stop slova jsou slova v textu, které nemají pro klasifikaci žádnou informativní hodnotu a tudíž je vhodné je z textu odstranit.Patří mezi ně například spojky, přeložky a zájmena.

1.3.4 Extrakce klíčových slov

Pro zjednodušení klasifikátoru provádíme extrakci charakteristických rysů–slov v případě klasifikace textu. Jednak tím snížime velikost vektorové formy dokumentu a také zvýšíme přesnost klasifikace odstraněním nežádoucích slov[3]. K výběru klíčových slov existuje několik přístupů.

Výběr nejfrekventovanějších výrazu

První přístup je vybrání nejfrekventovanějších slov. Frekventovanost výrazu můžeme brát jako počet výskytů slova v zpracovávaném textu nebo jako počet počet textů v určité třídě, obsahující daný výraz. I přes svou jednoduchost metoda výběru nejfrekventovanějších slov často vybírá výrazy, které nemají souvislost s danou kategorií, například dny v týdnu při klasifikaci aktuálních zpráv.

Výběr výrazů s největší vzájemnou informací

Dalším přístupem je metoda, kdy se se určuje, který výraz bude mít pro klasifikaci největší přínos na základě vzájemné informace výrazu t a třídy c. Vzájemná informace se spočítá jako:

$$I(U;C) = \sum_{e_t \in \{0;1\}} \sum_{e_c \in \{0;1\}} P(U = e_t; C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(U = e_c)}$$

kde U je náhodná proměná v rozsahu $\langle 0; 1\rangle$, přičemž 1 znamená, že termín se v dokumentu vyskytuje a 0, že se v dokumentu nevyskytuje. C je náhodná proměnná v rozsahu $\langle 0; 1\rangle$, kde 1 znamená,že dokument je ve třídě C a 0, že dokument ve třídě C není.
Výraz lze rozepsat jako

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_1 N_1} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_0 N_1} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_1 N_0} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_0 N_0}$$

kde N je počet dokumentů, které zastoupených výrazem t a třídou c, přičemž první index značí přítomnost nebo absenci výrazu v dokumentu a druhý index značí přítomnost nebo absenci dokumentu ve třídě. Pro extrakci nejlepších klíčových slov pak vybereme k termínů s největší hodnotou.

χ^2 metoda

Metoda χ^2 spočívá v zjištění nezávislosti jevů A a B podle pravidla $P(A|B) = P(A)P(B)$, kde A je v našem případě výskyt výrazu v dokumentu a B výskyt dokumentu ve třídě. Spočítání míry nezávislosti je podobné jako vzorec pro spočítání míry vzájemné informace:

$$\chi^2(D, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_t \in \{0,1\}} \frac{(N_{e_t} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

kde N je skutečný počet výskytů pro dokument D a E je očekávaný počet. Očekávaný počet výskytů spočítáme jako:

$$E_{e_t e_c} = N \cdot P(e_t) \cdot P(e_c)$$

což by pro E_{11} znamenalo počet dokumentů ve kterém se vyskytuje termín t a dokument zároveň patří do třídy c, za předpokladu že t a c jsou na sobě nezávislé. Jako klíčové výrazy vybereme ty, které mají nejmenší hodnotu χ^2

1.3.5 Vektorová reprezentace textu

Abychom mohli text klasifikovat je potřeba ho převést do matematického tvaru s nímž se dá počítat. To provedeme vyjádřením textu jako vektoru. Protože po extrakci klíčových výrazů je text reprezentován jako sekvence těchto výrazů, nejjednodužší vektorovou reprezentací této sekvence je vektor, vyjadřující přítomnost nebo nepřítomnost daného výrazu v textu. Protože počet klíčových výrazů vyskytujících se skrz všechny dokumenty je velký a v jednom textu bývá zastoupena pouze minoritní část těchto výrazů, často se texty reprezentují jako řídké vektory, kdy bereme v uvahu pouze výrazy, které se v daném textu vyskytují. Přístup, kdy výraz ve vektoru zastoupíme pouze hodnotou 1(výraz se v dokumentu vyskytuje) nebo 0(výraz se v dokumentu nevyskytuje) nazýváme *boolovské vážení*. Toto vyjádření přítomnosti a nebo nepřítomnosti výrazu v textu ovšem vylučuje možnost zohlednění vícenásobného výskytu výrazu v textu. Metodou jak vzít u uvahu i vícenásobné vyskyty výrazu v textu je vážení podle frekvence výrazů v dokumentu, inverzní vážení podle frekvence dokumentů obsahující výraz, a kombinace předchozích dvou přístupů.[12]

Vážení frekvence podle výrazů spočívá v přirazení každému výrazu hodnoty, která reprezentuje počet jeho výskytů v textu. Nicméně tato metoda trpí základním nedostatkem a to sice tím, že všechny výrazy jsou považovány za rovnocenné, ale ve skutečnosti mají některé výrazy malou vypovídací hodnotu o tom do které kategorie text patří. Způsobem jak eliminovat příliš velký vlivu výrazů, které se v dokumentu vyskytují příliš často a mají malou vypovídací hodnotu je snížení váhy výrazu na základě faktoru, který roste s jejich vzrůstajícím počtem.

Na základě toho definujeme inverzní vážení podle počtu textů obsahující výraz (inverse documment frequency) jako:

$$idf_t = log \frac{N}{df_t}$$

kde df_t je počet textů obsahující výraz t a N je celkový počet textů v trénovací sadě.[3]

Zatímco vážení podle frekvence výrazů předpokládá, že výraz,který se vyskytuje v textu častěji bude mít na klasifikaci větší vliv, inverzní vážení podle frekvence textů přikládá větší hodnotu výrazu vyskytujícímu se spíše méně často. Kombinací těchto dvou krajních metod

dostaneme metodu použitelnou pro většinu dokumentů. Nejjednodužší kombinace je prostý součin dvou výše zmíněných vzorců.

$$tf - idf_{t,f} = tf_{t,f} \times idf_t$$

Při zpracovaní textů rozdílné délky může ještě vzniknout potřeba omezit váhový koeficient kvůli znevýhodnění textů s menší délkou, které budou mít zákonitě většinou méně klíčových výrazů než dokumenty větší délky. To lze provést normalizací vektorové reprezentace dokumentu na jednotkovou velikost nebo použit sofistikovanější metodu při počítání výsledného hodnocení klasifikace. [3, 10]

1.3.6 N-Grams

Jako klíčové výrazy a vůbec základní prvky textu se ve většině případů klasifikace textu berou slova. Alternativní volba základního prvku vyskytujícím se ve vektorové podobě textu spočívá v použití N-gramu. N-gram je sekvence N po sobě jdoucích položek, které mohou být slova nebo znaky. Použití N-gramů skládajících se ze znaků oproti celému slovu přináší do klasifikace řadu výhod. Při poškození textu se chyba neprojeví na celém slovu ale jenom na N-gramech, které budou odpovídat danné části slova. Výhodné je použití N-gramů na jazyky, pro které neexistuje kvalitní stemmer, nebo jeho použití z technických důvodů není možné [5, 11]. Zajímavá metoda využití N-gramů je popsána v [11], kdy jsou texty klasifikovány na základě podobnosti N-gramových profilů spočítaných na základě četností jednotlivých N-gramů v dokumentu pro dvou až k znakové N-gramy.

1.4 Klasifikační algoritmy

1.4.1 Algoritmy na bázi učení s učitelem

Algoritmy fungující na principu učení s učitelem ke svému fungování potřebují trénovací sadu, která obsahuje texty o kterých víme do které kategorie patří. Po natrenování je algoritmus schopný určit, do které z kategorií s největší pravděpodobností nový neznámý text patří.

K Nearest Neighbour

Algoritmus klasifikace K Nearest Neighbour spočívá v nalezení k vektorově nejbližších sousedů testovaného textu v trénovací sadě. Na základě kategorií nalezených sousedů testovaného textu se pak vyčítává kategorie pro testovaný text nebo se jako výsledná kategorie určí kategorie textu v trénovací sadě nejvíce pobného testovanému textu.

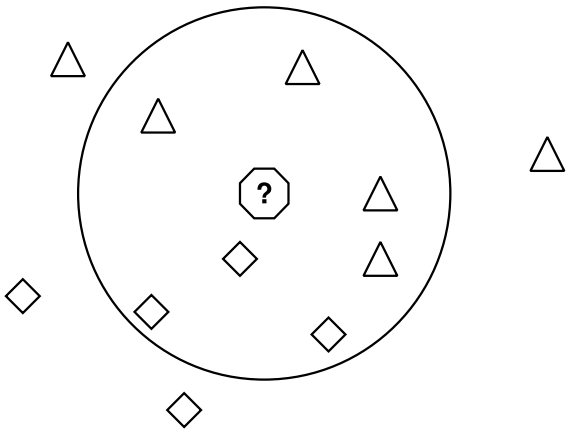
Způsobem jak vypočítat přesněji výslednou třídu pro testovací dokument je přidělení vah jednotlivým na základě podobnosti s testovacím dokumentem. Výsledné ohodnocení pravděpodobnosti že testovací text patří do kategorie c je pak suma ohodnocení nejbližších k textů k testovacímu textu patřící do dané kategorie.

$$P(c_j, x) = \sum_{d_i \in \mathrm{N}(x)} \cos(x, d_i) \cdot y(d_i, c_j)$$

kde c_j je třída pro kterou se počítá výsledné hodnocení, x je testovaný text, $\mathrm{N}(x)$ je množina nejbližších k textů z trénovací sady, $cos(x, d_i)$ je podobnost textu d_i s textem x a

$y(d_i, c_j)$ je funkce která nabývá 1 pokud dokument d_i do třídy c_j patří a 0 pokud do třídy nepatří.[12]

Jako výsledná třída je pak zvolena třída s nejvyšším ohodnocením. Princip metody je znázorněn na obrázku 1.2



Obrázek 1.2: Princip metody k nearest neighbour *Osmihran značí testovaný dokument. Kružnice symbolizuje hraniční mez maximálního okolí testovaného dokumentu. Trojúhelníky a kosočtverce reprezentují texty v testovací sadě příslušící v různým kategoriím*

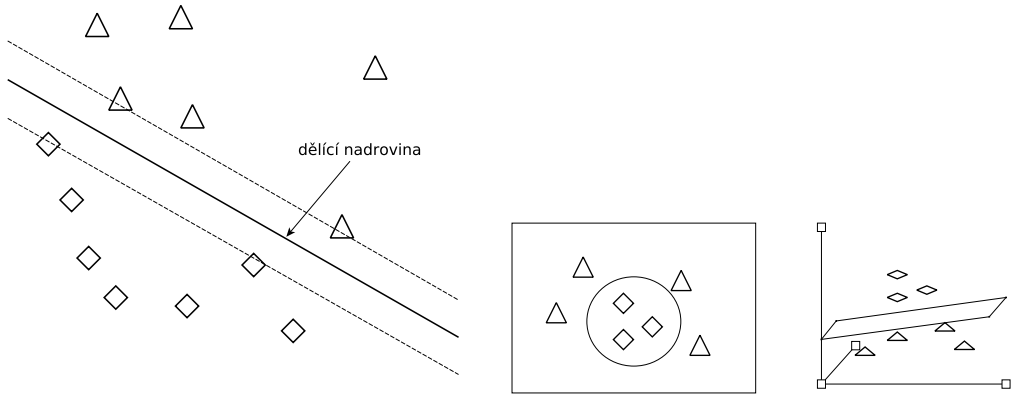
Support vector machines

Metoda support vector machines spočívá ve snaze najít nadrovinu oddělující vektory jedné třídy od vektorů druhé třídy, přičemž se hledaná nadrovina má maximální vzdálenost od krajních vektorů obou tříd, viz obrázek 1.3. Z výše uvedeného vyplývá že je potřeba vybrat z trénovacích dat ve vektorovém prostoru malou skupinu vektorů, definující pozici dělící nadroviny. Těmto vektorům se říká podpůrné vektory(support vectors). Dodržení maximální vzdálenosti dělící nadroviny od krajních vektorů obou tříd je důležité z hlediska klasifikace. Velká vzdálenost od dělících bodů znamená větší pravděpodobnost zařazení testovaného dokumentu do skupiny, protože body blízko rozpětí nejmenší vzdálenosti reprezentují prvky jejiž zařazení do dané skupiny je nejvíce nejisté. Při nemožnosti určení linearní nadroviny se používá *kernel trik*, který spočívá v transformaci vektorového prostoru do vyžší dimenze, kde již nadrovina určit jde.[3]

Rocchio

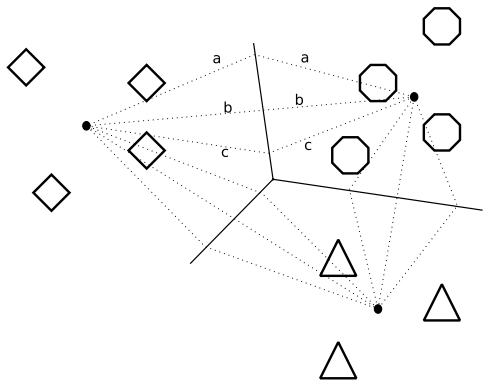
Klasifikace algorimem Rocchio pracuje podobně jako support vector machines na určení hranic oddělující vektory jedné třídy od druhé. K určení hranic nevyužívá nadrovinu, ale vzdálenost od průměrného středu skupiny vektorů reprezentující danou třídu, spočítaného jako:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$



Obrázek 1.3: Princip metody support vector machines *lineární dělicí nadrovina vlevo, kernel trick vpravo*

kde D_c je skupina textů patřící do kategorie c a $\vec{v}(d)$ je normalizovaný vektor textu. Hranici mezi skupinami pak reprezentuje úsečka vytyčená body v polovině vzdálenosti mezi jednotlivými středy sousedících skupin vektorů. Určení hranic metody rocchio znázorňuje obrázek 1.4



Obrázek 1.4: Princip metody rocchio *a,b,c značí vzdálenosti od středů k hranicím mezi skupinami vektorů*

1.4.2 Algoritmy na bázi učení bez učitelem

Algoritmy na na bázi učení bez učitele pracují obvykle na principu shlukování. Protože při učení bez učitele není dostupná informace od lidského zdroje v podobě zařazení testovacích dokumentů do tříd, algoritmy na tomto principu hledají podobnosti mezi dokumenty a vytvářejí shluky takovýchto dokumentů. Metody shlukování dělíme do dvou základních skupin:

- Ploché shlukování
- Strukturované(Hierarchické) shlukování

a nebo podle typu příslušnosti dokumentu do shluku na:

- *Tvrdé shlukování* – prvek je přiřazen právě jednomu shluku
- *Měkké shlukování* – prvek je obsažen ve částečně všech shlucích

Ploché shlukování

Cílem plochého shlukování je rozdělení prvků do shluků tak, aby nejvíce podobné prvky byly ve stejných shlucích a naopak nejméně podobné byly v různých shlucích[3]. To, které prvky jsou si podobné a které ne, záleží na uspořádání a obsahu jejich dat(klíčových vlastností). Podobnost mezi prvky vyjadřuje jejich vzdálenost od sebe ve vektorovém prostoru. Hlavní plochého shlukování je nutnost předem zvolit počet shluků. Nejznámějším algoritmem plochého shlukování je K-means

K-means

Principem shlukování K-means je minimalizovat průměrnou vzdálenost dokumentů od jejich středu na druhou. Střed shluku výpočítáme jako:

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

kde ω jsou dokumenty ve shluku. To jak moc prvky shluku patří nebo nepatří ke středu spočítáme jako sumu vzdálenosti vektoru prvku od středu na druhou. Tuto hodnotu nazýváme residuální suma čtverců(RSS):

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$

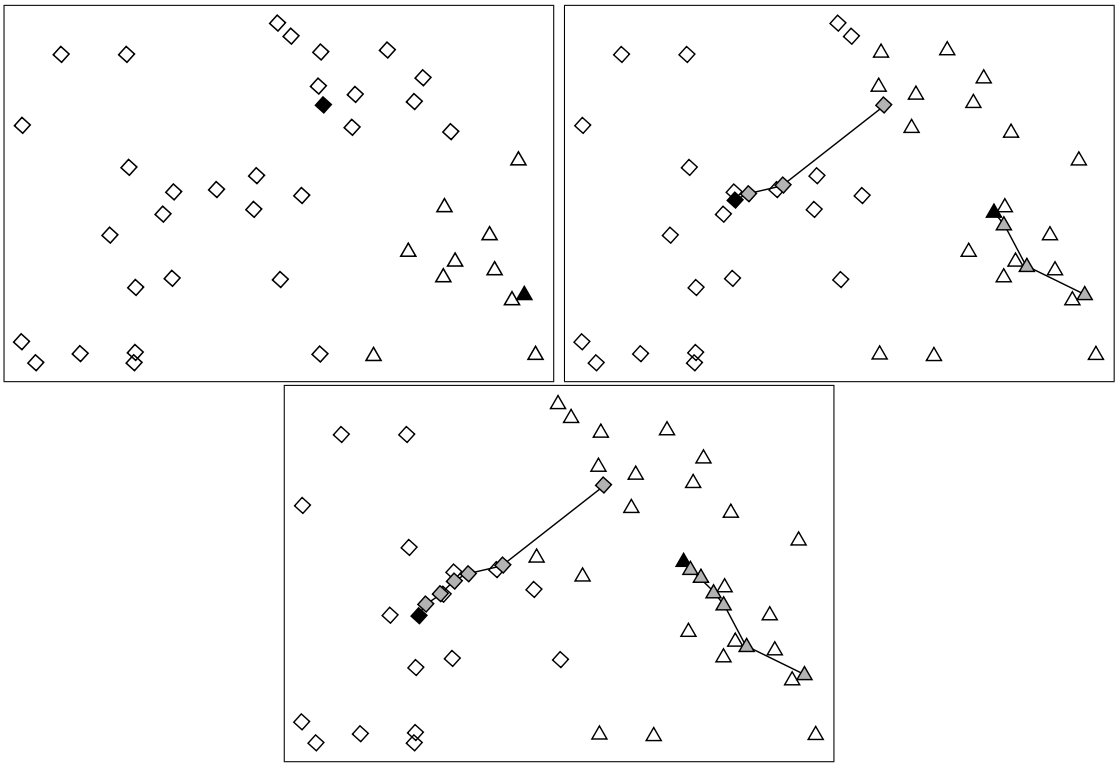
Cílem algoritmu K-means je tuto hodnotu minimalizovat. Protože však neznáme předem pozice středů, musí algoritmus optimální pozice středů nalézt. Jako první jsou středy zvoleny náhodně a poté se algoritmus iterativně počítá nové pozice středů v závislosti na vypočítaných hodnotách RSS.[3]. Počet iterací se nastavuje explicitně, jinak by algoritmus nemusel skončit nikdy. Činnost algoritmu K-means je znázorněna na obrázku 1.5

Strukturované(Hierarchické) shlukování

Hierarchické shlukování odstraňuje nedostatky plochého shlukování. Nemusíme explicitně uvést počet shluků, do kterých chceme prvky klasifikovat a výsledek shlukování nám dáva lepší představu o struktůře shluků. Zároveň je hierarchické shlukování oproti plochému deterministické. Oproti plochému shlukování, které má lieární složitost má strukturované nejméně složitost kvadratickou[3] Hierarchické shlukování může být:

- sjednocující (down top)
- rozdělující (top down)

Sjednocující hierarchické shlukování bere každý dokument jako jeden shluk a postupně tyto shluky slučuje až do jednoho veklého shluku. Rozdělující shlukování pracuje na opačném principu tj. rozdělování jednoho velkého shluku postupně až do stavy kdy v jednom shluku bude jeden dokument [3]. Ukázka principu Strukturovaného shlukování je na obrázku 1.6



Obrázek 1.5: Princip metody kmeans *1,4 a 7 iterace*

1.5 Techniky měřící kvalitu klasifikátoru

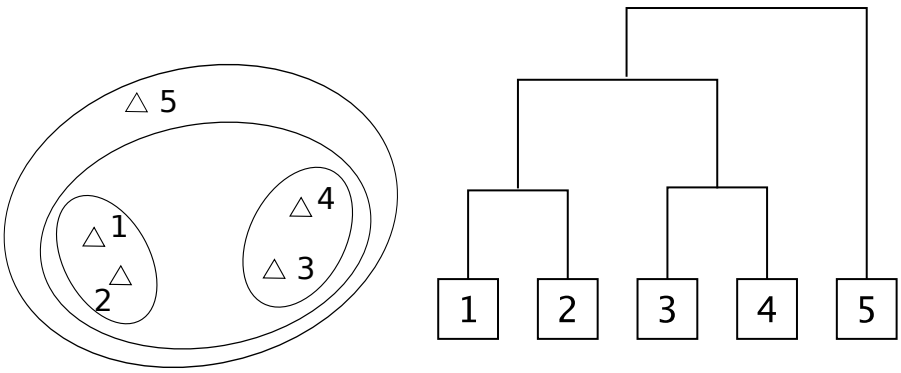
Po výběru klasifikační metody je vhodné otestovat kvalitu klasifikátoru. Nejběžnější kritéria pro testování kvality klasifikátorů jsou přesnost, správnost, citlivost a F-measure. Pro výše zmiňované techniky bude použito značení negativních a pozitivních předpovědí systému a skutečného zařazení do kategorie. Předpokládejme že systém klasifikuje položku jako positivní(P), pokud do třídy patří a negativní(N) pokud do třídy nepatří. Pokud položka do třídy skutečně patří, označíme ji jako True(T) a pokud do třídy nepatří, označíme ji jako False(F). Tím dostáváme čtyři kombinace možných výsledků: TP,TN,FP,FN.

Správnost(Accuracy) výjadřuje poměr správně určených kategorií testovaných textů ku celkovému počtu provedených předpovědí. Vzorec je tedy

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Přesnost(Precision) udává kolik testovaných textů označených jako pozitivních, do dané kategorie skutečně patří. Vzorec pro přesnost je:

$$\text{Precision} = \frac{TP}{TP + FP}$$



Obrázek 1.6: Princip hierarchického shlukování

Citlivost(Recall) vyjadřuje poměr dokumentů, které byly správně označeny jako pozitivní, vůči počtu všech spravných předpovědí. Z toho vyplývá vzorec pro přesnost:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F-measure je harmonický průměr citlivosti a přesnosti.Používá se tehdy, kdy chceme mít představu o kvalitě klasifikátorů vyjadřenou jedním číslem. Vzorec pro F-measure je:

$$\text{F} - \text{measure} = \frac{2 \cdot \text{Precission} \cdot \text{Recall}}{\text{Precission} + \text{Recall}}$$

1.6 Bayesův naivní klasifikátor

Protože je v této práci použit pro klasifikaci RSS příspěvků do kategorií Bayesův naivní klasifikátor, princip klasifikace potomcí tohoto klasifikátoru zde bude rozebrát podrobněji.

1.6.1 Klasifikace na základě pravděpodobnosti

Při použití zjednodušeného modelu klasifikovaného objektu nevíme s určitostí do jaké třídy objekt zařadit. Můžeme ale říci, že objekt patří do některé třídy s určitou pravděpodobností. Uvažujme klasifikaci objektu do dvou tříd, označných jako c_1 a c_2 bez toho abychom znali jakékoliv další informace o daném objektu. Můžeme tvrdit, že objekt patří s pravděpodobností $\text{P}(\omega_1)$ do třídy c_1 a s pravděpodobností $\text{P}(\omega_2)$ do třídy c_2, například podle znalosti, že objekty patří z 90 % případů do třídy c_1, zatímco objetky patřící do třídy c_2 jsou spíše vzácné. Abychom při klasifikaci zajistili nejmenší možnost chyby, stanovíme rozhodovací pravidlo, podle kterého bude objekt patřit do třídy c_1 pokud bude $P(\omega_1) > P(\omega_2)$ a v opačném případě do třídy c_2, což v našem případě znamená že všechny objekty klasifikujeme do třídy c_1 a budeme se dopouštět chyby s pravděpodobností $P(\omega_2)$.

Pokud o objektu známe nějakou informaci, na základě znalostí o klasifikovaných objektech můžeme stanovit hustotu rozložení pravděpodobnosti, že objekt patří do třídy c_1 nebo c_2, podle vlastnosti x

$$p(x|\omega_1) \text{ a } p(x|\omega_2)$$

Jinými slovy stanovíme funkci vyjadřující pravděpodobnost toho, že objekt patří do dané skupiny v závislosti na hodnotě získané vlastnosti objektu. Při znalosti vlastnosti x stanovíme podmíněnou pravděpodobnost toho, že objekt mající vlastnost x patří do třídy c jako:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \tag{1.1}$$

kde je v našem případě

$$p(x) = \sum_{i=1}^{2} p(x|\omega_i)P(\omega_i)$$

kde i nabývá hodnot 1 nebo 2. Výše uvedený vztah je Bayesův podmíněné pravděpodobnosti.[8]

1.6.2 Klasifikace Bayesovým klasifikátorem

Naivní Bayesův klasifikátor je jedním z nejednodužších klasifikátorů založený na pravděpodobnosti. Pro určení zda text patří do danné třídy nebo ne, počítá produkt pravděpodobností výskytů výrazu textu v danné třídě vynásobený pravděpodobností třídy. Z těchto hodnot vrací maximální pravděpodobnost. Naivní se nazývá proto, že neuvažuje vzájemnou závislost výskytu jednoho výrazu na druhém.

Protože nás obvykle zajímá pouze nejpravděpodobnější kategorie, můžeme bayesův vzorec 1.1 zjednodušit vypuštěním jmenovatele, který má za úkol zmenšit výslednou pravděpodobnost do rozmezí $\langle 0; 1\rangle$. Zjednodušený vzorec má pak podobu:

$$P(\omega_i|x) = \text{argmax}p(x|\omega_i)P(\omega_i)$$

Vzorec pro výpočet nejpravděpodobnější kategorie je následující:

$$D_c = \text{argmax}_{c_j \in C} P(c_j)\Pi_{i=1}^{d} P(w_i|c_j)$$

Kde C je množina všech kategorií a W množina všech klíčových výrazů v klasifikovaném textu.[3]

1.7 NLTK

NLTK je knihovna pro zpracování přirozeného jazyka. Původně byla vyvinutá v roce 2001 jako součást kurzu počítačové lingvistiky v oddělení počítačové a informační vědy univerzity v Pensilavanii. NLTK poskytuje velke množství nástrojů a hotových algoritmů, které oproštují řešitele od nutnosti zabývat se implementací algoritmů a dovolují mu soustředit se na problémy týkající se zpracování přirozeného jazyka. Mimo algoritmy knihovna obsahuje značné množství příkladů a spoustu testovacích dat, které mohou dobře posloužit pro testování a pomoci se orientovat v problematice. Významné je z pohledu této práce podpora následujících procesů. [1]

Tokenizace
Pro tokenizaci nabízí NLTK širokou škálu tokenizérů, které rozdělí text do sekvence významových celků. Pro rozdělení podle bilých znaků lze použít hned několik tokenizéru, dále pak tokenizér pracující s regulárními výrazy, tokenizér pracující na bázi algoritmu textTilling nebo tokenizer využívající konvencí Penn Treebank.

Stemming
Ze stemmerů poskytuje NLTK jako stemmery založené na odtrhávání koncovek jako Porterův stemmer, nebo Lancaster(Paice/Husk) stemmer, tak i stemmer založený na slovníkové bázi, Wordnet stemmer. Vlastní skupinu zastává stemmer pracující na bázi regulárního výrazu. Krom hotových stemmerů poskytuje NLTK i snowball stemmer, framework pro možnost vytvoření vlastního stemmeru. Modul snowball stemmeru poskytuje zároveň i ukázkové stemmery pro rozličné jazyky.

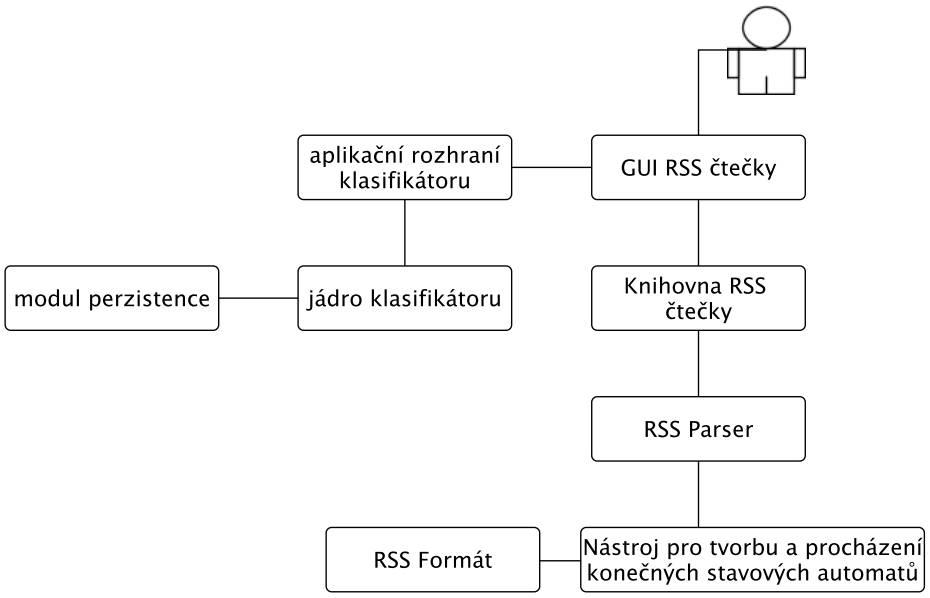
Podpora N-gramů
modul *tag* obsahuje třídy pro tvorbu N-gramů. K další možnostem tohoto modulu a k možnostem získání základních prvků vektoru ve vektorové reprezentaci dokumentu, patří RegexpTagger rozdělující slova podle regulárního výrazu nebo například AffixTagger zalžený na získání prvku podle předpony nebo přípony slova.

Klasifikátory
Z tříd pro klasifikaci jsou v nltk zastoupeny jak klasifikátory pracující na metodě učení s učitelem jmenovitě: naivní bayesův klasifikátor, rozhodovací stromy a klasifikátor pracující na principu maximální entropie, tak klasifikátory pracující na metodě učení bez učitele a principu shlukování. Pro ploché shlukování to je K-means, EM. Pro hierarchické shlukování je připraven modul aglomerativního(sjednodujícího) shlukování.

Testovací data Modul corpus nabízí obrovské množství testovacích a lexikografických dat připravených ke zpracování nebo už přímo rozdělených do kategorií přiravených pro testování a demonstrování. Mimo to obsahuje i slovníky stop slov pro různé jazyky.

Kapitola 2

Implementace



Obrázek 2.1: Schéma implementace

2.1 Technologie a použité nástroje

Pro implementaci používaných knihoven a čtečky byly použity následující nástroje:

- python 2.7.1

V jazyku python byly použity následující nadstandartní moduly:

- NLTK 2.0b9
- Tk interface eXtension 8.4.3-2 (pro GUI RSS čtečky)
- unittest2 0.5.1-2

2.2 Modul pro čtení RSS kanálu

Pro potřebu této práce byla vyvořena knihovna umožnující kompletní správu RSS kanálů. Základní myšlenkou této knihovny je vytvoření rozhraní poskytující operace nad určenými RSS kanály, jako je jejich správa, archivace získaných příspěvků a upozornění na nejnovější příspěvky. Důvodem bylo získání získání nezávislého základního jádra nad kterým je možné stvořit libovolné uživatelské rozhraní. Knihovna dále poskytuje funkce pro uložení a načtení nastavení RSS čtečky. Pro zpracování vybraných RSS kanálů používá knihovna RSS parser rovněž implementovaný v rámci této práce. Princip a funkce poskytnuté implementovaným parserem budou rozebrány v sekci 2.2.2.

2.2.1 Detailní popis funkcí a principů RSS knihovny

V této části budou pustupně rozbrány funkce nabízené modulem pro čtení RSS. Budou zde zmíněny vzniklé problémy při implementaci a popsáno jejich vyřešení.

Fukce poskytované modulem jdou schrnout do tří obecných oblastí:

- Perzistence
- Správa RSS kanálů
- Získávání informací z RSS kanálů

Perzistence

Aby byla knihovna použitelná, musí splňovat základní potřebu uživatele, kterou bezesporu je možnost obnovit předcházející stav používaného programu. Při zohlednění vysledného souboru konfigurace bereme v úvahu následující fakty:

- Nutná potřeba uchování všech kanálů
- Informace jako popis a titulek kanálu by měli být přístupné i když uživatel není v síti.
- Uživatel by měl mít přístup ke všem v minulosti získaným položkám v kanálu

Pro formát uložení konfigurace byl zvolen značkovací jazyk XML. Výhoda textového formátu konfigurace oproti binárnímu je v možnosti jeho modifikaci i mimo aplikaci. Formát souboru obsahuje seznam elementů pojmenovaných *Source*, které obsahují informace o konkrétním RSS kanálů. Informace o kanálu, které se budou uchovávat v konfiguraci byly zredukovány na nutné minimum a to: titulek, popis kanálu, odkaz na soubor kanálu, odkaz na stránku obsahující příspěvky uváděné v kanálu a čas poslední aktualizace kanály. Volitelně se ukládají informace o tom, které dny(skip days) a hodiny(skip hours) není potřeba přistupovat ke kanálu, doba platnosti aktuálnosti kanálu (ttl) a speciální informace, které umožnují do konfiguračního souboru ke kanálu uložit libovolnou skupinu informací. Tato možnost je využita pro uložení informace o přirozeném jazyku RSS kanálu, protože jak vyplývá ze sekce 1.1.3, informace o jazyku kanálu je ve formátu RSS volitelná a nelze na ni tudíž spoléhat.

Tato základní konfigurace je dostatečuje potřebám uvedeným výše. Volitelné ukládání informací o vynechaných dnech a hodinách pro kontrolu aktuálnosti a doba platnosti aktuálnosti kanálu, umožňuje respektovat tuto volbu u poskytovatele kanálu a nezatěžovat server.

Archivace získaných příspěvků je vyjmuta z konfiguračního souboru hned z několika důvodů:

- Oproti konfigurace zdroje by měl příspěvek obsahovat co nejvíce informací. Což by při velkém počtu archivovaných příspěvků vedlo k nakynutí konfiguračního souboru do velkých rozměrů
- Uchovávané archivované příspěvky v konfiguračním souboru by vedly k jeho nepřehlednosti a tím by zmizela výhoda textové konfigurace. Oddělení archivu příspěvků od konfigurace umožnuje číst jednotlivé příspěvky podle potřeby, nikoliv nutně při načítání konfigurace.
- Ukládání příspěvků do archivu by mělo být uskutečněno hned po jejich získání ze zdroje, zatímco ukládání konfigurace by mělo být spouštěno na žádost uživatele.

Archivace příspěvků

Protože je vhodné aby měl uživatel možnost přístupu ke starším příspěvkům, které již z důvodu aktualizace v kanálu nejsou dostupné, modul automaticky archivuje veškeré získané příspěvky. Při výběru formy archivace byly brány v potaz tyto fakty:

- při větším počtu kanálů může počet příspěvků a rychle narůstat
- Uživatel obvykle nepotřebuje přistupovat k informacím příspěvku jinak než skrz aplikaci (při hledání konkrétního článku si obvykle najde článek na stránkách na které kanál odkazuje)
- Čtečka by si měla pamatovat které příspěvky si uživatel přečetl a které zůstaly nepřečteny.
- Formát ukládání by měl být schopen rozlišit dva příspěvky lišící se v pouze v nepatrných detailech. V potaz je nutné vzít fakt, že element *guid* nelze využít protože je elementem volitelným, jak vyplývá ze sekce 1.1.3.
- Při smazání RSS kanálu z konfigurace a jeho opětovném přidání, by měla být čtečka schopna přidružit archivované příspěvky k příspěvkům ke kanálu, tzn. příspěvky v archivu by měli být jednoznačně identifikovatelné bez použití časem nebo stavem se měnících faktorů(čas získání příspěvku, pozice příspěvku v kanálu atd.)
- Čtečka by si měla pamatovat uživatelem smazané příspěvky do doby kdy je bude kanál obsahovat, aby nenastala situace, kdy bude označen za nový příspěvek, který byl již uživatelem v minulosti smazán

Při uvážení výše zmíněných důvodů byl zvolen přístup ukládání příspěvků jednotlivě do souborů, do složky odpovídající příslušnému kanálu. Pro pojmenování souborů bylo zvolen cyklický renundantní součet s přidaným flagem, značící zda byl nebo nebyl příspěvek uživatelem přečten. Složka s archivem příspěvků příslušného kanálu, dále obsahuje textový soubor *files*, který cyklické kontrolní součty archivovaných příspěvků oddělené koncem řádku a zjednodušuje tak přístup k jednotlivým příspěvkům a kontrolou zda daný příspěvek již byl z kanálu získán nebo ne. Složka dále volitelně obsahuje soubor *removed*, který obsahuje kontrolní součty uživatelem smazaných příspěvků, které se však stále vyskytují v kanálu.

Správa RSS kanálů

Modul RSS čtečky umožňuje uživateli přidání a odebrání kanálu na základě jednoznačného identifikátoru kanálu, kterým je URL spravovaného kanálu. Při odebírání kanálu defaultně nejsou mazány příspěvky v archivu, což lze změnit parametrem purge. Přidání kanálu defaultně následuje kontrola dostupnosti kanálu a stažení aktuálních příspěvků z kanálu. Pokud kanál není dostupný, do seznamu kanálů se nepřidá. Při odstranění příspěvku z archivu se smaže soubor obsahující příspěvek a jeho název se přidá do souboru obsahující smazané příspěvky. Knihovna poskytuje možnost měnit stav příspěvků z nepřečtených na přečtené a obráceně. Změna stavu způsobí pouze přejmenovaní souboru obsahující příspěvek na název bez nebo s přefixem, značícím zda byl příspěvek přečten nebo ne.

kontrola aktualnosti kanálu

Pro kontrolu aktuálnosti kanálu slouží metoda *isRecent*. Rozhodování zda je či není kanál aktuální, resp. zda má smysl posílat požadavek na server, probíhá podle několika kritérií:

- čas poslední aktualizace
- ttl hodnota kanálu
- skipDays, skipHours hodnoty kanálu

Pokud není nastaven čas poslední aktualizace metoda automaticky vrací nepravdu, protože od tohoto času se počítají všechny ostatní. Čas poslední aktualizace odpovída elementu *pubDate* případě elementu *lastBuildDate*, který by měl být ještě aktuálnější, v schématu RSS formátu. Dále se zjišťuje zda nemá zdroj nastaveny některé dny nebo hodiny k vynechání kontroly. Jako poslední se kontroluje zda není ve zdroji dostupná hodnota ttl a v případě kdy je, tak zda aktuální čas překročil nebo nepřekročil hodnotu stanovenou v ttl. Pokud se u kanálu nepotvrdí aktuálnost díky dnům nebo hodinám, ve které kanál zůstává neaktualizovaný, ani díky hodnotě ttl, považuje se kanál za neaktuální, protože není dostupná jakákoliv informace, která by tento předpoklad vyvracela nebo potvrzovala Funkce tedy vrací nepravdu a čtečka nebude vysílat požadavek na server o novou verzi kanálu.

Aktualizace kanálu

Aktualizace kanálu probíhá přes standartní HTTP požadavek GET na server poskytující RSS kanál. Po úspěšném získání aktuálního souboru kanálu se tento soubor přečte a jeho obsahu se předhodí parseru zmiňovaném v sekci 2.2.2. Dál se zkontroluje zda kanál neobsahuje elementy *pubDate* případně *lastBuildDate* a pokud ano, nastaví se poslední aktualizace kanálu na novější z těchto hodnot. O konverzi časových pásem je postaráno již v parseru takže obě hodnoty jsou v GMT V následujícím kroku se zkontroluje za existuje složka, ve které se nachází zaarchivované příspěvky kanálu. Pokud ne vytvoří se. Prov všechny příspěvky které se v aktuálně nachází v kanálu se spočítá jejich CRC a zaarchivují se. Při této procedůře se kontroluje zda již přispěvek v archivu není a zda není příspěvek v seznamu vymazaných příspěvků, které jsou uloženy v soboru *removed*. Při tomto ukládání se zároveň kontroluje,které příspěvky v souboru *removed* jsou ještě aktuálně dostupné v kanálu a které již není možné při další aktualizace získat. Smazané příspěvky, které již v kanálu nejsou dostupné se odeberou ze seznamu smazaných příspěvků. Takto lze jednoduše udržet maximální velikost seznamu odebraných příspěvků, která je rovná maximálnímu počtu příspěvků vyskytujících se v kanálu, který bývá konstatní.

Získávání informací z kanálu

Pro získávání informací z kanálu jsou dostupné tyto funkce:

- získej seznam kanálů
- získej seznam kanálů (detaily) – tato funkce vrací oproti předchozí seznam kanálů, ke kterým jsou dostupné základní informace jako název, adresa a popis
- získej seznam příspěvků z kanálu – vrací seznam trojic (přečteno/nepřečteno, jméno souboru obsahující příspěvek, název příspěvku) a počet vrácených příspěvků které jsou nepřečtené.
- získej podrobné informace o příspěvku
- získej seznam nepřečtených příspěvků ze zdroje
- získej extra data zdroje – vrací extra atributy přiřazené ke zdroji

2.2.2 RSS parser

Pro zpracování zdrojového souboru byla použita knihovna expat poskytující třídu pro parsování XML dokumentů metodou SAX. Výhoda metoda SAX je v sekvenčním zpracování XML souboru a tudíž i možnosti průběžně kontrolovat povolené možnosti formátu jednotlivých elementů. Nevýhoda spočívá v nutnosti explicitně ukládat zpracovávaná data a vytváře struktůru jejich uložení, která je použitím DOM parseru k dispozici automaticky. Manuální tvorba hierarchie dat ovšem umožňuje ukládat pouze vybraná data a nepotřebné informace ignorovat. Využitím této třídy byl implementován parser pro formát RSS. Jak je patrno ze sekce 1.1.3, rozsah formátu RSS je obšírný a kontrola správnosti formátu není jednoduchá. Z tohoto důvodu byl implementován nástroj umožnující vytvoření konečného stavového automatu respektující syntaktická pravidla jazyka XML.

2.2.3 Nástroj pro tvorbu stavového automatu

Protože ruční tvorba automatu, který by pokryl všechny možnosti formátu RSS, by byla značně náročná, těžko spravovatelná a náchylná na chyby, byl v této práci implementován nástroj na zjednodušení tvorby konečného stavového automatu, pokrývající veškeré možnosti specifikovaného formátu a umožnující kontrolu správnosti tohoto formátu.

Obecně lze pomocí toho nástroje vytvořit libovolný stavový automat, ovšem funkce jsou uzpůsobeny tak aby splňovaly zásady jazyka XML a poskytnuly co nejjednodužší a nejméně pracnou tvorbu automatu. Nástroj nabízí tři možnosti přechodů z počatečního stavu do koncového:

- standartní přechod realizovaný operátorem porovnání
- přechod realizovaný na základě regulárního výrazu
- defaultní přechod provedený při výskytu jakéhokoliv vstupu

Díky možnosti kontrolovat vstup pomocí regulárního výrazu lze pokrýt i specifické formaty jednotlivých elementů vyskytujících se ve formátu RSS jako je například datum nebo výčet dnů v týdnu.

Pro konstrukci přechodů nabízí nástroj následující možnosti:

- přechod z jednoho stavu do jiného stavu
- přechod z jednoho stavu do vice stavů s ohledem na různé vstupy
- přechod ze stavu vracející se zpět do stejného stavu
- přechod vracející se zpět do stejného stavu pro více stavů s ohledem na různé vstupy

U všech přechodů je navíc možnost volby akce při, které se může přechod vykonat (například začátek elementu, konec elementu atd.) a možnost zpětné akce tj. akce při které se vykoná opačně (z koncového stavu do cílového). Dále může automat při provedení přechodu reagovat zadanou funkcí, která se vykoná po provedení přechodu a nese sebou informace a typu přechodu a vstupních datech. Díky této možnosti je automat schopný nejen kontrolovat správnost syntaxe formátu, ale i extraovat z něho určité informace a navrátit je ve struktuře. V této práci jsou jako parametry přechodů voleny statické funkce třídy nesoucí potřebné informace extraované z formátu.

konfigurace nástroje pro formát RSS

V rámci této práci byla implementována konfigurace konečného stavového automatu pro formát RSS v 2.0. Tato konfigurace vystihuje veškeré možnosti nabídnuté formátem. Konfigurace kontroluje správnost povolených hodnot u prvků jako je datum, ttl apod. Finální automat má 46 stavů a 127 přechodů.

2.2.4 Modul klasifikátoru

Pro klasifikaci byl bayesův naivní klasifikátor z knihovnny NLTK. Protože je však tento modul pojmut tak že pro uživatel neznalý principů klasifikace bayesovým klasifikátorem nemusí být snadné ho používat, byl v rámci této práce implementován modul, který obaluje modul poskytlý knihovnou NLTK a zároveň odstiňuje uživatele od nutnosti předzpracování textů jako je tokenizace, lematizace, stemming a odstranění stopslov. Modul je rozdělen na tři hlavní části:

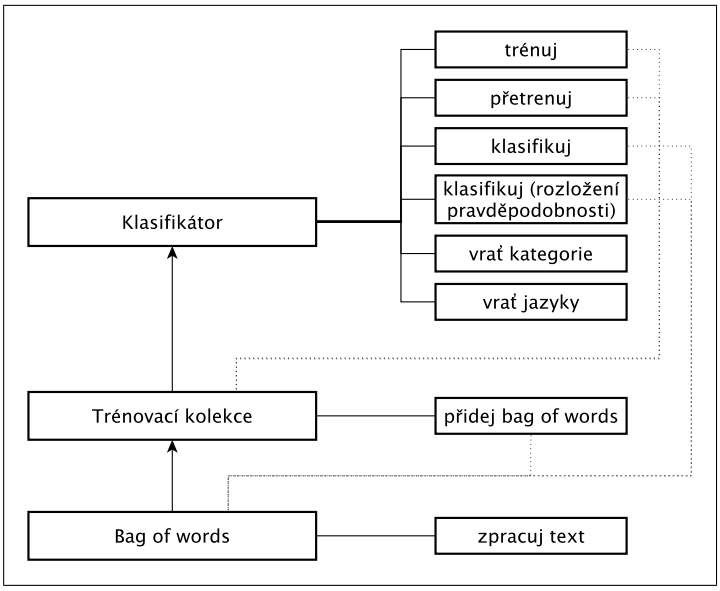
- Jádro
- Modul perzistence
- Aplikační rozhraní

Modul byl rozdělen na více částí kvůli možnosti pro případné budoucí rozšíření.

Jádro

Jádro je hlavní modul, který obsahuje samotný klasifikátor. Zde jsou definované používané metody pro klasfikaci. I když tento modul klasifikaci značně zjednodušuje, stále vyžaduje znalosti uživatele ohledně klasifikace. Na obrázku 2.2 je zobrazeno schéma jádra

Jako základní klíčové prvky jsou použity v klasifikátoru slova. O zpracování textu se stará modul *bagofwords*. Výstupem tohoto modulu je zpracovaný text do vektorové podobny s binárním vážením. Pro odstranění stop slov používá tento modul extérní slovníky ve složce klasifikátoru. Důvodem použití extérních slovníku stop slov je chybějící slovník pro česká stop slova v knihovně NLTK. K dispozici jsou slovníky pro české a anglické stop slova převzaté z [2]



Obrázek 2.2: Schéma jádra klasifikátoru

Modul bagofword při zpracování textu přímá paramatr *id*. Tento paramatr má za úkol jednoznačně identifikovat text který bude modulem zpracován. Pokud je parametr *id* vynechán použije se jako *id* cyklický redundantní součet textu. Důležitost parametru *id* spočívá v následné identifikaci textu v množině natrénovaných textů a možnosti jeho odebrání z trénovací sady. Výstup tohoto modulu používá modul *trainCollection*, který vytváří trénovací sadu pro klasifikátor. Z důvodu nutnosti přetrénovat vždy celý klasifikátor při změně trénovací sady, podporuje tento modul *trainCollection* i operátor plus, umožňující jednoduché spojení dvou trénovacích sad.

Aplikační rozhraní

Modul aplikačního rozhraní byl vybudován z důvodu absolutní abstrakce uživatele od klasifikace do podoby kdy jednoduše určí texty kterými chce klasifikátor natrénovat a následně texty, které chce klasifikovat. Další myšlenkou oddělení aplikačního rozhraní od jádra byla možnost použití například vzdálených klasikátorů, které budou uloženy někde na vzdáleném servru a přes aplikační rozhraní bude možnost tyto klasifikátory využívat. Trénovaní a klasifikace klasifikátoru také nemusí probíhat pouze přes textový formát, ale může pracovat například s xml formátem nebo nějakým komprimovaným archivem obsahující text.

V rámci této práce bylo navrženo aplikační rozhraní, které by mohlo být jakýmsi prototypem pro ostatní aplikační rozhraní. Obsahuje však všechny nezbytné funkce,které jsou ke klasifikaci a trénovaní klasifikátoru potřeba. Implementované aplikační rozhraní pracuje pouze s lokálním klasifikátorem předáváným přes konstrukor instance. Trénování a klasifikace jsou prováděny pouze textovým vstupem. K dispozici jsou následující funkce:

- trainStr(text,kategorie,jazyk) – natrenuje klasifikátor novým textem, který je přiřazený do kategorie. Jazyk je nutné uvést aby klasifikátor věděl ze kterého slovníku má odstraňovat stopslova.
- classifyStr(text,jazyk) – zpracuje text a vrátí pro něj nejpravděpodobnější kategorii.

- accuracyStr(text,jazyk) – zpracuje text a vrátí pole procentuální pravděpodobnosti příslušenství textu každé kategorie, kterou klasifikátor obsahuje.
- getLabels – vrací seznam všech kategorií, které klasifikátor obsahuje
- removeTrained – Podle určeného id odstraní tento text z trénovací sady a celý klasifikátor přetrénuje

Perzistence

Perzistence byla zavedena z důvodu obecnějšího použití klasifikátoru. Může se stát, že z důvodu rozsáhlého počtu kategorií a velikosti trénovací sady bude potřeba natrénovaný klasifikátor ukládat v komprimované podobě. Nebo pro rychlejší přístup může vyvstat požadavek uložení natrénovaného klasifikátoru do databáze. V této práci byl implementován prototypový modul perzistence, který poskytuje serializaci a deserializaci klasifikátoru z textu a dále modul, který na prototypový modul navazuje a umožňuje ukládání a načítání jádra klasifikátoru do a ze souboru. I přesto, že po serializaci výsledný text reprezentující jádro klasifikátoru obsahuje spoustu nepotřebných informací, umožňuje ponechat modulu perzistence jistou nezávislost a při případné změně jádra klasifikátoru není nutné měnit modul perzistence.

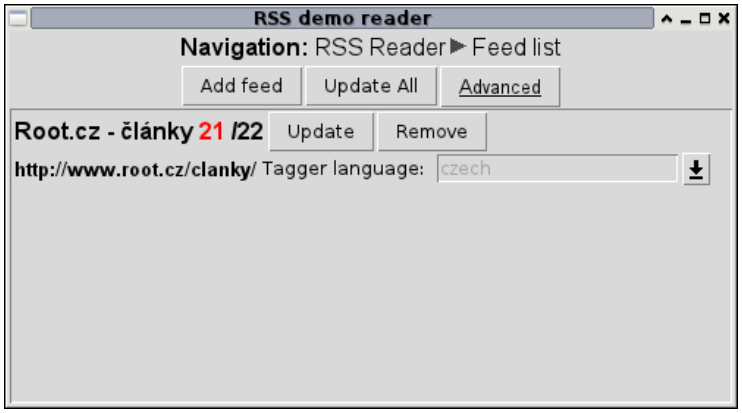
2.3 Uživatelské rozhraní

Pro demonstraci funčnosti RSS čtečky bylo implementováné jednoduché uživatelské rozhraní v prostředí Tk s využitím Tk interface eXtesion library. Aplikace propojuje modul RSS čtečky s modulem klasifikátoru. Konfigurace aplikace je ve dvou souborech s příponami ".conf"pro konfiguraci čtečky a ".cls"pro konfiguraci klasifikátoru. Rozhraní je koncipováno na minimální rozměry a jako základní navigační prvek slouží navigační lišta, která je viditelná po celou dobu používání programu a umožňuje uživateli okamžitý návrat ke kterékoliv z předchozích otevřených sekcí. Zanesení klasifikátorů je do rozhraní programu zavedeno přes tlačítka *Advanced* viz obr 2.3 a zaškrtávací pole verbose viz obr 2.4. V prvním případě se jedná o možnost stanovit přirozený jazyk RSS kanálu. To je nutné pro odstranění stop slov, protože automatické rozpoznání přirozeného jazyka modul neumožňuje. V druhém případě je zobrazeno přiřazení kategorií k příspěvku uživatelem a jejich možnost odebrání. Při zapnutí rozšířeného módu je také možnost přidat ke příspěvku novou kategorii.

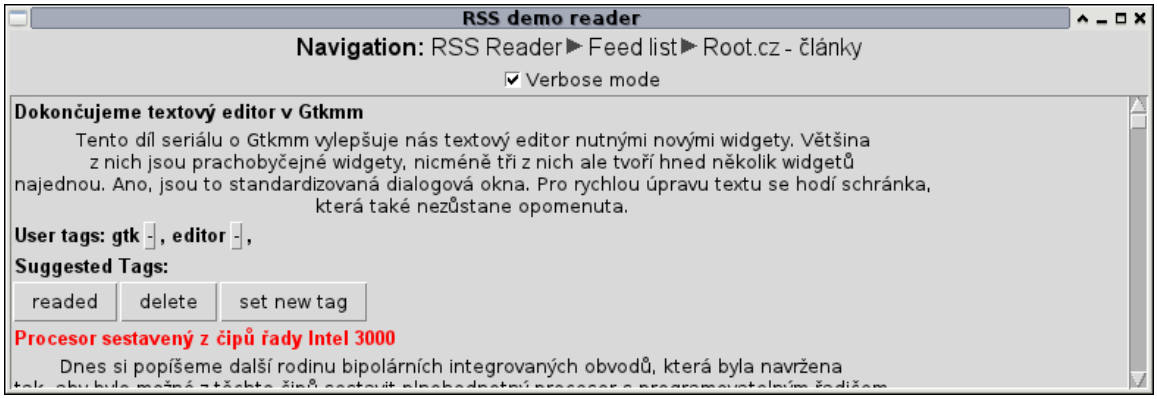
2.4 Omezení v rámci implementace

Omezení modulu klasifikátoru
Implementovaný modul klasifikátoru umí pracovat pouze s Bayesovým naivním klasifikátorem a neumožňuje tuto volbu změnit. Rovněž zpracování textů je řešeno pouze Porterovým stemmerem a jako základní jednotka klíčových vlastností dokumentu se berou slova s binárním vážením. Modul předpokládá existující slovníky stopslov (které v rámci této práce obsahují pouze češtinu a angličtinu).

Omezení modulu čtečky
Modul neumožňuje nastavení síťového připojení přes proxy. Pro archivaci je použita složka ve které se nachází čtečka a předpokládá se že uživatel bude mít opravnění zápisu a čtení v této složce.



Obrázek 2.3: GUI demostrace RSS čtečky



Obrázek 2.4: GUI demostrace RSS čtečky

Omezení modulu parseru

Modul umí parsovat pouze formát RSS v 2.0(a zpětně kompatibilní), protože cílem práce není poskytnout plnou podporu všech formátů online syndikace. Modul pro procházení a tvorbu konfigurací konečných stavových automatů, který je parserem využívám, nedokáže kontrolovat atributy vstupních elemenentů. Tyto atributy pouze předává uložné tříde, kde se o kontrolu musí postarat programátor této třídy.

Kapitola 3

Testování

3.1 Schromáždění testovacích dat

Pro zjištění zda bude naivní bayesův klasifikátor vyhovovat pro klasifikaci RSS příspěvků byla v rámci této práce provedena simulavaná klasifikace příspěvků, které by mohly odpovídat příspěvkům RSS kanálu. Pro získání testovacích příspěvků bylo staženo 9011 článku ze servru examiner.com, který poskytuje obrovské množství článků z různým oblastí. Pro testování byly vybrány kategorií:tv,education,movies,NHL (více tabulka 3.1)

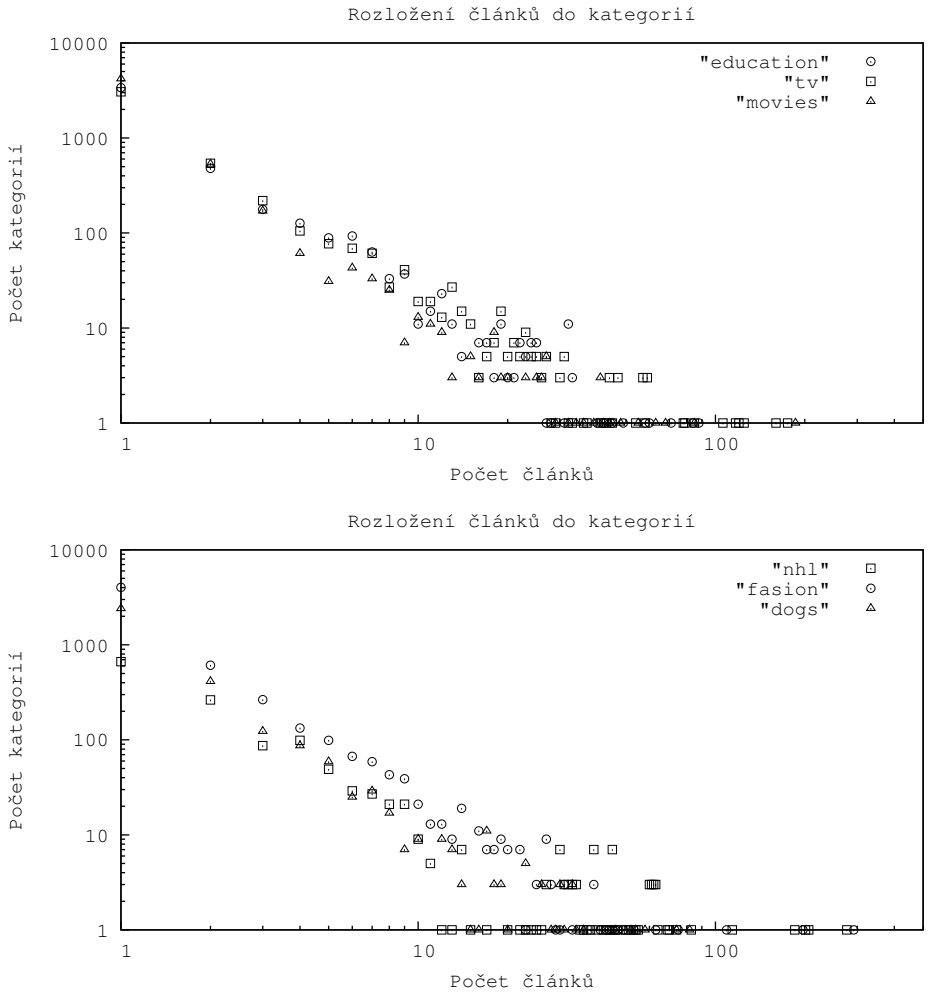
Kategorie	Staženo článků	Počet kategorií	Průměrný počet článků v kategorii [1]
tv	1785	2236	6.45
education	1364	2338	4.91
movies	1185	2617	3.87
NHL	1465	715	19.34
dogs	1427	1640	4.40
fasion	1785	2776	4.92

Tabulka 3.1: Testované kategorie

Tyto články lze použít jako trénovací data, díky značkám, které jsou zobrazeny na stránce se článkem. Při procesu sbírání testovacích dat vznikly následující poznatky:

- Příspěvek může obsahovat pouze velmi strohý nebo téměř žádný popisek. Často může být příspěvekem fotoreportáž nebo pouze upozornění na odkaz. V takovém případě je může klasifikátor klasifikovat příspěvek pouze podle velmi malé informativní hodnoty.
- Podle rozdělení článků z testovací dat do kategorií vyplynulo (viz obr 3.1), že značný počet kategorí bude obsahovat pouze velmi málo článků, což lze předpokládat i u příspěvků v RSS kanálu
- Většina příspěvků bude spadat do více než jedné kategorie(jak lze usoudit z tabulky 3.1). Což znamená klasifikaci pro více než jednu skupinu

[1] Pro počítání průměrného počtu článku v kategorii se bere v úvahu že jeden článek může být ve více kategoriích



Obrázek 3.1: Hustota rozložení článků

3.2 Testování klasifikátoru

Při testování klasifikátoru byly testovány kategorie které měli alespoň 10 příspěvků. Klasifikátor byl testován dvěmi způsoby:

- Kontrola odlišností podkategorií v různých hlavních kategorií
- Kontrola odlišností podkategorií ve stejné hlavní kategorii

3.2.1 Kontrola odlišností podkategorií v různých hlavních kategorií

Tento test spočíval v otestování zda klasifikátor bez problému rozliší dvě podkategorie dvou různých hlavních kategorií, například podkategorie "ponytail"z kategorie "fasion"proti podkategorii "Edmonton"z kategorie "NHL". Kategorie mezi sebou byli testovány podle počtu článků, které obsahovaly a to v čtyřech intervalech: 50-40,40-30,20-10. Výsledky testu jsou uvedeny v tabulce 3.2

Tabulka 3.2: Výsledky kontroly odlišnosti podkategorií v různých hlavních kategoriích *Uváděné hodnoty jsou průměrnou hodnotou uvedené v prvních řádcích tabulek*

Tabulka výsledků pro interval 50-40 článků v kategorii:

Dvojice hlavních kategorií	Accuracy	Precission	Recall	F-measure
tv – nhl	0.994318181818	0.989583333333	1.0	0.994565217391
nhl movies	0.989898989899	0.990740740741	0.989898989899	0.989878076835
movies fasion	0.977272727273	0.974358974359	0.984848484848	0.978174603175
fasion education	0.993506493506	1.0	0.987012987013	0.993197278912
education dogs	1.0	1.0	1.0	1.0
dogs tv	1.0	1.0	1.0	1.0

Tabulka výsledků pro interval 40-30 článků v kategorii:

Dvojice hlavních kategorií	Accuracy	Precission	Recall	F-measure
tv – nhl	1.0	1.0	1.0	1.0
nhl – movies	1.0	1.0	1.0	1.0
movies – fasion	0.973611111111	0.952272727273	1.0	0.974937343358
fasion – education	1.0	1.0	1.0	1.0
education – dogs	0.986111111111	0.987654320988	0.986111111111	0.98605664488
dogs – tv	1.0	1.0	1.0	1.0

Tabulka výsledků pro interval 30-20 článků v kategorii:

Dvojice hlavních kategorií	Accuracy	Precission	Recall	F-measure
tv – nhl	0.996894409938	1.0	0.993788819876	0.996655518395
nhl – movies	0.979591836735	0.964285714286	1.0	0.980952380952
movies – fasion	0.972321428571	0.9875	0.958928571429	0.97
fasion – education	0.989795918367	1.0	0.979591836735	0.989010989011
education – dogs	0.980576441103	1.0	0.961152882206	0.976812661023
dogs – tv	1.0	1.0	1.0	1.0

Tabulka výsledků pro interval 20-10 článků v kategorii:

Dvojice hlavních kategorií	Accuracy	Precission	Recall	F-measure
tv – nhl	0.996111111111	0.994	1.0	0.996613756614
nhl – movies	1.0	1.0	1.0	1.0
movies – fasion	0.992156862745	1.0	0.98431372549	0.990849673203
fasion – education	0.977645502646	0.97037037037	0.994708994709	0.97993082755
education – dogs	0.980503144654	0.996855345912	0.964779874214	0.974624955757
dogs – tv	0.994252873563	1.0	0.988505747126	0.993103448276

3.2.2 Kontrola odlišností podkategorií ve stejné hlavní kategorii

Tento test spočíval v ověření zda bude bude klasifikátor schopen dostatečně rozlišit dvě podkategorie jedné hlavní kategorie.Například hlavní kategorie "NHL"a dvě podkategorie:"Toronto"a "Tampa". Klasifikují se vždy dvě sousední kategorie při seřazení kategorií od nejméně zastoupené kategorie po nejvíce zastoupenou.Výsledky toto testu jsou uvedeny v tabulce 3.3

Tabulka 3.3: Výsledky kontroly odlišnosti podkategorií ve stejné hlavní kategorii *Uváděné hodnoty jsou průměrnou hodnotou uvedené v prvních řádcích tabulek*

Tabulka výsledků pro interval 50-40 článků v kategorii:

Hlavních kategorie	Accuracy	Precission	Recall	F-measure
tv	1.0	1.0	1.0	1.0
nhl	0.845779220779	0.848045605188	0.839826839827	0.839184149184
movies	0.943181818182	0.925595238095	0.977272727273	0.947225672878
fasion	0.92196969697	0.912564102564	0.94696969697	0.925396825397
education	0.909090909091	1.0	0.818181818182	0.9
dogs[2]	–	–	–	–

Tabulka výsledků pro interval 40-30 článků v kategorii:

Hlavních kategorie	Accuracy	Precission	Recall	F-measure
tv	1.0	1.0	1.0	1.0
nhl	0.940476190476	0.94715007215	0.946103896104	0.945678708346
movies	1.0	1.0	1.0	1.0
fasion	0.895833333333	0.888888888889	0.962962962963	0.920634920635
education	0.785714285714	0.932773109244	0.746031746032	0.727472527473
dogs	0.886574074074	0.92962962963	0.866666666667	0.892787524366

Tabulka výsledků pro interval 30-20 článků v kategorii:

Hlavních kategorie	Accuracy	Precission	Recall	F-measure
tv	0.989648033126	0.994565217391	0.988354037267	0.990858416945
nhl	0.869047619048	0.877777777778	0.942857142857	0.905882352941
movies	0.883928571429	0.913034188034	0.921875	0.902292968469
fasion	0.928571428571	0.940079365079	0.942307692308	0.935629616738
education	0.915266106443	0.966314731021	0.911764705882	0.912753970366
dogs	0.966666666667	1.0	0.942857142857	0.969230769231

Tabulka výsledků pro interval 20-10 článků v kategorii:

Hlavních kategorie	Accuracy	Precission	Recall	F-measure
tv	0.98299086758	0.986497064579	0.9899543379	0.98685059233
nhl	0.988888888889	0.986666666667	1.0	0.992592592593
movies	0.9625	0.981473214286	0.963541666667	0.967473498723
fasion	0.932650273224	0.955451470206	0.953825136612	0.947137470908
education	0.893954248366	0.942741136859	0.922222222222	0.907653214401
dogs	0.92037037037	0.932892416226	0.946296296296	0.934833890389

Kapitola 4

Závěr

Cílem této práce bylo úvést základní problémy řešené při obecné klasifikaci a následná specializace těchto problémů při klasifikaci textu. Byly zde zmíněny techniky a metody používající se při zpracování textu, který se má klasifikovat. Pro lepší obrazek o tom, jak klasifikace probíhá zde byly popsány některé klasifikátory, které se při klasifikaci textu používají.

Stručně se zde pojednává o jazyku XML a nastiňuje se struktůra formátu RSS, aby si i neznalý čtenář tohoto formátu mohl udělat představu o jeho možnostech. Dále jsou zde probírány praktické problémy, které mohou vzniknout při tvorbě RSS čtečky a je zde ukázáno jejich možné řešení s upozorněním na výhody a nevýhody zvoleného řešení. Z implementačního hlediska jsou zde popsány možnosti knihovny NLTK, která slouží pro zpracování přirozeného jazyka. Je zde ukázán návrh modulu pro klasifikaci textů s objasněnými principy a upozorněním na možné vylepšení. Pro otestování použitelnosti klasifikátoru jsou zde probrány metody testování a přidány výsledky dvou testů použitého klasifikátoru na testovací sadě dokumentů.

Mezi další možnost rozšíření této práce patří určitě možnost automatické rozpoznání jazyku článku. Uživateli by tak odpadla nutnost specifikovat jazyk kanálu. Dále by učitě byla prospěšná možnost zavední aktivního učení klasifikátoru. Uživateli by pak byly nabídnuty položky, které jsou nejvíce vhodné pro natrénovaní klasifikátoru. Tím by bylo dosaženo lepších výsledků klasifikace. Zajímavým rozšířením by bylo určitě obohacení klasifikátoru o aplikační rozhraní umožňující používat vzdáleně sdílený klasifikátor a nebo jiné způsoby uložení klasifikátoru, například do databáze.

Literatura

[1] NLTK. [online], [cit. 17. 5. 2010].
URL http://www.nltk.org//

[2] Ranks.nl stopwords. [online], [cit. 17. 5. 2010].
URL http://www.ranks.nl/stopwords/

[3] Ch. D. Manning, P. Raghavan, H. Schütze: *Introduction to Information Retrieval*. Cambridge University Press, 2009, iSBN 0521865719.
URL http://nlp.stanford.edu/IR-book/information-retrieval-book.html

[4] D. A. Hull: Stemming Algorithms - A Case Study for Detailed Evaluation. [online], 1995, [cit. 17. 5. 2010].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.8310&rep=rep1&type=pdf

[5] Furnkranz, J.: A Study Using n-gram Features for Text Categorization. [online], [cit. 17. 5. 2010].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.133&rep=rep1&type=pdf

[6] Internet Content Syndication Council: Internet Content Syndication. [online], [cit. 17. 5. 2010].
URL http://www.internetcontentsyndication.org/downloads/whitepapers/content_creation.pdf

[7] J. Kosek: *XML pro každého*. Grada, 2000, iSBN 80-7169-860-1.

[8] R. O. Duda, P. E. Hart, D. G. Stork: *Pattern classification*. A Wiley-Interscience Publication, druhé vydání, 2001, iSBN 0 471 05669-3.

[9] S. Holzner, J. Šindelář: *RSS Automatické doručování obsahu vašich www stránek*. Computer Press, 2007, iSBN 978-80-251-1479-7.

[10] S.E. Robertson, K. S. J.: Simple, proven approaches to text retrieval. [online], 1994, [cit. 17. 5. 2010].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.57&rep=rep1&type=pdf

[11] W. B. Cavnar, J. M. Trenkle: N-Gram-Based Text Categorization. [online], [cit. 17. 5. 2010].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.3248&rep=rep1&type=pdf

[12] Özgür, A.: SUPERVISED AND UNSUPERVISED MACHINE LEARNING TECHNIQUES FOR TEXT DOCUMENT CATEGORIZATION. [online], 2002, [cit. 17. 5. 2010].

Příloha A

Obsah CD

Složka `sources` obsahuje výslednou ukázku čtečky i s modulem čtečky a klasifikátorem. Struktůra adresáře:

```
/sources/rss_reader_tk.py
/sources/classifier/
/sources/classifier/core.py
/sources/classifier/appIO.py
/sources/classifier/perzistence.py
/sources/classifier/test.py
/sources/classifier/stopwords/
/sources/classifier/stopwords/czech
/sources/classifier/stopwords/english

/sources/rss_module/
/sources/rss_module/rssmod.py
/sources/rss_module/webSyndParser.py
/sources/rss_module/SyndParser/
/sources/rss_module/SyndParser/aut.py
/sources/rss_module/SyndParser/rss2_conf.py
```

Příloha B

Manual

Po nainstalování knihovny nltk stačí spustit ukazkovou čtečku způsobem:
`python rss_reader_tk.py`

B.1 Instalace NLTK

Knihovna NLTK je dostupná ve většině linuxových distribucí a nebo na *http://nltk.org* kde je taky uveden návod k instalaci