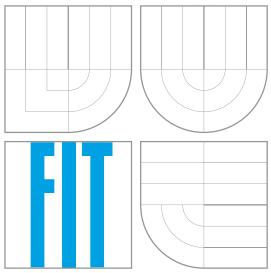
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLASIFIKACE DOKUMENTŮ PODLE TÉMATU

DOCUMENT TOPIC CLASSIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB ORAVEC

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2008

Abstrakt

Táto bakalárska práca sa zaoberá automatickou klasifikáciou dokumentov podľa témy a poskytuje stručný úvod do tejto oblasti výskumu. V prvej časti obsahuje prehľad základných postupov používaných v strojovom spracovaní prirodzeného jazyka s dôrazom na metódy klasfikácie textu. V ďalšej časti sa popisuje návrh a implementácia systému pre automatickú klasifikáciu dokumentov podľa témy. Posledná časť obsahuje informácie o testovaní vytvoreného systému vrátane vytvorenia testovacej sady a popisu štandardných metrík.

Kľúčové slová

klasifikácia dokumentov podľa témy, support vector machines, trénovacia sada, metriky

Abstract

This bachelor's thesis deals with automatic document topic classification and provides a brief introduction to this area of research. The first part contains summary of basic techniques used in natural language processing with emphasis on text classification methods. The next part describes concept and implementation of system for automatic document topic classification. The last part contains information about testing of created system including composition of testing set and standard metrics description.

Keywords

document topic classification, support vector machines, training set, metrics

Citácia

Jakub Oravec: Klasifikace dokumentů podle tématu, bakalářská práce, Brno, FIT VUT v Brně, 2008

Klasifikace dokumentů podle tématu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Doc. RNDr. Pavla Smrža Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jakub Oravec
11. mája 2008

Obsah

Úvod

S rozvojom internetu a informačných technológií v posledných rokoch sa človek dostáva do styku stále s väčším množstvom informácií v podobe elektronických dokumentov a tieto informácie potrebuje triediť podľa určitých kritérií. Na začiatku to bolo triedenie podľa metrík exaktne vyhodnotiteľných, napríklad dokumenty obsahujúce konkrétny výraz, wordovské dokumenty, obrázky s rozmermi 32x32 pixlov vhodné pre použitie ako ikonky a podobne.

S príchodom techník strojového učenia a výkonných techník automatizovanej klasifikácie sa však otvárajú dvere do klasifikácie podľa vágnejšie definovaných kritérií, ako sú napríklad subjektivita novinového príspevku, volebné preferencie autora politického blogu, riziko výskytu ochorenia podľa genetickej výbavy človeka alebo klasifikácia článkov podľa témy.

V poslednej dobe sa na internete objavuje množstvo knižníc a nástrojov pre rýchle a efektívne použitie výkonných klasifikačných techník. Tým sa automatická klasifikácia dostáva z výskumných ústavov medzi širokú verejnosť, čo umožnuje aj záujemcom s minimálnymi znalosťami problematiky vytvoriť v pomerne krátkom čase napríklad filter „zaujímavých“ jedál z on-line jedálnička školskej jedálne, či triedič došlej pošty.

Táto práca pojednáva o nasadení moderných klasifikačných techník v praxi. Vychádza z teoretických poznatkov získaných v rámci riešenia semestrálneho projektu (ISP). V kapitole 1 sú prebraté základné postupy používané pre prípravu textu na klasifikáciu spolu s prehľadom používaných klasifikačných metód. Sú spomenuté aj problémy a úskalia v realizácii jednotlivých techník. Kapitola 2 popisuje návrh a implementáciu systému pre klasifikáciu textu poďla témy a kapitola 3 oboznamuje so štandardnými metrikami používanými pre hodnotenie klasifikátorov, popisuje požiadavky na testovaciu sadu, prezentuje a diskutuje dosiahnuté výsledky systému popísaného v kapitole 2.

Kapitola 1

Klasifikácia dokumentov

Táto kapitola poskytuje stručný úvod do základných princípov a postupov klasifikácie dokumentov. V prvej časti sa venuje príprave textu dokumentu na klasifikáciu vrátane jeho prevodu do vektorovej podoby spracovávateľnej počítačom. Ďalšia časť 1.2 sa venuje používaným metódam automatizovanej klasifikácie, ich výhodám a nevýhodám. V poslednej časti 1.3 sú podrobnejšie popísané princípy uplatňované v metódach z rodiny support vector machines, ktorá bola použitá aj pre implementáciu systému pre klasifikáciu dokumentov podľa témy.

1.1 Príprava textu

Pred použitím akejkoľvek metódy klasifikácie dokumentov je potrebné text predpripraviť. V tejto časti sa budem venovať práve postupom prípravy textu na trénovanie alebo klasifikáciu.

Medzi základné používané postupy podľa [4] patria:

- tokenizácia,
- odstránenie stop slov,
- získavanie základného tvaru slov,
- vektorizácia.

1.1.1 Tokenizácia

Tokenizácia podľa [9] predstavuje transformáciu postupnosti bajtov na významové celky – najčastejšie slová, niekedy frázy. Ďalej je často výhodné nahradiť určité skupiny znakov definované napríklad regulárnym výrazom ich všeobecným popisom. Pri klasifikáci dokumentov asi nemá význam evidovať konkrétne telefónne číslo, podsatné je, že sa v texte vyskytlo *nejaké* telefónne číslo. Ďalšími príkladmi môžu byť roky, dátumy, chemické vzorce, ISBN, internetové adresy (web, mail,...), IP adresy a iné. Nahradzovať sa však nemusia len skupiny čísiel, [9] uvádza príklad, kde anglické slová končiace na *-rase* budú zrejme referovať na nejaký enzým.

V tejto časti sa často prevádzajú konverzie veľkých písmen na malé. Jednou z možností je previesť úplne všetky veľké písmená na malé, čím sa však môže stratiť určitá informácia. Ďalšou možnosťou je použiť slovník a niektoré slová ponechať s veľkým počiatočným písmenom (bush / Bush - prezident, urna / URNA - útvar rýchleho nasadenia).

Pri klasifikácii môžu činiť problém zložené slová v jazykoch ako je nemčina. Napríklad slovo *Prüfmittelfähigkeitsuntersuchung* (zisťovanie spôsobilosti meradiel) by malo z hľadiska klasifikácie väčší prínos v rozdelenej forme. Preto sa v rámci tokenizácie môže previesť aj rozdeľovanie zložených slov.

1.1.2 Odstránenenie stop slov

Ako vyplýva z [9], stop slovom sa rozumie slovo, ktoré nemá vplyv na to, do ktorej triedy bude dokument zaradený. Ide najmä o niektoré zámená, predložky, spojky a častice. Množina stop slov je špecifická pre konkrétny jazyk a oblasť záujmu. Napríklad pri klasifikácii podľa témy zrejme veľmi nezáleží na osobných zámenách, tie však môžu byť zdrojom cenných informácií pri klasifikácii podľa subjektivity (napríklad príspevky v novinách).

Zoznam stop slov sa väčšinou tvorí ručne (viď. [9]) tak, že sa odfiltrujú najčastejšie sa vyskytujúce slová a následne človek posúdi, ktoré z nich nemajú vplyv na rozhodovanie pri klasifikácii. Ich počet sa pri klasifikácii dokumentov typicky pohybuje v rádoch desiatok.

1.1.3 Stemming

Pri klasifikácii dokumentov podľa témy (a v ďalších oblastiach spracovania prirodzeného jazyka) je irelevantný tvar spracovávaných slov. Slovo popisujúce jednu skutočnosť alebo jav môže byť v texte použité v mnohých tvaroch. Avšak tvar slova zrejme nemá vplyv na kategóriu, do ktorej je dokument zaradený. Preto je potrebné pracovať s reprezentáciou pomenovania nezávisle na použitom tvare. Popis stemmingu a jeho metód v tejto časti bol prevzatý z [12], kde je možné dohľadať ďalšie informácie.

Stemming je proces získavania koreňa slova, pričom však nemusí ísť, a väčšinou ani nejde, o koreň zhodný s koreňom v linguistickom ponímaní. Cieľom je zaistiť, aby sa čo najviac slov odvodených zo spoločného základného tvaru namapovalo na jediné slovo – *stem*. Program alebo algoritmus prevádzajúci stemming sa nazýva *stemmer*.

Algoritmy pracujúce s odtŕhaním koncoviek

Prvý publikovaný popis stemmeru pochádza z roku 1968 od Julie Beth Lovins. Tento algoritmus je určený pre anglický jazyk a pri svojej činnosti používa 11 pravidiel pre postupné odstraňovanie sufixov slova. Štandardom sa stal algoritmus od Martina Portera z osemdesiatych rokov minulého storočia. Martin Porter neskôr vytvoril framework pre tvorbu stemmerov nazvaný *Snowball*.

Algoritmy „hrubou silou“

Ďalším možným prístupom je použitie slovníka, ktorý obsahuje preklady odvodených slov na ich koreň. Tento prístup pochopitelne vyžaduje veľké množstvo pamäťového priestoru. Jeho výhoda je však v tom, že dokáže nájsť správny koreň aj pre problematické slová ako napríklad: urážka na *cti* → *česť*. Tento postup sa niekedy kombinuje s postupmi založenými na odtŕhaní prípon, kde spracováva len slová, ktoré pri zmene tvaru menia aj základ.

Lematizácia

Lematizácia je sofistikovaný prístup k hľadaniu základných tvarov slov. Využíva sa znalosti syntaktických a morfologických pravidiel v danom jazyku pre určenie slovných druhov jednotlivých slov. Následne sa použijú pravidlá špecifické pre konkrétny slovný druh pre

získanie základného tvaru slova. V niektorých zdrojoch sa lematizácia považuje za jednu metódu stemmingu – [12], inde sa vyčleňuje ako samostatná oblasť – [9].

Význam použitia stemmingu

V [9] sa uvádza, že použitie stemmingu vo všeobecnosti zvyšuje hodnotu *recall* a znižuje hodnotu *precision* (viď. časť venovanú metrikám 3.1). Použitie stemmingu pre anglicky písané dokumenty nie je nevyhnutnosťou, pri malých objemoch trénovacích dát však môže zlepšiť výsledky. Toto tvrdenie ilustruje porovnanie uvedené v prílohe A. Pre jazyky s bohatou morfológiou je použitie základných tvarov slov účinným spôsobom ako redukovať veľkosť slovníka a znížiť požiadavky na objem trénovacích dát.

1.1.4 Vektorová reprezentácia dokumentov

V spracovaní dokumentov sa často používa ich vektorová reprezentácia, kde každá zložka vektora reprezentuje určitý term (viď [9]). Dokument je teda reprezentovaný ako vektor (niekedy sa hovorí o *bode* – danom vektorom a počiatkom súradnicovej sústavy) v mnohorozmernom priestore, kde platí, že vektory dokumentov obsahujúcich mnoho spoločných termov majú v tomto priestore menšie vzdialenosti. Naopak, vektory dvoch dokumentov, ktoré sú z hľadiska použitých výrazov úplne odlišné, budú mať svoju vzájomnú vzdialenosť väčšiu. Vytváranie zhlukov bodov obsahovo podobných dokumentov sa využíva pri klasifikácii bez učiteľa.

Podľa [9] existuje niekoľko možností prevodu dokumentu do vektorovej podoby. Triviálnym prípadom je, keď jednotlivé zložky vektora môžu nadobúdať len hodnoty 0 alebo 1 v závislosti od toho, či dokument príslušný term obsahuje, alebo nie. Ďalšou možnosťou je zohľadniť aj počet výskytov daného termu v dokumente. Potom je však potrebné vziať do úvahy aj dĺžku dokumentu napríklad normalizáciou vektora. Často sa využívajú aj rôzne štatistiky o celej trénovacej sade, napríklad inverzné dokumentové frekvencie (označované ako *TF-IDF*), keď sa termom vyskytujúcim sa vo veľkom množstve dokumentov priraďuje menšia váha ako termom, ktoré sa nachádzajú iba v úzkej skupine dokumentov. Porovnanie úspešnosti týchto metód možno nájsť v časti 3.3.1 na obrázku 3.3.

Je zrejmé, že každý dokument obsahuje len malý zlomok termov z termov vo všetkých spracovávaných dokumentoch. Toto vedie na vektory, ktoré majú väčšinu zložiek nulovú (resp. inú hodnotu reprezentujúcu neprítomnosť daného termu v dokumente). Pre praktické nasadenie sa preto používa reprezentácia pomocou *riedkych vektorov (sparse vectors)*.

Pri klasifikácii textu s vektorovou reprezentáciou dokumentov sa podľa [9] používajú normalizované vektory, majú teda veľkosť rovnú 1. V trojrozmernom priestore sada dokumentov potom vyzerá ako množina vektorov smerujúca na povrch jednotkovej gule. Rôzne triedy dokumentov vytínajú na povrchu tejto gule rôzne časti.

1.2 Používané metódy

V tejto časti budú predstavené niektoré metódy používané pri klasifikácii dokumentov spolu s ich výhodami a nevýhodami. Ich popis vychádza z [9].

1.2.1 Naivný Bayesov klasifikátor

Naivný Bayesov klasifikátor patrí medzi metódy strojového učenia s učiteľom. Pracuje s využitím Bayesovho teorému (tiež Bayesovo pravidlo) o podmienenej pravdepodobnos-

ti. Predpokladá nezávislosť výskytu jednotlivých termov v dokumente (pravdepodobnosť výskytu slova je nezávislá od pravdepodobnosti výskytu ostatných slov), čo všeobecne v reálnych dokumentoch neplatí. S uvážením tohoto zjednodušenia však táto metóda dosahuje nečakane dobré výsledky.

Výhody

- Veľmi rýchly,
- jednoduchý na imlementáciu.

Nevýhody

- Má dobré výsledky len pre určité úlohy,
- silný vplyv dĺžky dokumentu na výsledky trénovania.

Táto metóda je vhodná napríklad na filtrovanie nevyžiadanej elektronickej pošty.

1.2.2 Umelé neurónové siete

Umelé neurónové siete sú zložené z umelých neurónov inšpirovaných živými neurónmi, ktoré sú rôzne prepojené. Používajú sa *perceptróny*, ktoré sa však obmedzujú len na lineárne separovateľné úlohy. Zástupcom neurónových sietí učiacich sa s učiteľom sú neurónové *siete so spätným šírením (backpropagation neural networks)*. V procese trénovania sa iteračne upravujú váhy vstupov umelých neurónov tak, aby výsledok pre všetky trénovacie príklady čo najviac zodpovedal očakávanému výsledku.

Neurónové siete sa používajú aj pre učenie bez učiteľa. Typickými zástupcami tohoto prístupu sú *Kohonenove* siete a *Hopfieldove siete*.

Výhody

- Dobre sa vysporiadavajú s nelineárnymi vzťahmi v dátach.

Nevýhody

- Trénovanie má často veľké časové nároky,
- sieť sa dokáže ľahko preučiť (nachádza súvislosti tam, kde nie sú).

1.2.3 Rocchio

Tento algoritmus pracuje na princípe hľadania ťažísk tried v trénovacej sade pomocou priemerovania vektorov obsiahnutých v triede. Deliacou rovinou je potom množina bodov, ktoré majú rovnakú vzdialenosť od dvoch ťažísk. Ťažisko nájdeme ako:

$$\boldsymbol{\mu_c} = \frac{0}{|D_c|} \sum_{d \in D_c} \boldsymbol{v}(d)$$

Kde D_c je množina trénovacích dokumentov patriacich do triedy c a $\boldsymbol{v}(d)$ označuje normalizovaný vektor reprezentujúci dokument d. Kritériom pre zaradenie dokumentu do tej–ktorej triedy je potom euklidovská vzdialenosť od jednotlivých ťažísk.

Výhody

- Jednoduchá a rýchla metóda,
- prirodzene klasifikuje do viacerých tried,
- po natrénovaní postačuje skladovať informácie o ťažisku každej triedy.

Nevýhody

- Pri potrebe rozšíriť trénovaciu sadu, je potrebné znovu prepočítať ťažiská.

1.2.4 kNN – k nearest neighbor

kNN je pravdepodobne najjednoduchšia metóda strojového učenia. Používa vektorovú reprezentáciu a predpokladá, že body ležiace blízko seba patria do rovnakej triedy. Fáza trénovania pozostáva len z uloženia vektorových reprezentácií dokumentov spolu s ich zaradením do triedy.

Vo fáze klasifikácie potom o zaradení dokumentu do určitej triedy rozhoduje väčšina z jeho $k \in \mathbb{N}$ najbližších susedov. Pri klasifikácii do dvoch tried je vhodné voliť k nepárne, aby sa predišlo rovnosti hlasov pri rozhodovaní. Určitým vylepšením môže byť priradenie váhy hlasom susedov podľa ich vzdialenosti tak, aby zaradenie blízkych susedov zavážilo viac ako zaradenie vzdialenejších.

Výhody

- Veľmi jednoduchá metóda,
- pri rozšírení trénovacej sady postačuje pridať nové príklady,
- pre klasifikáciu dokumentov dosahuje veľmi dobré výsledky.

Nevýhody

- Nutnosť skladovať celú trénovaciu sadu,
- pomalšia klasifikácia (v prospech rýchleho trénovania).

1.2.5 Support vector machines

Táto časť bola zaradená len pre úplnosť prehľadu metód. Keďže som túto metódu použil pre realizáciu systému pre klasifikáciu dokumentov požadovaného zadaním, bude sa jej podrobnejšie venovať časť 1.3.

Metódy patriace do skupiny support vector machines sú metódy strojového učenia s učiteľom používané pre klasifikáciu a regresiu. Niekedy sa tiež nazývajú klasifikátormi s *maximalizáciou okraja (maximum margin classifiers)*. Používajú vektorovú reprezentáciu objektov (v tomto prípade dokumentov) v mnohorozmernom priestore (high-dimensional feature space).

Principiálne tieto metódy v procese trénovania hľadajú v n-rozmernom priestore $(n-1)$-rozmernú nadrovinu oddeľujúcu *pozitívne príklady* od *negatívnych* tak, aby bola maximalizovaná jej vzdialenosť (margin) od najbližších trénovacích príkladov. Nájdená nadrovina

je potom daná len týmito najbližšími vektormi. Tieto vektory deliacu nadrovinu akoby podopierali – odtiaľ názov *support vectors*.

Pre klasifikáciu neznámeho objektu postačuje zistiť, na ktorej strane nájdenej nadroviny sa skúmaný objekt nachádza.

Výhody

- Veľmi dobré výsledky,
- široké možnosti voľby parametrov a ďalšieho vylepšovania,
- po natrénovaní postačuje skladovať tzv. model (viď. časť 1.3) narozdiel od metódy kNN.

Nevýhody

- Náročná implementácia,
- pri zmene trénovacej sady (pridanie príkladov) je potrebné opätovné natrénovanie.

1.3 Support vector machines

V tejto časti budú podrobnejšie popísané princípy využívané v support vector machines (ďalej SVM) a ich využitie pri klasifikácii dokumentov. Ďalej bude prebratá varianta umožňujúca klasifikáciu do viacerých tried (multiclass klasifikácia).

1.3.1 História

[1] a [13] zhrňujú históriu SVM:

- 1963 – Vladimír Vapnik predstavil algoritmus pre hľadanie lineárnej nadroviny.
- 1968 – Smith prekonal problém zašumených dát pomocou tzv. *slack variables*.
- 1992 – B. Boser, I. Guyon a V. Vapnik navrhli nelineárny klasifikátor využívajúci tzv. kernel triky.
- 1979 – v súvislosti s rozvojom téorie štatistického učenia sa začína hovoriť o metódach SVM.
- 1995 – Cortes a Vapnik predstavili klasifikátor s mäkkým okrajom (soft margin). V tom istom roku Vapnik v [11] rozšíril tento algoritmus pre *regresiu* (SVR – support vector regresion).

V [2] sa hovorí, že metódy z rodiny SVM sa dlhú dobu veľmi nepoužívali, najmä pre zložitosť implementácie. Avšak koncom deväťdesiatych rokov sa objavilo niekoľko projektov využívajúcich práve metódy SVM, čím sa značne zviditelnili. SVM sa nasadzovali pre rozpoznávanie výrazu tváre, rukou písaných čísel, výskumy v oblasti genetiky a mnoho iných. Za oblasť klasifikácie textu to bol Thorsten Joachims, ktorý v roku 1997 publikoval článok [7].

1.3.2 Princíp

Táto časť čerpá z [9], kde je možné dohľadať detaily fungovania a optimalizácie SVM.

Cieľom SVM je nájsť nadrovinu oddeľujúcu pozitívne a negatívne príklady tak, aby prechádzala čo najdaľej od najbližších vektorov trénovacej sady. Preto sa SVM označuje ako metóda s maximalizáciou okraja. Na parametre deliacej nadroviny majú teda vplyv len vektory, ktoré sú k nej najbližšie, pretože ovplyvňujú maximalizáciu okraja v procese trénovania. Tieto vektory sa nazývajú *podporné vektory (support vectors)*, pretože deliacu nadrovinu akoby podopierajú (odtiaľ názov support vector machines). Metódam SVM tým pádom stačí pre natrénovanie pomerne malá trénovacia sada pozostávajúca z čo najnegatívnejších pozitívnych príkladov a čo napozitívnejších negatívnych príkladov. Tento prístup sa podobá ľudskému uvažovaniu tým, že potrebuje vedieť len „čo ešte patrí a čo už nepatrí“ do danej triedy.

Z matematického hľadiska je deliaca rovina daná jej normálovým vektorom $\boldsymbol{w}$ (tento je kolmý na nadrovinu) a posunutím b. Vektoru $\boldsymbol{w}$ sa hovorí aj váhový vektor. Deliacej nadrovine potom patria všetky body $\boldsymbol{x}$, ktoré spĺňajú podmienku:

$$\boldsymbol{w} \cdot \boldsymbol{x} = b$$

Majme množinu trénovacích bodov v tvare $D = \{(\boldsymbol{x_i}, y_i)\}$, kde $\boldsymbol{x_i}$ označuje i-ty trénovací bod a $y_i \in \{-1, 1\}$ svojou hodnotou označuje tento trénovací bod ako negatívny alebo pozitívny príklad. Klasifikácia je potom jednoduchá:

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

Funkčný okraj príkladu $\boldsymbol{x_i}$ z triedy y_i je daný ako $y_i(\boldsymbol{w} \cdot x_i + b)$. Tento funkčný okraj je závislý na veľkosti vektora $\boldsymbol{w}$. *Funkčný okraj celej dátovej sady* je potom dvojnásobok funkčného okraja trénovacieho bodu s minimálnym funkčným okrajom z celej trénovacej sady D.

Geometrickým okrajom príkladu $\boldsymbol{x_i}$ patriaceho do triedy y_i je $r = y_i \frac{\boldsymbol{w} \cdot \boldsymbol{x_i} + b}{|\boldsymbol{w}|}$. Veľkosť vektora $\boldsymbol{w}$ vystupujúca v menovateli zlomku zabezpečí, že veľkosť tohoto vektora nebude mať vplyv na veľkosť geometrického okraja. *Geometrickým okrajom klasifikátora* sa rozumie maximálna šírka pásu oddeľujúceho podporné vektory oboch tried.

Zvyčajne sa požaduje, aby funkčný okraj všetkých dátových bodov v trénovacej sade bol minimálne 1 a bol rovný 1 aspoň pre jeden dátový bod:

$$\forall(\boldsymbol{x}, y) \in D\colon \quad y(\boldsymbol{w} \cdot \boldsymbol{x} + b) \geq 1 \quad \wedge \quad \exists(\boldsymbol{x}, y) \in D\colon \quad y(\boldsymbol{w} \cdot \boldsymbol{x} + b) = 1$$

Keď má existovať aspoň jeden bod, ktorého funkčný okraj je 1, potom veľkosť geometrického okraja klasifikátora je $\rho = 2/|\boldsymbol{w}|$. Pri trénovaní je cieľom dosiahnuť čo najväčší geometrický okraj, to znamená minimalizovať veľkosť vektora $\boldsymbol{w}$.

Tieto úvahy vedú na optimalizačnú úlohu:
Nájsť $\boldsymbol{w}$ a b také, aby:

- $|\boldsymbol{w}|$ bola minimálna a
- $\forall(\boldsymbol{x}, y) \in D\colon \quad y(\boldsymbol{w} \cdot \boldsymbol{x} + b) \geq 1$

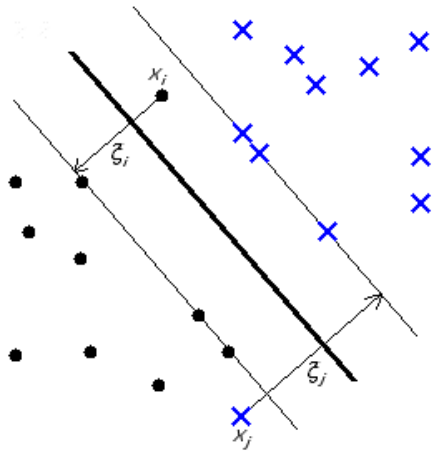
Pre optimalizáciu takýchto kvadratických problémov existujú matematické postupy, vysvetlenie ktorých presahuje rámec tejto práce.

1.3.3 Trénovanie s mäkkým okrajom – soft margin

Často sa stáva, že v trénovacej sade sa vyskytnú zle zaradené príklady (pozitívny príklad označený ako negatívny alebo naopak), prípadne príklady, ktoré z hľadiska vektorovej reprezentácie nie sú tak jednoznačne rozlíšiteľné ako sú rozlíšiteľné pre človeka. Pre tieto prípady odvodili Cortes a Vapnik v roku 1995 klasifikátor s „mäkkým okrajom“ (soft margin), kde sa určité nezrovanlosti tolerujú. Nasledujúci text popisuje trénovanie s mäkkým okrajom čerpajúc z [9].

V procese trénovania sa hľadá nadrovina s čo najširším okrajom, ale je dovolené urobiť niekoľko chýb. Každý dokument na nesprávnej strane deliacej nadroviny má potom priradenú cenu podľa jeho vzdialenosti od nadroviny. Cieľom je nájsť nadrovinu tak, aby mala čo najširší okraj a zároveň bola minimalizovaná cena za nesprávne klasifikované trénovacie vektory.

Pre účely vyhodnotenia ceny vektorov na nesprávnej strane deliacej nadroviny boli zavedené takzvané *volné premenné (slack variables)* (viď. obrázok 1.1). Tieto predstavujú vzdialenosť nesprávne oddeleného vektora od okraja triedy, do ktorej mal byť zaradený. Parametrom pri trénovaní je hodnota c, ktorá udáva, či je prioritou získať čo najširší okraj (malá hodnota c), alebo čo najmenšie odchýlky zle zaradených vektorov od ich správnej polohy (veľká hodnota c). Udáva vlastne váhu penalty za zle zaradený objekt.



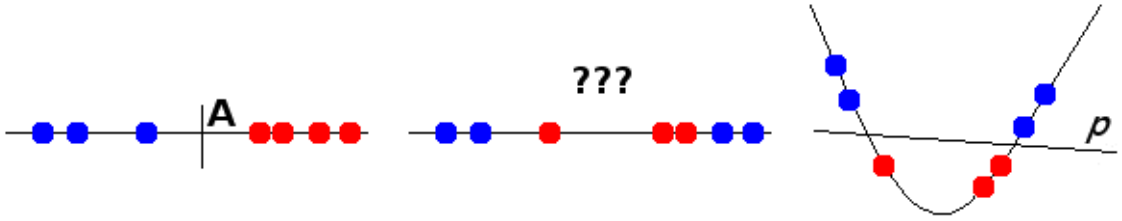
Obrázok 1.1: K výpočtu ceny zle zaradených vektorov

1.3.4 Nelineárna klasifikácia – „kernel trik“

Pri niektorých úlohách nie je možné dve triedy separovať lineárnou nadrovinou. V roku 1992 B.Boser, I.Guyon a V. Vapnik ukázali, že pre takéto problémy je možné použiť pomerne dobre známu techniku kernel triku.

Skúmajme veličinu „teplota“. Nízke teploty označíme popisom „chlad“ a ostatné teploty popisom „teplo“. Ide o jednorozmernú veličinu, a teda chceme oddeliť „chlad“ od „teplo“ (1−1)-rozmernou nadrovinou. V tomto veľmi zjednodušenom príklade sa teda pojem deliaca nadrovina redukuje na deliaci bod. Situácia je znázornená na obrázku 1.2 vľavo (modrou farbou sú znázornené hodnoty považované za „chlad“ a červenou hodnoty považované za „teplo“). Bod A oddeľuje „chlad“ od „tepla“. Tieto dve množiny hodnôt sú lineárne separovatelné.

Skúsme klasifikovať teploty inak. Obrázok 1.2 v strede ukazuje teploty klasifikované ako „príjemné“ (červenou) a „nepríjemné“ (modrou). Tieto dve triedy teplôt zrejme nie je možné lienárne oddeliť. Riešenie spočíva v použití kernel triku. Dáta premietneme do viacrozmerného priestoru s vhodnou transformáciou ako ukazuje obrázok 1.2 vpravo. Teraz sú dáta reprezentované v 2-rozmernom priestore a teda deliaca nadrovina bude $(2-1)$-rozmerná, teda priamka (p). Takto upravené dáta už sú lineárne separovateľné. Použitie



Obrázok 1.2: Ilustrácia kernel triku

kernel triku teda spočíva v transformácii vstupných vektorov do viacrozmerného priestoru pomocou vhodného kernelu. Pozitívne a negatívne príklady sa následne oddeľujú stále lineárnou nadrovinnou, ale v inom priestore. Vysvetlenie podstaty kernel triku a popis používaných kernelov sú prevzaté z [2].

Vektory sa aj v trénovacom aj klasifikačnom algoritme vyskytujú len v skalárnom súčine $\boldsymbol{x_i} \cdot \boldsymbol{x_j}$. Po zobrazení $\Phi : \boldsymbol{R}^d \mapsto \mathcal{H}$ do viacrozmerného Euklidovského priestoru $\mathcal{H}$ by uvedený skalárny súčin prešiel do tvaru $\Phi(\boldsymbol{x_i}) \cdot \Phi(\boldsymbol{x_j})$. Po zavedení funkcie K (kernelu) tak, že $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \Phi(\boldsymbol{x_i}) \cdot \Phi(\boldsymbol{x_j})$ nepotrebujeme vôbec vedieť ako zobrazenie Φ vyzerá a ani explicitne vyčíslovať zložky vektora po zobrazení do $\mathcal{H}$.

V praxi sa bežne používa niekoľko kernelov:

Polynomický kernel stupňa p

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \cdot \boldsymbol{x} \cdot \boldsymbol{y} + c)^p$$

Gaussian radial basis

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\frac{||\boldsymbol{x}-\boldsymbol{y}||^2}{2\sigma^2}}$$

V tomto prípade je $\mathcal{H}$ nekonečnerozmerným priestorom. Tento kernel poskytuje možnosť odeliť dve triedy objektov, keď vektorové reprezentácie objektov jednej triedy akoby obklopovali vektory objektov triedy druhej. Gaussian radial basis kernel dokáže takýchto zhlukov oddeliť ľubovolné množstvo.

Problém môže činiť voľba parametru σ. Pri nevhodnej voľbe môže dôjsť k úkazu v literatúre označovanému ako *efekt vianočného stromčeka*, keď sa pri trénovaní vytvorí veľa roztrúsených ostrovčekov obsahujúcich len niekoľko málo (1-2) pozitívnych resp. negatívnych príkladov (vianočné gule). Toto predstavuje preučenie klasifikátora.

Sigmoid

$$K(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\gamma \cdot \boldsymbol{x} \cdot \boldsymbol{y} + c)$$

Tento druh kernelu predstavuje určitý druh dvojvrstvovej sigmoidálnej neurónovej siete.

Lineárny

$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$$

V tomto prípade nejde o žiadny kernel trik (nahradzuje skalárny súčin tým istým skalárnym súčinom). Systémy používajúce SVM používajú toto značenie kvôli jednotnosti napriek tomu, že vlastne žiaden kernel nie je potrebné použiť. Pri klasifikácii dokumentov podľa témy sa ukázalo ako najvhodnejšie použiť práve lineány klasifikátor.

1.3.5 Klasifikácia do viacerých tried

Pri použití klasifikátorov klasifikujúcich do dvoch tried (resp. klasifikácie typu: dokument patrí / nepatrí do triedy C) nastáva problém pri potrebe klasifikácie do viacerých tried.

V [9] sa uvádzajú dva druhy viactriednej klasifikácie. Jedným je klasifikácia, kde objekt môže patriť do viacerých tried súčasne, alebo dokonca nemusí patriť do žiadnej triedy. Tento druh klasifikácie sa v literatúre označuje ako *any-of klasifikácia*. Opakom je klasifikácia kde každý dokument musí byť zaradený práve do jednej triedy (tzv. *one-of klasifikácia*).

Any-of klasifikácia

V prípade, keď objekt (dokument) môže byť zaradený do viacerých tried súčasne, je riešenie pomerne jednoduché. Stačí natrénovať samostatný klasifikátor pre každú triedu. Ako pozitívne príklady sa použijú objekty patriace do tejto triedy a ako negatívne sa použijú dokumenty, ktoré do danej triedy určite nepatria. Skúmaný objekt sa klasifikuje postupne všetkými klasifikátormi a zaradí sa do tried podľa výsledku príslušných klasifikátorov.

Problémom pri tomto prístupe býva voľba vhodných negatívnych príkladov. Často býva väčší problém nájsť jeden dobrý negatívny príklad ako desať pozitívnych.

One-of klasifikácia

Hoci bežne objekty môžu patriť do viacerých tried súčasne, v automatickej klasifikácii sa často problém zjednodušuje na klasifikáciu do jedinej triedy, čo sa mnohokrát javí ako dostačujúce.

Riešenie spočíva v použití samostatného klasifikátora pre každú triedu. Ako pozitívne príklady sa použijú príklady, ktoré do danej triedy patria, ako negatívne príklady sa použijú zvyšné príklady. Neznámy objekt sa potom skúma všetkými klasifikátormi a zaradí sa práve do tej triedy, ktorej klasifikátor rozpoznal skúmaný objekt ako pozitívny s najväčším funkčným okrajom (objekty blízko deliacej nadroviny môžu byť klasifikované chybne).

Multiclass SVM

Popis multiclass SVM možno nájsť v [9] odkiaľ čerpala aj táto časť. SVM klasifikuje prirodzene do dvoch tried. Jedným z možných (a bežných) spôsobov riešenia viactriednej klasifikácie je natrénovanie viacerých klasifikátorov, pričom každý odlišuje jednu triedu od všetkých ostatných – tzv. *one-versus-rest*. Počet potrebných klasifikátorov je teda rovný počtu tried, do ktorých sa má klasifikovať.

Druhým možným prístupom je vytvorenie množiny klasifikátorov, kde každý klasifikátor rozlišuje len dve triedy medzi sebou. Tento postup sa označuje ako *one-versus-one*. Toto síce vedie na väčšie množstvo klasifikátorov, avšak tieto pracujú s podstatne menšou trénovacou sadou, čo sa prejaví na rýchlosti trénovania.

Multiclass SVM používa elegantnejší prístup. Spočíva vo vytvorení dvojtriedneho klasifikátora klasifikujúceho vektory v tvare $\Phi(\boldsymbol{x}, y)$, kde $\boldsymbol{x}$ je vektor trénovacieho príkladu a y je označenie triedy, do ktorej príklad $\boldsymbol{x}$ patrí. Vo fáze klasifikácie sa pre skúmaný vektor x odvodí množina vektorov $\Phi(\boldsymbol{x}, y) \quad y \in C$, kde C je množina všetkých tried. Skúmaný vektor sa potom zaradí do triedy podľa toho, ktorý odvodený vektor je klasifikovaný ako pozitívny s najväčším okrajom. Formálne:

$$y = \arg\max_{y'} \boldsymbol{w} \cdot \Phi(\boldsymbol{x}, y')$$

Naposledy menovaný prístup má veľkú výhodu v tom, že pre trénovanie klasifikátora nie je potrebné hľadať žiadne negatívne príklady. Ako negatívne príklady si poslúžia zástupcovia tried, ktoré sú vo vektorovom priestore najbližšie a sú teda najpravdepodobnejšie zameniteľné.

1.3.6 Existujúce implementácie

Táto časť poskytuje stručný prehľad voľne dostupných implementácií SVM vo forme knižníc. Tieto a ďalšie možno nájsť napríklad v [13].

Knižnice od Thorstena Joachimsa

Thorsten Joachims v rámci svojho výskumu uvolňuje súbor knižníc pre experimentovanie s SVM. Súbor obsahuje knižnice pre klasifikáciu do dvoch tried – svmlight. Ďalej knižnicu pre klasifikáciu štruktúrovaných dát – svmstruct, ktorého súčasťou je aj svmmulti pre viactriednu klasifikáciu.

Tieto knižnice sú šírené pod vlastnou licenciou, kde autor žiada byť informovaný o výskume, na ktorý boli knižnice použité atď.

Knižnica libSVM

Knižnica libsvm je publikovaná pod pomerne voľnou licenciou pánmi Chih-Chung Changom a Chih-Jen Linom. Obsahuje nástroje pre trénovanie, klasifikáciu a normalizáciu vektorov. Použitím rôznych prepínačov umožnuje voliť medzi klasifikáciou a regresiou a taktiež voliť použitý kernel vrátane parametrov.

Súčasťou balíka sú rozhrania pre jazyky Python a Java, ako aj jednoduchý program (SVM Toy) umožnujúci vizualizáciu klasifikácie v trojrozmernom priestore. To všetko vo verzii pre systémy Windows aj Linux. Niektoré distibúcie Linux-u majú túto knižnicu dokonca zahrnutú v repozitároch. Viac informácií možno získať na stránkach [3].

Knižnica libLinear

Knižnica liblinear od autorov knižnice libsvm (viď vyššie) implementuje techniky SVM a je optimalizovaná pre lineárnu klasifikáciu. Dosahuje podstatne lepšie časy pre trénovanie a použije menej pamäte. Je veľmi vhodná pre problematiku klasifikácie textu a iné úlohy s veľkým počtom rozmerov vstupných vektorov, kde sa využíva práve lineárna klasifikácia.

dlib C++

Táto knižnica je zameraná na portovateľnosť a jednoduchosť použitia. Je šírená pod Boost Software License. Viac informácií možno nájsť na `http://dclib.sourceforge.net/`.

Kapitola 2

Systém pre klasifikáciu dokumentov

Táto kapitola sa zaoberá návrhom a implementáciou systému pre klasifikáciu dokumentov podľa témy. Cieľom je navrhnúť a implementovať systém, ktorý by správne klasifikoval čo najviac dokumentov a nepožadoval od užívateľa hlbšie znalosti problematiky. Inými slovami: aby fungoval čo najlepšie aj s implicitnými parametrami.

2.1 Analýza

Zadanie požaduje realizáciu systému pre klasifikáciu dokumentov podľa témy. Z uvedeného v kapitole 1 vyplýva, že konkrétna realizácia systému a hlavne časti pripravujúcej dokumenty na klasifikáciu je silne závislá na viacerých faktoroch ako je formát vstupných dát, jazyk dokumentov, počet tried, do ktorých sa má klasifikovať (2 alebo viac), oblasť reálneho sveta, ktoré dokumenty pokrývajú (vplyv hlavne na voľbu zovšeobecňovaných reťazcov a prevodu na malé znaky v procese tokenizácie – viď časť 1.1.1). Toto činí realizáciu univerzálneho systému na klasifikáciu dokumentov podľa témy pomerne obtiažnou.

Na druhej strane, pre klasifikáciu podľa témy postačuje vyhodnocovať výskyt slov (resp. termov) v dokumente, na rozdiel od klasifikácie podľa subjektivity alebo postoja autora, kde je potrebné brať do úvahy aj rôzne syntaktické vzťahy v texte a ďalšie skutočnosti.

Táto časť analyzuje jednotlivé možné prístupy k riešeniu spomínaných problémov.

2.1.1 Vstupné dáta

Pojem „dokument“ je značne obšírny. Za dokument môže byť považovaný textový súbor, súbor vo formáte pdf, za dokument sa môže považovať aj obrázok a mnoho iných. Keďže ide o klasifikáciu podľa témy, obmedzíme sa chápaním dokumentu na textové dokumenty.

I tak tu ale zostáva priestor pre diskusiu o formáte vstupných dát, ktoré sa majú klasifikovať. Jednou možnosťou je model, kde bude každý dokument v jednom súbore, ďalšou možnosťou je jeden súbor obsahujúci množinu dokumentov. Napríklad súbor vo formáte XML, prípadne databáza obsahujúca viaceré dokumenty spolu s ďalšími metadátami. Za ďalší možný prístup možno považovať klasifikáciu jednotlivých častí dokumentu samostatne v prípade, že sa jedná o rozsiahlejší dokument. Niekedy sa môže požadovať klasifikácia jednotlivých dokumentov zabalených v archíve.

2.1.2 Načítanie dát

Pre načítanie vstupných dát zo súborov v rôznych formátoch sa ponúka možnosť použiť rôzne voľne šírené utility používané na systémoch Unixového typu ako napríklad `antiword` pre načítanie dát vo formáte programu Microsoft Word, `pdftotext` pre načítanie dokumentov vo formáte pdf, `html2text` pre načítanie súborov s webovými stránkami.

2.1.3 Tokenizácia

Tokenizácia vo svojej najjednoduchšej podobe pozostáva z prevodu textu (napríklad) na malé písmená, jeho rozdelenia na jednotlivé výrazy (slová, čísla) podľa bielych znakov, prípadne odstránenia stop slov.

Ďalšou činnosťou, ktorá môže byť požadovaná v procese tokenizácie, je zjednotenie kódovaní znakov rôznych dokumentov, keďže klasifikované dokumenty môžu byť napísané v rôznych kódovaniach. Pre anglicky písaný text je tento problém minimálny, avšak pre jazyky s bohatou diakritikou je potrebné vziať do úvahy aj tento činiteľ.

Úspešnosť procesu klasifikácie môže zlepšiť zavedenie náhrad určitých postupností znakov za ich všeobecnejšie pomenovanie (viď časť 1.1.1). Niekedy však môže dôjsť k chybnej identifikácii reťazca, napríklad reťazec „10:13“ môže reprezentovať čas, výsledok zápasu v hokeji, alebo zmiešavací pomer. Ďalším problémom môže byť skutočnosť, že niektoré údaje sa v rôznych kultúrach zapisujú rôzne. Za všetky snáď postačí spomenúť rôzne formáty dátumov.

Ďalšou možnosťou ako zvýšiť úspešnosť klasifikácie je zaviesť zoznam slov, v ktorých sa nebudú nahradzovať veľké písmená za malé. Avšak vzhľadom na malú početnosť slov s významom závislým na spôsobe písania veľkých a malých písmen je možné, že takáto úprava zvýši úspešnosť klasifikácie len nepatrne a naopak prispeje k zvýšeniu množstva času potrebného na prípravu textu (každé slovo obsahujúce veľké písmená sa musí preveriť, či sa nenachádza v slovníku).

2.1.4 Stop slová

Odstránením stop slov je možné hlavne zmenšiť veľkosť spracovávaného textu a tiež počet rozmerov vektora reprezentujúceho dokument. Experimentálne sa ukázalo, že odstránením stop slov sa zmenšila veľkosť spracovávaného textu asi o 20 %, čo sa už môže pozitívne prejaviť na čase potrebnom pre ďalšie spracovanie.

V porovnaní klasifikácie podľa témy s klasifikáciou podľa iných kritérií má klasifikácia podľa témy výhodu v možnosti použitia podstatne väčšieho množstva stop slov. Určité obmedzenie pri návrhu systému klasifikujúceho dokumenty s odstraňovaním stop slov predstavuje to, že pre rôzne jazyky sa množiny stop slov pochopiteľne líšia.

2.1.5 Stemming

Stemming je najproblematickejšou časťou celej prípravy textu na klasifikáciu. Problém spočíva hlavne v tom, že každý jazyk má svoje špecifické pavidlá pre odvodzovanie slov, teda aj získavanie koreňov slov vyžaduje pre rôzne jazyky odlišné algoritmy. Podľa [9] je pre angličtinu v prípade dostatočne veľkej trénovacej sady možné stemming úplne vynechať. Pre morfologicky bohaté jazyky, ako je čeština alebo slovenčina, je použitie stemmingu výhodné (pre menšie trénovacie sady a kratšie dokumenty táto výhoda prechádza až do potreby).

Určitým kompromisom by mohlo byť degradovanie stemmingu ako systematického odstraňovania koncoviek slova až na postup, keď by sa jednoducho ignorovalo niekoľko posledných znakov slova (vyjadrené napríklad percentuálne s odstupňovaním podľa celkovej dĺžky slova).

2.1.6 Vhodné metódy

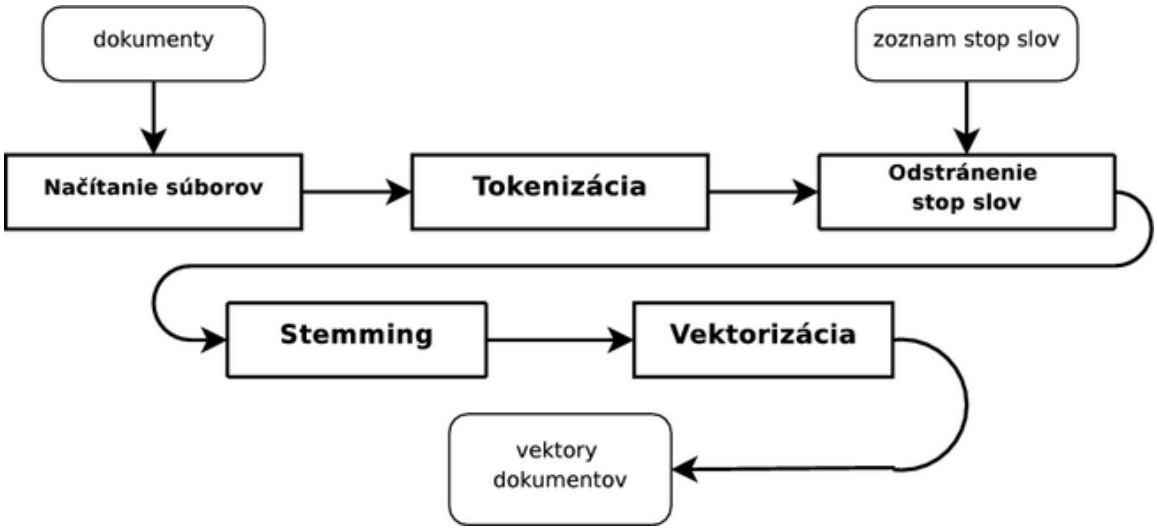
V časti 1.2 boli prezentované rôzne metódy klasifikácie používané pre klasifikáciu dokumentov. Keďže sa požaduje čo najvyššia úspešnosť klasifikácie, do úvahy pripadajú metódy kNN a SVM, ktoré podľa viacerých výskumov (napríklad [7] – porovnaj [14]) vykazujú najlepšie výsledky. Metóda kNN má istú výhodu v jednoduchosti implementácie, SVM zas v rýchlosti klasifikácie. V [10] a [6] sa uvádza, že pre úlohy ako klasifikácia dokumetnov, je výhodnejšie použiť lineárny klasifikátor, ktorý dosahuje rovnaké výsledky ako nelineárny, avšak trénovanie prebieha podstatne rýchlejšie. To hovorí pre použitie lineárnej verzie SVM.

2.2 Návrh

Táto časť sa zaoberá návrhom systému pre klasifikáciu dokumentov. Výsledky jeho testovania je potom možné nájsť v kapitole 3.

Návrh bol prevedený s ohľadom na požiadavky zadania a prevedenú analýzu. Z uvedeného vyplynula značná variabilita požiadavkov na predprípravu trénovacieho aj klasifikovaného textu, preto som systém navrhol ako *modulárny*, aby bola zabezpečená možnosť jednoduchej zámeny nevyhovujúcej časti za inú, pre požadovaný účel vhodnejšiu, časť.

Na obrázku 2.1 je znázornený postup pri predspracovaní dokumentov. V prípade predspracovania pre účely trénovania sa k vektoru každého dokumentu poznačí aj jeho správne priradenie do niektorej z tried.



Obrázok 2.1: Príprava dokumentov na trénovanie resp. klasifikáciu

2.2.1 Príprava textu

Tokenizácia

Modul obstarávajúci tokenizáciu som navrhol tak, aby prevádzal všetky veľké písmená na malé bez použitia slovníka s výnimkami (viď 1.1.1) z dôvodu uvedeného v časti venovanej analýze 2.1.3. Ďalej tento modul nahradzuje niektoré často sa vyskytujúce zoskupenia znakov za ich všeobecný popis. Sú to najmä:

- dátumy vo formáte s oddeľovacím znakom „.“,
- údaje o roku (4 číslice začínajúce na „20“ alebo „19“),
- ceny uvedené v dolároch,
- webové adresy.

Stemming

Z dôvodu neistého prínosu stemmingu (viď [9]) a časovej ako aj implementačnej náročnosti lematizéru, navrhujem použiť pomerne jednoduchý *Porterov stemmer*, ktorý by mal pre daný účel postačovať. Každopádne, implementácia bude musieť myslieť aj na prípadnú požiadavku tento stemmer nahradiť iným a úpravu maximálne uľahčiť.

Vektorizácia

Ako už bolo uvedené v časti 1.1.4, existuje niekoľko možností prevodu textu na jeho vektorovú reprezentáciu. Najpoužívanejší postup je pomocou inverzných dokumentových frekvencií, ako je uvedené v [7] alebo [9]. Podľa experimentov, ktorých výsledky sú prezentované v časti 3.3.1, sa ukázalo, že pri použití normalizovaných vektorov odvodených len z počtu jednotlivých termov v dokumente, je možné aspoň pre malé trénovacie sady (približne 6 000 dokumentov) dosiahnuť lepšie, alebo aspoň porovnateľné výsledky ako s použitím inverzných dokumentových frekvencií (viď obrázok 3.3). Preto navrhujem ponechať v systéme možnosť prepínať medzi týmito dvoma metódami.

2.2.2 Klasifikácia

Klasifikátor je navrhnutý tak, aby dokázal roztriediť súbory prítomné v adresári súborového systému do podadresárov podľa príslušnosti k triede určenej klasifikátorom. Každej triede, do ktorej sa má klasifikovať, prislúcha jeden podadresár. Takéto usporiadanie umožnuje klasifikovať aj do celej stromovej štruktúry tried tak, že sa v každom adresári spustí proces trénovania nad všetkými prítomnými podadresármi. Proces sa následne rekurzívne opakuje pre všetky podadresáre, pokým sú nejaké k dispozícii.

Ako bolo naznačené v časti 2.1, pre použitie v klasifikátore som zvolil metódu SVM vo verzii pre lineárnu viactriednu klasifikáciu (multiclass SVM) pre svoje výborné výsledky a rýchlosť klasifikácie. V tomto prípade je potrebné vhodne zvoliť konštantu c používanú pri trénovaní (viď časť 1.3.3). Zlatou strednou cestou bude zvoliť $c = 1$, keď má šírka okraja rovnakú váhu ako veľkosť chyby pri zle klasifikovanom príklade.

2.3 Implementácia

V tejto časti je popísaný systém klasifikujúci dokumenty tak, ako je implementovaný spolu so stručným popisom použitia a možností úprav.

2.3.1 Použité nástroje

Systém bol vyvíjaný na platforme Linux, avšak nič nebráni tomu, aby bol použitý aj na iných POSIX-ových systémoch. Klasifikačný systém je implementovaný s použitím nasledovných jazykov:

- C – implementačný jazyk nástrojov libSVM,
- Python – použitý pre implementáciu skriptov pre načítanie html súborov a prípravu textu. Taktiež je to implementačný jazyk modulu Porterovho stemmeru.
- Jazyk interprétu *Bash*, použitý pre spojenie jednotlivých modulov do celku.

Implementácia klasifikátora pomocou SVM v princípe nie je zložitá, avšak pre reálne nasadenie je potrebné použiť pomerne náročné optimalizačné techniky, ktoré siahajú za rámec bakalárskej práce. Z tohoto dôvodu som pre implementovaný systém použil hotový nástroj libSVM (viď časť 1.3.6), ktorý je dostupný na stránkach [3]. Balík obsahuje okrem iných aj nástroje `svm-train` a `svm-predict`, pomocou ktorých sa prevádza trénovanie SVM a klasifikácia pomocou SVM. Na niektorých linuxových distribúciách je možné nainštalovať tento balík priamo z repozitárov.

Ďalším modulom, ktorý nebol vytváraný v rámci tejto práce je stemmer. Použil som Porterov stemmer v implementácii [5]. Programátorské rozhranie obsahuje jedinú funkciu, takže práca s ním je nanajvýš jednoduchá.

Naproti tomu, modul prevádzajúci html súbory na text musel byť realizovaný špeciálne pre použitie klasifikačným systémom, hoci je dostupný nástroj `html2text`. Tento však výstupný text nežiadane formátuje a navyše sa nevyhovujúcim spôsobom vysporiadava so syntaktickými chybami vo vstupnom súbore. Preto bol vytvorený jednoduchý skript `readhtml` v jazyku Python, ktorý vypíše obsah webovej stránky s ignorovaním skriptov, hlavičky a komentárov. V prípade chyby vo vstupných dátach poškodenú časť ignoruje, záhlási chybu a pokračuje ďalej v činnosti. Vynechanie neveľkej časti textu pre toto nasadenie nemá príliš veľký význam.

2.3.2 Popis modulov

file2text

Modul `file2text` slúži pre načítanie textových dát zo vstupných súborov. Očakáva názov súboru, ktorý má byť prevedený na text, ako parameter. Ak je tento súbor vo formáte pdf, doc alebo html, je tento prevedený do textovej podoby a vypísaný na štandardný výstup, pričom sa ignoruje akékoľvek formátovanie, komentáre a skripty (v prípade html).

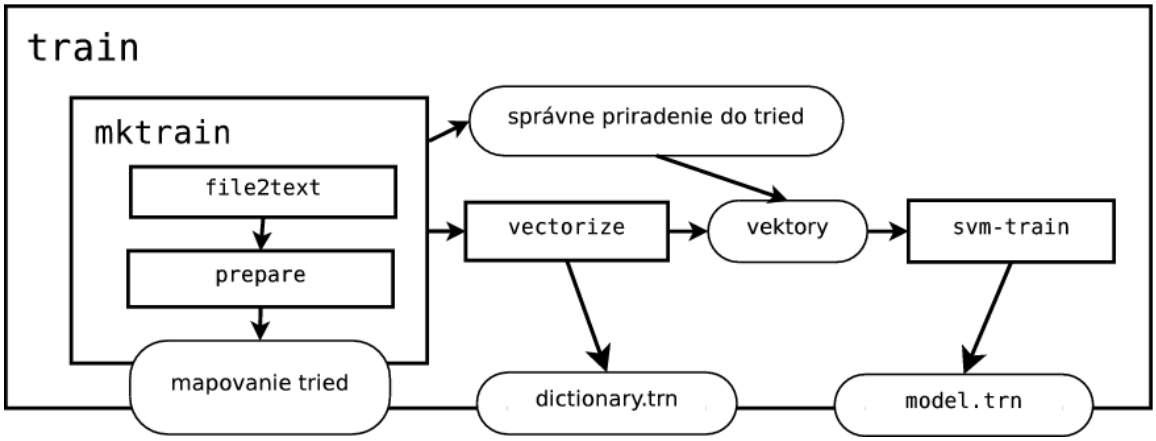
tokenize

Nástroj `tokenize` prevádza odstránenie interpunkcie a stop slov zo vstupného textu, nahradenie určitých postupností znakov za ich všeobecné pomenovanie (roky, dátumy, webové adresy a podobne) a taktiež stemming pomocou Porterovho algoritmu. Zoznam stop slov

sa získava zo súboru daného nepovinným parametrom. Ak nie je tento parameter udaný, považuje sa zoznam stop slov za prázdny.

train

Skript `train` prevádza trénovanie SVM. V podstate sa postará o načítanie dokumentov určených na trénovanie, pripraví ich na trénovanie a spustí trénovací proces. Toto trénovanie prebieha rekurzívne vo všetkých podadresároch, ktoré obsahujú ďalšie podadresáre (teda pokiaľ je do čoho dokumenty triediť). Činnosť a štruktúra tohoto modulu je zachytená na obrázku 2.2.



Obrázok 2.2: Štruktúra modulu `train`. Na obrázku nie je pre prehľadnosť zahrnutý prepočet inverzných dokumentových frekvencií v prípade použitia prepínača `-idf`

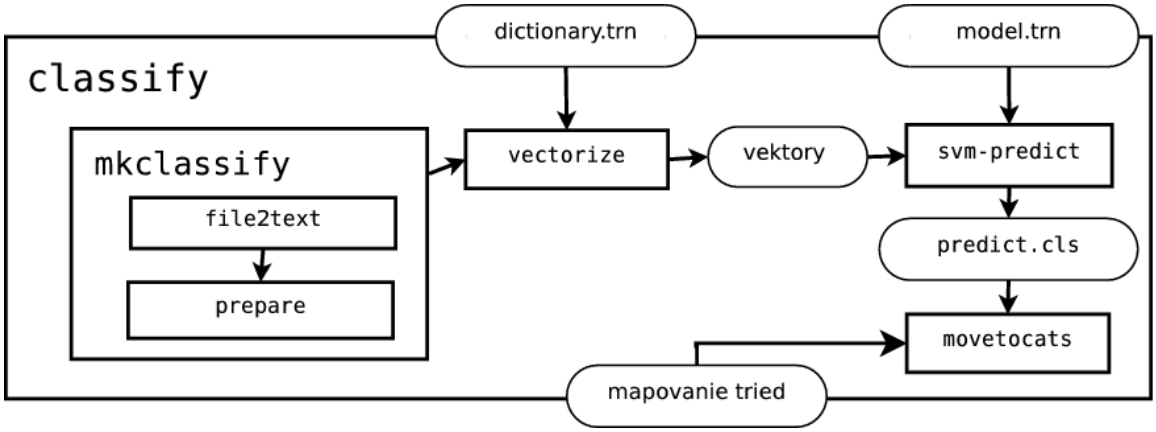
Skript rozpoznáva prepínač `-idf`, ktorý indikuje, že sa má použiť TF-IDF vektorová reprezentácia dokumentov. Implicitne sa použije reprezentácia pomocou normalizovaných vektorov zložených z počtu výskytov jednotlivých termov v dokumente.

Výsledkom vykonania tohoto skriptu sú súbory obsahujúce slovník, zoznam kategórií, natrénovaný model a súbor s vektormi dokumentov trénovacej sady. V prípade použitia prepínača `-idf` sa ešte vytvorí súbor obsahujúci informácie o dokumentových frekvenciách jednotlivých termov. Všetky súbory vytvorené týmto modulom majú príponu `.trn`.

classify

Modul `classify` prevádza samotnú klasifikáciu. Načíta a pripraví dokumenty z aktuálneho adresára a klasifikuje ich pomocou modelu prítomného v tom istom adresári. Klasifikované súbory následne presunie do podadresára určeného výsledkom klasifikácie. Potom sa klasikácia rekurzívne spustí v každom podadresári, ktorý obsahuje ďalšie podadresáre. Takýmto spôsobom sa klasifikované dokumenty postupne popresúvajú až do najnižšej úrovne hierarchie tried. Činnosť tohoto modulu je znázornená na obrázku 2.3.

Tento skript taktiež pozná prepínač `-idf`, ktorý spôsobuje použitie reprezentácie pomocou inverzných dokumentových frekvencií (nie je zobrazené na obrázku). V tomto prípade je nutné aj trénovanie previesť s prepínačom `-idf`, aby sa vytvoril súbor s dokumentvými frekvenciami jednotlivých termov.



Obrázok 2.3: Štruktúra modulu `classify`. V obrázku nie je pre prehľadnosť zahrnutý prepočet inverzných dokumentových frekvencií v prípade použitia prepínača `-idf`

2.3.3 Formát dát

Systém pracuje s rôznymi druhmi dát. Ide najmä o zoznamy (zoznam stop slov, slovník, príslušnosť dokumentov k triedam atď.) a o sady vektorov určené pre trénovanie, resp. klasifikáciu. Pre uchovávanie textových aj číselný údajov som volil formáty čitateľné človekom.

Zoznamy

Zoznamy ukladané vo forme súborov sú reprezetnované postupnosťou hodnôt oddelených prechodom na ďalší riadok. Takto formátované súbory sú ľahko čitateľné pre človeka aj pre programy, ktoré ich používajú a je možné ich ďalej ručne spracovávať programami ako sú `sed`, `awk`, `paste` a iné.

Vektory dokumentov

Pri štúdiu problematiky a dokumentácie dostupných nástrojov sa ukázalo, že existuje akýsi zažitý štandard pre ukladanie vektorových reprezentácií do súboru, kde sa využíva zápis pomocou riedkych vektorov – zapisujú sa teda len nenulové hodnoty. Formát vektora popisujúceho jeden dokument je nasledovný:

```
<class> <feature1>:<value1> <feature2>:<value2> ... <featureN>:<valueN>
```

kde

<**class**> je označenie skutočnej príslušnosti do triedy. Nadobúda celočíselné hodnoty a mapovanie na názvy tried je teda potrebné realizovať inak. Táto položka má zmysel pochopiteľne len pri trénovaní a testovaní systému. V prípade reálneho použitia, keď sa klasifikujú neznáme dokumenty, je hodnota `<class>` irelevantná a môže byť nastavená napríklad na hodnou 0.

<**featureI**> označuje poradové číslo zložky vektora, ktorej sa bude týkať hodnota uvedená za symbolom „dvojbodka“. `<featureI>` je celé číslo > 0. Hodnoty zložiek vektora musia byť uvedené v poradí s rastúcimi číslami zložiek vektora.

<**valueI**> udáva hodnotu zložky `<featureI>` vektora dokumentu. Je to celé číslo, alebo číslo s plávajúcou desatinnou čiarkou.

N – počet nenulových zložiek vektora.

Súbor obsahujúci celú sadu dokumentov potom obsahuje ľubovoľný počet riadkov v tomto formáte – pre každý dokument v sade jeden riadok.

2.4 Použitie a obmedzenia

Použitie vo svojej najzákladnejšej podobe spočíva v spustení programov `train` a následne `classify` v adresári obsahujúcom nezaradené dokumenty a podadresáre reprezentujúce jednotlivé triedy s trénovacími príkladmi. Nezaradené dokumenty budú v procese klasifikácie zaradené do tried. Viac o použití systému je možné nájsť v prílohe B.

Obmedzenia použitia systému vychádzajú najmä z jazykovej závislosti niektorých fáz predspracovania dokumentov. Ide teda o stemming a generalizáciu termov v rámci tokenizácie. Ako obmedzenie môžu byť videné časové nároky trénovania a klasifikácie pri spracovávaní veľkých objemov dát.

2.5 Možnosti úprav

Jednou z možností úprav je zámena klasifikačného nástroja – vymeniť libSVM, napríklad za libLinear. Oba nástroje majú rovnaké rozhranie, takže postačí zameniť názvy programov v skriptoch `train` a `classify`. Pri použití libLinear je možné dosiahnuť výrazne rýchlejšie trénovanie aj klasifikáciu, avšak dosahuje mierne odlišné výsledky (viď [6]).

Ďalšou možnosťou úpravy je rozšírenie množiny generalizovaných výrazov pomocou tokenizéra. Možno doplniť napríklad ďalšie formáty dátumov, názvy mesiacov, politických strán, chemických vzorcov, športového žargónu a mnoho iných.

V neposlednom rade je možné experimentovať s množinou stop slov a použitým stemmerom. Zámenou stemmera je možné použiť systém aj pre iné jazyky ako je angličtina.

Kapitola 3

Testovanie

Táto kapitola popisuje štandardné metriky používané pri vyhodnocovaní systémov klasifikujúcich dokumenty. Ďalej budú prezentované požiadavky na trénovaciu sadu spolu s popisom vytvorenej sady a výsledkami dosiahnutými systémom implementovaným podľa popisu uvedeného v kapitole 2.

3.1 Štandardné metriky

V tejto časti budú popísané štandardné metriky používané pri vyhodnocovaní systémov klasifikujúcich text. Popis metrík a postup vyhodnocovania vychádza z publikácie [9].

Pre označovanie výsledku klasifikácie pri testovaní sa používa tabuľka 3.1 zobrazujúca štyri možné výsledky testu. V tejto tabuľke **tp** (true positive) označuje dokument správne označený ako pozitívny – patriaci do kategórie; a **tn** (true negative) označuje dokument správne označený ako negatívny – nepatriaci do kategórie. Čím lepšie klasifikátor pracuje, tým sú počty takto označených testovacích dokumentov vyššie na úkor počtu dokumentov označených zvyšnými dvoma značkami. Naopak, neuspešnú klasifikáciu označujú značky **fn** (false negative) – dokument nesprávne označený klasifikátorom ako nepatriaci do kategórie a **fp** (false positive) – dokument klasifikátorom nesprávne označený za patriaci do triedy. Čím lepšie klasifikátor pracuje, tým je počet dokumentov s týmito značkami menší.

		klasifikácia	
		+	−
skutočnosť	+	**tp**	**fn**
	−	**fp**	**tn**

Tabuľka 3.1: Štyri možné výsledky testu klasifikácie

Metriky vyhodnocované pri testovaní klasifikačných systémov vychádzajú z počtov testovacích príkladov označených týmito značkami. Preto sa v ďalšom texte označeniami **tp**, **tn**, **fp** resp. **fn** budú myslieť počty dokumentov s týmto označením.

Nasledovný zoznam zhŕňa štandardne používané metriky.

- *Správnosť (accuracy)* – pomer správne zaradených objektov k ich celkovému počtu, určený ako $A = \frac{tp+tn}{tp+tn+fp+fn}$. Táto metrika je z pohľadu používateľa asi najzaujímavejšia, pretože hovorí o tom, koľko dokumentov systém správne zaradil. Ideálna hodnota je 1.

- *Presnosť (precision)* – pomer správne zaradených pozitívnych objektov k počtu označených za pozitívne. Určí sa ako $P = \frac{tp}{tp+fp}$. Táto hodnota hovorí o tom, koľko dokumentov určených ako pozitívne je skutočne pozitívnych.
- *Citlivosť (recall)* – pomer správne zaradených pozitívnych objektov k počtu skutočne pozitívnych objektov. Vyjadruje teda schopnosť hľadať pozitívne príklady. Vypočíta sa ako $R = \frac{tp}{tp+fn}$.
- *Chyba (error)* – hodnota odvodená od hodnoty accuracy: $E = 1 - A$. Niekedy sa používa namiesto hodnoty *správnosť*.
- *Matica zámen (confusion matrix)* – hovorí o tom, koľko dokumentov jednej skutočnej triedy bolo klasifikátorom zaradených do ktorej triedy – ideálne všetky do správnej triedy a do ostatných tried žiadne.
- *Závislosť úspešnosti od veľkosti trénovacej* sady umožňuje odhadnúť, koľko dokumentov je potrebné použiť pre trénovanie, aby sa pri klasifikácii dosiahli uspokojivé výsledky.
- *ROC krivka* nepriamo odráža vzťah medzi citlivosťou a presnosťou. V extrémnom prípade klasifikátora, ktorý bude všetky objekty klasifikovať ako pozitívne, zaiste dosiahneme vysokú citlivosť (našiel všetky pozitívne objeky), avšak presnosť bude nízka (z objektov klasifikovaných ako pozitívne je len málo skutočne pozitívnych). Sklon k takémuto správaniu možno odhaliť práve pomocou ROC krivky.

3.1.1 Matica zámen – confusion matrix

Matica zámen dáva komplexný pohľad na výsledok testovania klasifikátora. Je možné z nej určiť hodnoty A, P a R pre každú triedu, navyše je možné veľmi jednoducho pozorovať, ktoré triedy činia klasifikátoru najväčšie problémy, teda triedy, ktoré sa najčastejšie medzi sebou zamieňajú.

Pre ukážku je v tabuľke 3.2 uvedená matica zámen prevzatá z [9]. Hodnoty mimo hlavnej diagonály hovoria o nesprávne zaradených dokumentoch. Napríklad je možne vidieť, že všetky dokumenty z kategórie *interest* boli nesprávne zaradené do kategórie *money-fx*. Taktiež je možné vyčítať, že často dochádza k zámenám v rámci troch poľnohospodárskych tried (*wheat*, *corn* a *grain*) a v rámci troch finančných tried (*money-fx*, *trade* a *interest*).

priradená trieda / skutočná trieda	money-fx	trade	interest	wheat	corn	grain
money-fx	95	0	10	0	0	0
trade	1	1	90	0	1	0
interest	13	0	0	0	0	0
wheat	0	0	1	34	3	7
corn	1	0	2	13	26	5
grain	0	0	2	14	5	10

Tabuľka 3.2: Ukážka matice zámen prevzatá z [9].

3.2 Testovacia sada

Táto časť sa zaoberá zostavením trénovacej sady pre otestovanie a vyhodnotenie systému pre klasifikáciu dokumentov popísaného v kapitole 2.

3.2.1 Požiadavky na testovaciu sadu

Cieľom je vytvoriť testovaciu sadu zodpovedajúcu reálnym potrebám, avšak takú, aby odhaľovala aj slabé miesta systému. Ide hlavne o voľbu rozdelenia do tried. V testovacej sade Reuters-21578 môžeme napríklad vidieť triedy „corn“, „grain“ a „wheat“, ktoré ukážu schopnosť systému rozlíšiť pomerne podobné dokumenty z poľnohospodárskej oblasti.

Aby testovanie, pokiaľ možno, čo najlepšie preverilo implementovaný systém, navrhol som pre zostavenie testovacej sady nasledovné kritériá:

- testovacia sada musí obsahovať dostatočné množstvo dokumentov (minimálne 5 000),
- musí obsahovať minimálne 1 000 dokumentov určených pre testovanie, aby bolo možné vyhodnotiť úspešnosť s krokom maximálne desatiny percenta,
- mala by byť hierarchicky členená,
- mala by zahrňovať kategórie s rôznym počtom dokumentov, ako aj kategórie s porovnateľným počtom dokumentov,
- mala by obsahovať triedy dokumentov, ktoré sú podobné, ako aj úplne odlišné,
- dokumenty by nemalo byť možné správne zaradiť podľa výskytu jediného slova (všetky dokumenty v kategógii „Hudba“ obsahujú slovo „music“, ...).

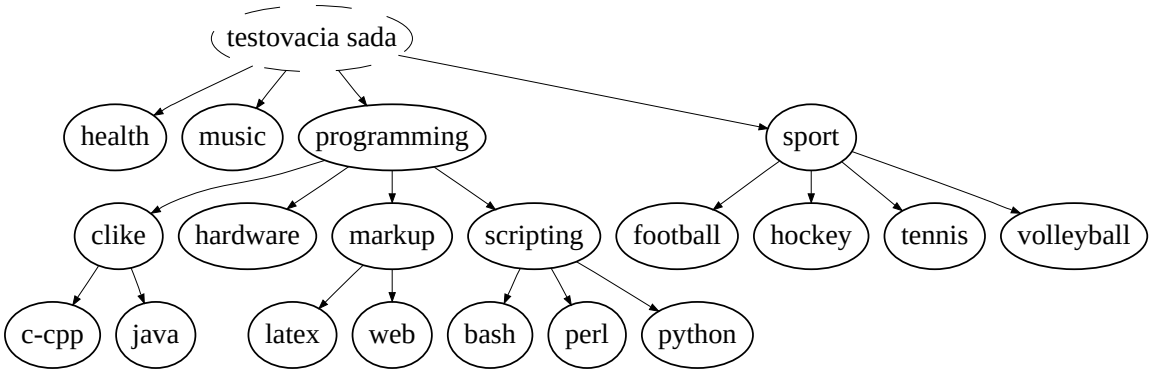
3.2.2 Popis sady

Pri vytváraní sady som sa snažil zvoliť hierarchiu tried tak, aby zodpovedala rozdeľovaniu človekom – používateľom. Triedy v jednej úrovni stromu tried teda nie sú delené „príliš jemne“. Vychádzal som z toho, že človek ručne triediaci dokumenty na svojom disku s najväčšou pravdepodobnosťou nebude voliť delenie dokumentov o športe do tried podľa toho, či ide o futbalový škandál, futbalový výsledok, zranenie futbalistu, alebo dokumenty o podplácaní rozhodcov, hoci aj takéto delenie je možné. Volil som teda rodelenie pomerne hrubé a s ohľadom na to, aby som otestoval schopnosť systému klasifikovať do viacerých úrovní tried, je hierarchia tried viacúrovňová. Štruktúra testovacej sady je znázornená na obrázku 3.1.

Hoci systém dokáže pracovať aj s dokumentami vo formáte pdf a doc, trénovacia sada pozostáva najmä z dokumentov vo formáte html, obsahuje však aj niekoľko málo pdf súborov. Ako dokumenty som použil webové stránky písané v angličtine verejne dostupné na internete. Každý dokument prešiel ručnou klasifikáciou a bol tak zaradený do triedy, o ktorej si človek – klasifikátor myslel, že najlepšie popisuje daný dokument. Od systému budeme požadovať, aby zaradil dokumenty z časti sady určenej na testovanie do rovnakých tried, ako ich zaradil človek.

Popis tried

Nasledujúci prehľad uvádza popis jednotlivých tried vo vytvorenej testovacej sade dokumentov (viď obrázok 3.1).



Obrázok 3.1: Štruktúra testovacej sady

health – zahŕňa články o chorobách (od chrípky až po rakovinu) a liečebných postupoch, očkovaniach, starostlivosti o chrup, vitamínoch, chudnutí, diétach, zdravom stravovaní a životnom štýle. Dokumenty o zdravom životnom štýle často hovoria o rôznych športoch, preto očakávam, že budú klasifikátorom zamieňané s dokumentami z kategórie „sport“.

music – obsahuje dokumenty o hudobných skupinách, koncertoch, predaji vstupeniek na koncerty, ďalej blogy opisujúce účasť na hudobnom podujatí, recenzie nových albumov a iné.

programming – táto kategória obsahuje články s témou programovania a ďalej sa rozvetvuje na niekoľko podtried:

- **clike** – pozostáva z dvoch podtried: *c-cpp* a *java*. *c-cpp* obsahuje dokumenty o programovaní v jazykoch C a C++. Do triedy *java* boli zaradené dokumenty o programovaní v jazyku Java a niekoľko dokumentov vygenerovaných pomocou nástroja `javadoc`.
- **hardware** – táto kategória pozostáva z dokumetnov o programovaní hardvéru – vstavané systémy, mikrokontroléry, FPGA a iné.
- **markup** – obsahuje dokumenty o značkovacích jazykoch. V kategórii *latex* sú to dokumenty o práci so systémom LaTeX a v kategórii *web* dokumenty o webdizajne, html značkách a podobne.
- **scripting** – pozostáva z troch podtried: *bash*, *perl* a *python*, ktoré som vybral ako zástupcov skriptovacích jazykov.

sport – zahrnuje dokumenty so športovou témou. Podtriedy *football*, *hockey*, *tennis* a *volleyball* zaiste nepokrývajú celé spektrum športov, avšak pre testovacie účely bohato postačujú. Obsahujú dokumenty o odohraných, aj plánovaných zápasoch, blogy fanúšikov, správy o zraneniach hráčov a podplácaní rozhodcov, stránky profesionálnych, aj školských klubov, tabuľky s výsledkami a ďalšie.

Sada obsahuje spolu 5 090 dokumentov rozdelených na časť určenú na trénovanie klasifikátora (3 811 dokumentov) a časť určenú na jeho testovanie (1 279 dokumentov). Tabuľka 3.5 ukazuje počty dokumentov v testovacej sade, aj s ich rozdelením na trénovacie a testovacie príklady.

Trieda	Trénovanie	Testovanie	Trieda	Trénovanie	Testovanie
health	722	268	bash	63	15
music	402	120	perl	89	20
c-cpp	207	73	python	159	46
java	234	84	football	203	97
hardware	494	169	hockey	214	66
latex	340	86	tennis	227	66
web	234	98	volleyball	223	71

Tabuľka 3.3: Rozdelenie dokumentov vo vytvorenej testovacej sade.

Napriek tomu, že sa jednotlivé triedy javia ako značne odlišné, je tomu tak len pre ľudského pozorovateľa. Webové stránky, ktoré boli použité ako testovacie dokumenty, však predstavujú pomerne zašumené dáta. Toto je spôsobené prítomnosťou ďalších informácií popri zaujímavom texte. Takýmito rušivými elementami sú najmä prvky navigácie, reklamné pruhy, novinky a ďalšie informácie, ktoré človek akosi prirodzene nezahrnuje do určenia témy dokumentu. Ďalším nežiaducim prvkom sú chybové správy obsiahnuté na stránke. Napríklad hlásenie o chybe v PHP skripte si ľudský klasifikátor pri zostavovaní trénovacej sady nemusí vždy všimnúť a text chybovej správy môže negatívne ovlyvniť automatickú klasifikáciu.

Nasledujúci prehľad uvádza predpokladané kolidujúce skupiny dokumentov vo vytvorenej testovacej sade:

- programovanie vstavaných systémov v jazyku C/C++ (trieda *hardware*) a „klasické“ programovanie v kategórii *programming*,
- články o vitamíne C (trieda *health*) a programovanie v jazyku C,
- stránky prezentujúce relaxačnú a aerobikovú hudbu môže systém zaradiť medzi dokumenty o zdraví,
- príklady s radami pre trénerov rôznych športov sa môžu značne podobať,
- články o hudobnom vybavení (ozvučovacia technika a elektronické hudobné nástroje) a kategória *hardware*.

Reuters-21578

Kvôli porovnaniu výsledkov s odbornými publikáciami som sa rozhodol použiť, okrem mnou vytvorenej sady, aj štandardnú sadu používanú pri výskumoch po celom svete. Z pomedzi množstva používaných sád som vybral už spomínanú sadu Reuters-21578, resp. desať tried obsahujúcich najviac dokumentov z tejto sady. Tento postup bol použitý napríklad v [7] alebo [14]. Táto sada obsahuje 21 578 správ a článkov zaradených do tried podľa témy, krajiny, spomenutých ľudí a iných. V [9] sa o nej hovorí ako o jednej z najpoužívanejších sád pre klasifikáciu textu. Ten istý zdroj hovorí, že sa najčastejšie používa rozdelenie sady na trénovacie a testovacie dokumenty „ModApte“. Sadu možno získať na stránkach [8], kde sa nachádza aj podrobný popis sady vrátane rozdelenia „ModApte“.

Túto sadu som použil na vyhodnotenie úspešnosti klasifikácie, kvôli možnosti porovnať výsledok s odbornými publikáciami. Ďalej bola sada Reuters-21578 použitá pre vyhodnote-

nie závislosti úspešnosti klasifikácie od veľkosti trénovacej sady z dôvodu väčšieho objemu dát, ako má k dispozícii testovacia sada vytvorená v rámci tejto práce.

3.3 Výsledky

Pre zostavenie matice zámen bola použitá testovacia sada zostavená v rámci tejto práce (viď časť 3.2.2). Výsledky sú uvedené v tabuľke 3.4. Táto tabuľka ukazuje, do ktorej triedy boli zaradené dokumenty rôznych skutočných tried. Možno pozorovať niektoré nečakané zámeny, napríklad dokumenty o futbale a hokeji zaradené do kategórie venovanej hudbe, alebo dokumenty z triedy *latex* zaradené do triedy *c-cpp*.

		Trieda priradená klasifikátorom													
		health	music	c-cpp	java	hardware	latex	web	bash	perl	python	football	hockey	tennis	volleyball
Skutočná trieda	health	**258**	3			2	3	2							
	music		**113**			5		1			1				
	c-cpp		1	**68**	2	1					1				
	java			1	**77**	5		1							
	hardware	2		2	1	**165**		1							
	latex	3		10	5	4	**64**		1		2				
	web	2	1	3	10	4		**75**	1		3				
	bash							2	**13**						
	perl				1	1				**18**					
	python			1	1	2		1			**41**				
	football		3	1		2						**83**	3		4
	hockey		3		1	2		1				1	**53**		5
	tennis			1		1						1		**63**	
	volleyball		1					1							**69**

Tabuľka 3.4: Matica zámen získaná testovaním na vytvorenej testovacej sade. Prázdne hodnoty predstavujú nulovú hodnotu.

Trieda	*Accuracy*	*Precision*	*Recall*	Trieda	*Accuracy*	*Precision*	*Recall*
health	99.06	99.23	96.27	bash	99.69	86.67	86.67
music	98.51	90.40	94.17	perl	**99.84**	**100.00**	90.00
c-cpp	98.12	**78.16**	93.15	python	99.06	85.42	89.13
java	97.81	78.57	91.67	football	98.75	97.65	85.57
hardware	**97.42**	85.05	**97.63**	hockey	98.75	94.64	80.30
latex	98.05	95.52	**74.42**	tennis	99.77	**100.00**	95.45
web	**97.42**	88.24	76.53	volleyball	99.14	88.46	97.18

Tabuľka 3.5: Výsledky získané testovaním na sade vytvorenej v rámci tejto práce. Maximálne a minimálne hodnoty sú vysádzané hrubým písmom. Uvedené hodnoty sú v percentách.

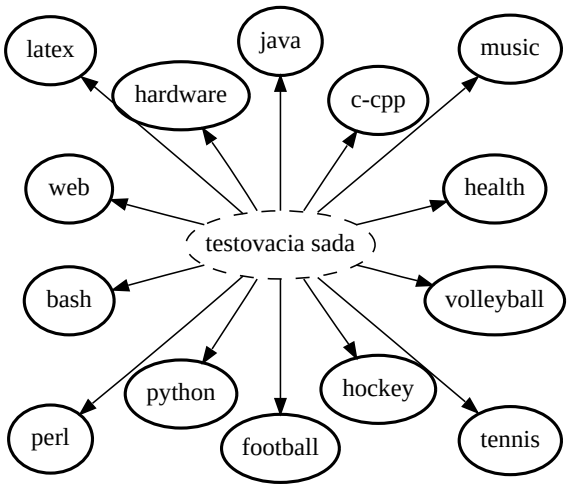
Z tabuľky 3.4 boli ďalej určené hodnoty správnosti, presnosti a citlivosti pre jednotlivé triedy uvedené v tabuľke 3.5, kde sú zvýraznené extrémne hodnoty. Z tejto tabuľky možno pozoravať, že klasifikátor má najmenšiu presnosť pre triedu *c-cpp* – do tejto triedy bolo zaradených najviac dokumentov, ktoré tam byť nemali (v relatívnom vyjadrení). Naopak najvyššia presnosť bola dosiahnutá pre triedy *perl* a *tennis*.

Zo stĺpca *Recall* možno vyčítať, že klasifikátor je najmenej citlivý pre triedu *hardware*. Z tejto triedy bolo najviac dokumentov zaradených nesprávne. Najvyššiu citlivosť systém preukázal pre triedu *latex*.

3.3.1 Úspešnosť klasifikácie

Pre testovaciu sadu vytvorenú v rámci tejto práce bola podľa tabuľky 3.4 určená úspešnosť klasifikácie na **90,7 %** (mikrovážený priemer – viď [9]). Pre sadu Reuters-21578 to bolo **94.55 %**. V oboch prípadoch bola použitá vektorová reprezentácia pomocou normalizovaných vektorov počtov výskytov termov. Dosiahnuté výsledky mierne prekonávajú výsledky uvedené v [7] a [14]. Toto môže byť spôsobené použitím, oproti SVMstruct, menej optimalizovanej knižnice libSVM.

Ďalej bolo prevedené testovanie úspešnosti klasifikácie pri použití jednoúrovňovej štruktúry tried vytvorenej testovacej sady podľa obrázka 3.2, keď systém určoval príslušnosť dokumentu priamo – bez použitia medzitried. Tento spôsob je rýchlejší a dosiahol úspešnosť **88,27 %** správne zaradených dokumentov.

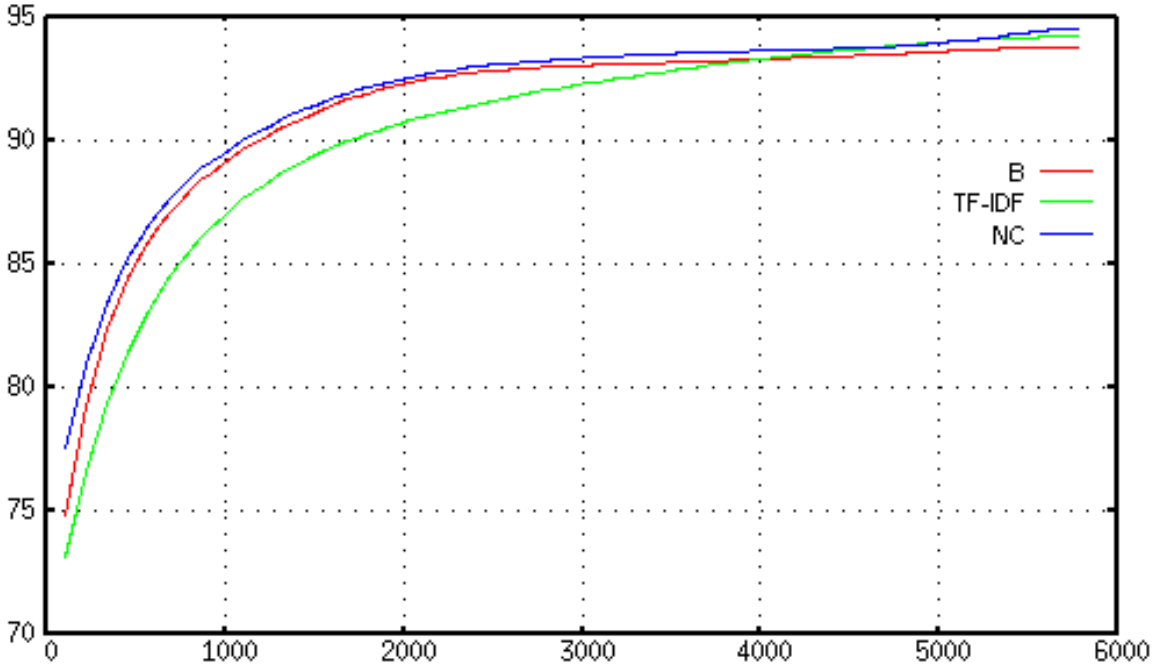


Obrázok 3.2: Štruktúra jednoúrovňovej testovacej sady.

Vplyv veľkosti trénovacej sady

Porovnanie závislostí úspešnosti klasifikácie od veľkosti trénovacej sady pre rôzne druhy vektrorovej reprezentácie dokumentov je uvedené na obrázku 3.3. Pre zostavenie týchto kriviek bola použitá sada Reuters-21578 rozdelená na trénovacie a testovacie príklady podľa ModApte. Klasifikátor bol trénovaný s rôznym počtom trénovacích príkladov – od 100 do 5 900 s krokom 200 dokumentov.

Na obrázku označenie B označuje použitie binárnych vektorov – zložka vektora nadobúda hodnotu 1 alebo 0 v závislosti od toho, či sa príslušný term v dokumente vôbec



Obrázok 3.3: Závislosť úspešnosti klasifikácie na veľkosti trénovacej sady pre rôzne druhy vektorizácie. Horizontálna os predstavuje počet dokumentov v trénovacej sade, vertikálna potom počet správne zaradených dokumentov z testovacej sady v percentách. Použitá sada Reuters-21578, rozdelenie na trénovacie a testovacie dokumenty ModApte. Hodnoty boli aproximované Bezierovou krivkou.

vyskytol. Označenie NC predstavuje použitie normalizovaných vektorov obsahujúcich počty výskytov jednotlivých termov a krivka pre TF-IDF označuje použitie inverzných dokumentových frekvencií (viac v časti 1.1.4).

Toto vyhodnotenie ukázalo, že veľkosť trénovacej sady nad 3 000 dokumentov zvyšuje úspešnosť klasifikácie len pozvoľna. Taktiež možno pozorovať, že vektorová reprezentácia pomocou normalizovaných vektorov počtov výskytov termov dosiahla prakticky pre všetky počty trénovacích príkladov lepšie výsledky ako ostatné metódy. Je však celkom možné, že pre väčšie sady (rádovo desaťtisíce dokumentov) by lepšie výsledky dosahovala metóda TF-IDF, kedže práve táto metóda je vo výzkumoch najpoužívanejšia (viď [9]).

Záver

Táto práca mala za úlohu zhrnúť poznatky o metódach a postupoch používaných pri klasifikácii textu podľa témy, na ich základe vytvoriť systém pre klasifikáciu dokumentov a ten následne otestovať na vytvorenej testovacej sade.

V prvej časti tejto práce boli zbežne popísané metódy používané pri strojovom spracovaní dokumentov. Informácie boli podané tak, aby aj čitateľ neorientujúci sa v problematike klasifikácie mohol nájsť východiská pre svoju prácu v odbore automatického spracovania textu. Špeciálna pozornosť bola venovaná metódam SVM, ktoré v poslednom čase zaznamenávajú značný úspech. V ďalšej kapitole bola popísaná analýza a realizácia jednoduchého systému pre klasifikáciu dokumentov podľa témy. Systém bol implementovaný tak, aby bol okamžite použiteľný aj bez akejkoľvek znalosti problematiky strojového učenia a klasifikácie. Systém tiež poskytuje možnosti pre ďalšie rozširovania a úpravy. S výhodou sa používa knižnica libSVM so svojimi nástrojmi, ktoré zapuzdrujú detaily implementácie a optimalizácie samotného procesu trénovania a klasifikácie. Realizovaný systém sa snaží posunúť automatickú klasifikáciu dokumentov zo sféry výskumu viac do oblasti bežného nasadenia, k čomu má napríklad slúžiť aj možnosť hierarchickej klasifikácie. Záverečná kapitola bola venovaná vytvoreniu testovacej sady a samotnému testovaniu vytvoreného systému. Štandardné metriky aj dosiahnuté výsledky som sa snažil okomentovať tak, aby získané hodnoty priniesli úžitok aj nezainteresovanému čitateľovi.

Smery ďalšieho uberania sa projektu vidím hlavne v rozširovaní možností v oblasti prepínania rôznych jazykov – ide hlavne o integráciu stemmerov pre rôzne jazyky. Vytvorenie stemmera pre český alebo slovenský jazyk by mohlo byť námetom na samostatnú bakalársku prácu. Identifikácia jazyka a voľba vhodného stemmeru by mohla dokonca prebiehať automaticky. Inou možnosťou vylepšenia je zavedenie rôznych pravidiel generalizácie tokenov v etape tokenizácie v závislosti od oblasti nasadenia klasifikačného systému. Ďalším smerom zdokonalenia systému je zrýchlenie prípravy dokumentov na klasifikáciu, kedže táto fáza zaberie porovnateľné alebo dokonca väčšie množstvo času ako samotné trénovanie, resp. klasifikácia.

Literatúra

[1] History of Support Vector Machines. [online], [cit. 6. 4. 2008].
URL <http://www.svms.org/history.html>

[2] BURGES, C. J.: A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, ročník 2, č. 2, 1998: s. 121–167.

[3] CHANG, C.-C.; LIN, C.-J.: *LIBSVM: a library for support vector machines*. 2001, [cit. 19. 4. 2008].
URL <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

[4] COHEN, W.: Text Classification. [online], [cit. 7. 4. 2008].
URL <http://videolectures.net/mlas06_cohen_tc>

[5] GUPTA, V.: Porter Stemming Algorithm. 2001, [cit. 19. 4. 2008].
URL <http://tartarus.org/~martin/PorterStemmer/python.txt>

[6] HSU, C.-W.; CHANG, C.-C.; LIN, C.-J.: A Practical Guide to Support Vector Classification. [online], 2008, [cit. 17. 4. 2008].
URL <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

[7] JOACHIMS, T.: Text categorization with support vector machines: Learning with Many Relevant Features. Technická zpráva, University of Dortmund, 1997, [cit. 6. 4. 2008].
URL
<http://www.springerlink.com/content/drhq581108850171/fulltext.pdf>

[8] LEWIS, D. D.: Reuters-21578. [online], [cit. 19. 4. 2008].
URL
<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

[9] MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H.: *An Introduction to Information Retrieval*. Cambridge University Press, England, 2008, knižne víde v priebehu roka 2008, [cit. 6. 4. 2008].
URL <http:
//www-csli.stanford.edu/~hinrich/information-retrieval-book.html>

[10] RENNIE, J. D. M.; RIFKIN, R.: Improving Multiclass Text Classification with the Support Vector Machine. Technická zpráva, Massachusetts Institute of Technology, 2002, [cit. 17. 4. 2008].
URL <http://people.csail.mit.edu/jrennie/papers/aimemo2001.pdf>

[11] VAPNIK, V. N.: *The Nature of Statistical Learning Theory*. Springer-Verlag New York, 1995, ISBN 0-387-94559-8.

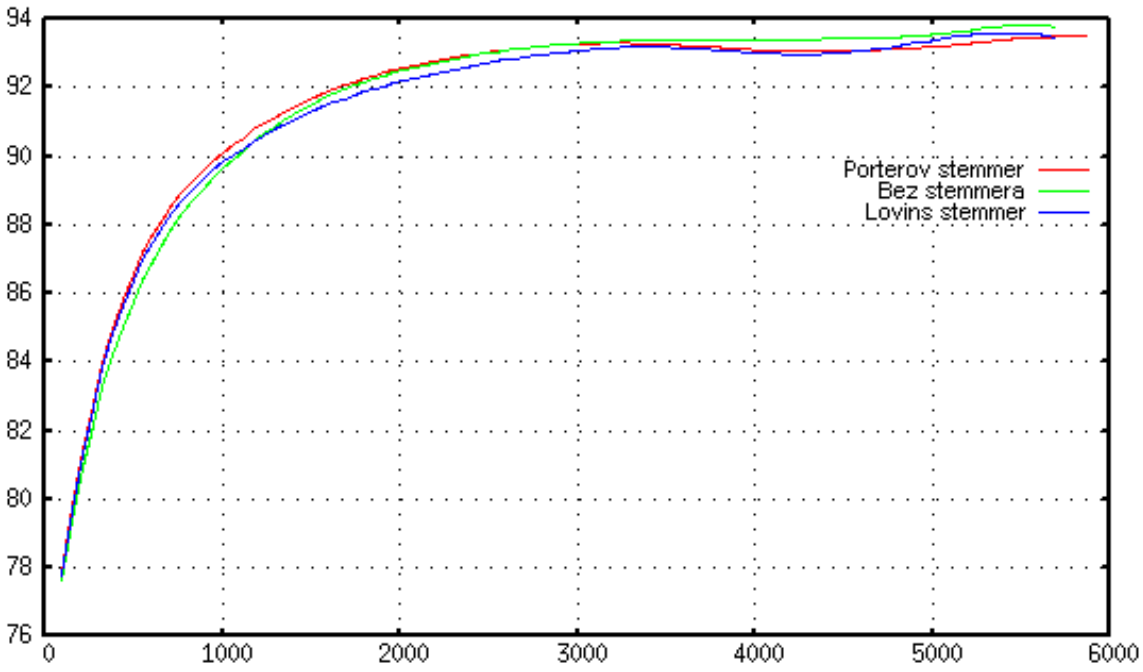
[12] Wikipedia: Stemming – Wikipedia, The Free Encyclopedia. [online], 2008, [cit. 23. 4. 2008].
URL
<http://en.wikipedia.org/w/index.php?title=Stemming&oldid=207372248>

[13] Wikipedia: Support vector machine – Wikipedia, The Free Encyclopedia. [online], 2008, [cit. 23. 4. 2008].
URL <http://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=206610626>

[14] YANG, Y.; LIU, X.: A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA: ACM, 1999, ISBN 1-58113-096-1, s. 42–49, [cit. 17. 4. 2008].
URL <http://doi.acm.org/10.1145/312624.312647>

Dodatok A

Vplyv stemmingu na úspešnosť klasifikácie



Obrázok A.1: Porovnanie vplyvu použitého stemmera na úspešnosť klasifikácie. Hodnoty boli aproximované pomocou Bezierovej krivky; krok testovania 200 dokumentov; použitá sada Reuters-21578 (10 najväčších tried); rozdelenie ModApte.

Na obrázku A.1 sú zachytené krivky závislosti úspešnosti klasifikácie od veľkosti trénovacej sady pre použitie *Porterovho stemmeru*, *Lovins stemmeru* a bez použitia stemmeru. Tento test prebiehal na desiatich najväčších triedach zo sady Reuters-21578, ktorá obsahuje anglicky písané dokumenty.

Ako už bolo spomenuté v časti 1.1.3 citujúc z [9], pre spracovanie anglických dokumentov použitie stemmingu vo všeobecnosti neprináša zvýšenie úspešnosti, čo je možné vidieť z uvedeného obrázka. Pre malé trénovacie sady prinieslo použitie stemmingu mierne zlepšenie, avšak s rastúcim počtom dokumentov určených na trénovanie sa rozdiel zotiera, dokonca sa ukazuje, že pre väčšie trénovacie sady je výhodnejšie stemming nepoužiť.

Dodatok B

Užívateľský manuál

B.1 Inštalácia

Najprv je potrebné získať zdrojové kódy nástroja libSVM. Tieto kódy sú umiestnené na CD priloženom k bakalárskej práci, na stránkach libSVM[1], alebo na niektorých distribúciách Linux-u je možné nainštalovať nástroj libSVM priamo z repozitárov.

1. Ak nemôžete nainštalovať potrebné nástroje z repozitárov, skopírujte do pracovaného adresára archív `libsvm-2.86.tar.gz` umiestnený na priloženom CD v adresári `system`. Archív rozbaľte a preložte:

   ```
   tar xvzf libsvm-2.86.tar.gz
   cd libsvm-2.86
   make
   ```

   Vzniknú tri sputiteľné súbory `svm-predict`, `svm-scale` a `svm-train`. Tieto súbory je potrebné skopírovať do niektorého adresára obsiahnutého vo vašej premennej prostredia `PATH`. Ja používam adresár `~/bin`.

2. Ďalej je potrebné skopírovať a rozbaliť archív `text-cls.tar.gz` umiestnený na CD v adresári `system`.

   ```
   tar xvzf text-cls.tar.gz
   cd text-cls
   make
   ```

   Po preklade pribudnú dva nové spustiteľné súbory `vectorize` a `linearize`.

3. Ďalej je potrebné skopírovať súbor `stopwords.dat`, napríklad do vášho domovského adresára. **Umiestnenie tohoto súboru je nutné nastaviť** v konfiguračnom súbore `svmconfig` v premennej `stopwdfile`.

4. Všetky spustiteľné súbory z adresára `text-cls` a tiež súbor `porter.py` skopírujte do vhodného adresára obsiahnutého vo vašej premennej prostredia `PATH`. Nemusí to byť nutne adresár zhodný s umiestnením nástrojov libSVM.

5. Odporúčam podobným postupom nainštalovať aj pomocné skripty z adresára `tools`. Tieto nástroje uľahčujú prácu pri testovaní.

[1] `<http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html>`

B.1.1 Príprava testovacej sady

Na priloženom CD sa v adresári `testsets/my` nachádzajú podadesáre `train` a `test`. Tieto obsahujú trénovacie a testovacie dokumenty sady použitej na testovanie systému implementovaného v rámci tejto bakalárskej práce. Skopírujte oba adresáre do pracovného adresára na vašom pevnom disku.

B.2 Základné použitie

Keď ste úspešne nainštalovali všetky súčasti systému a pripravili testovaciu sadu, ukážeme si ako systém použiť.

1. Prepnite sa do adresára `train` testovacej sady.

2. Zadajte príkaz `train`, tento spustí trénovanie nad dokumentami v jednotlivých podadresároch. V procese trénovania sa zobrazí niekoľko chybových správ oznamujúcich syntaktickú chybu vo vstupnom dokumente. Každopádne, trénovanie pokračuje ďalej. Proces trénovania môže trvať aj niekoľko desiatok minút.

3. Po skončení trénovania je potrebné presunúť natrénované modely a ďalšie informácie o trénovaní do adresára s dokumentami, ktoré sa majú klasifikovať. Toto zariadite príkazom

   ```
   cptrn ../test
   ```

   Skript `cptrn` rekurzívne prechádza adresáre trénovacej sady a vytvára v cielovom adresári rovnakú adresárovú štruktúru, ako mala trénovacia sada, vrátane kópií súborov potrebných na klasifikáciu.

4. Prepnite sa do adresára s dokumentami určenými na testovanie (klasifikáciu) a spustite samotnú klasifikáciu:

   ```
   cd ../test
   classify
   ```

   Dokumenty prítomné v adresári `test` budú roztriedené do adresárovej štruktúry vytvorenej skriptom `cptrn` v predchádzajúcom kroku.

V prípade, že si želáte použiť pre vektorovú reprezentáciu dokumentov inverzné dokumentové frekvencie, použite prepínač `-idf`:

```
train -idf
classify -idf
```

Pre jednoduché odstránenie súborov vytvorených v procesoch trénovania a klasifikácie je možné použiť jednoduchý skript `cleandtc` umiestnený v adresári `tools`. Tento skript rekurzívne prechádza aktuálny adresár a jeho podadresáre a maže súbory s príponami `.cls` a `.trn`. V prípade zmeny konfigurácie systému na vytváranie súborov s inými príponami nebude fungovať správne.

B.2.1 Vyhodnotenie

Pre zostavenie matice zámen sú v adresári `tools` na CD dva skripty `confusionmatrix` a `confusionrow`, ktoré počítajú správne a nesprávne zaradené dokumenty.

Skript `confusionmatrix` postačuje spustiť v adresári `test` a na štandardný výstup vypíše informácie o výsledkoch klasifikácie. V prípade použitia skriptu `confusionrow` je potrebné ako parameter odovzdať názov skutočnej triedy. Na štandardný výstup budú vypísané počty dokumentov zo zadanej triedy zaradené do iných tried, čo predstavuje jeden riadok matice zámen.

Skripty `confusionmatrix` a `confusionrow` sú jednoúčelové – vytvorené špeciálne pre túto testovaciu sadu. Keďže spoliehajú na stanovené pomenovania adresárov a testovacích dokumentov, je možné ich použiť len na tejto testovacej sade. Avšak pri dodržaní určitých zásad pomenovaní testovacích dokumentov (názov testovacieho dokumentu začína názvom triedy, do ktorej dokument patrí) je možné tieto skripty upraviť aj pre iné sady.

B.3 Zmena konfigurácie

Celá konfigurácia systému sa nachádza v súbore `svmconfig`. Tento súbor obsahuje hlavne nastavenia názvov súborov, ktoré sa vytvárajú počas činnosti systému. Je tu tiež možné nastaviť parameter trénovania c.

Prípony súborov vzniknutých pri trénovaní a klasifikácii odporúčam ponechať tak, ako sú, teda `.trn` a `.cls`. Toto je dôležité pre správnu funkciu skriptov, ako napríklad `cptrn` a `cleandtc`.

Asi najzaujímavejšou časťou konfigurácie je nastavenie cesty k súboru obsahujúcemu stop slová v premennej `stopwdfile`. Zadaná cesta musí byť **absolútna**.

B.4 Čo je v tých súboroch?

Táto časť popisuje obsah jednotlivých súborov. Odkazy na súbory sú uvádzané pomocou názvov premenných v konfiguračnom súbore a v zátvorkách je uvedená prednastavená hodnota, aby bolo možné dohľadať informácie aj po prípadnej zmene konfigurácie.

classListFile (classes.trn) – Zoznam názvov tried (podadresárov), do ktorých je trénovacia sada na aktuálnej úrovni hierarchie rozdelená. Tento súbor sa používa pre mapovanie názvov tried na číselné hodnoty tak, že prvý jeho záznam má čislenú hodnotu 1, druhý 2, atď. V celom procese trénovania a klasifikácie sa pracuje len s číselnými označeniami tried, ich skutočné názvy vstupujú do hry až po skončení klasifikácie, keď sa jednotlivé dokumenty presúvajú do priradených tried – adresárov.

trainIds (train.ids) – Obsahuje zoznam so správnou príslušnosťou trénovacích dokumentov do tried (ich číselné označenie - viď `classListFile`), tento súbor bude po natrénovaní skriptom `train` odstránený – má charakter dočasného súboru.

testIds (classify.ids) – Podobne ako `correctClassIdFile (train.ids)`, ale pre proces testovania. Obsahuje samé hodnoty 0, keďže pri klasifikácii správnu príslušnosť k triede nepoznáme. Slúži len pre vytvorenie správneho formátu súboru.

trnVectorsFile (train.dat) – Dočasný súbor pre uloženie vektorových reprezentácií trénovacích dokumentov. Po natrénovaní bude automaticky odstránený. Ich prislužnosť

k triede možno zistiť v súbore `correctClassIdFile (train.ids)` – poradie záznamov si zodpovedá. Z týchto dvoch súborov vytvára skript `train` súbor určený na trénovanie `trnSVMvectorsFile (data.trn)`.

clsVectorsFile (classify.dat) – Rovnako ako `trnVectorsFile (train.dat)`, ale pre proces klasifikácie.

trnSVMvectorsFile (data.trn) – Tento súbor obsahuje vektorovú reprezentáciu trénovacích dokumentov spolu s ich správnou príslušnosťou k triede vo formáte používanom libSVM a inými nástrojmi používajúcimi SVM.

clsSVMvectorsFile (data.cls) – Rovnako ako `trnSVMvectorsFile (data.trn)`, ale obsahuje vektory pre proces klasifikácie, pozícia obsahujúca príslušnosť k triede je pri všetkých vektoroch nulová – priradiť ju musí klasifikátor.

idftmp (tmp.idf) – Dočasný súbor pre uloženie informácií o inverzných dokumentových frekvenciách jednotlivých termov.

dfsFile (dfs.trn) – Súbor, kam budú uložene informácie o počte výskytov termov v celej trénovacej sade a počte dokumentov v trénovacej sade. Použije sa v prípade vektorovej reprezentácie pomocou inverzných dokumentových frekvencií (prepínač `-idf`).

modelFile (model.trn) – Natrénovaný model. Tento súbor sa používa v procese klasifikácie.

stopwdfile (/home/xorave00/stopwords.dat) – Absolútna cesta k súboru so stop slovami. Tento súbor obsahuje zoznam slov, ktoré budú pri spracovaní dokumentov ignorované. Každé slovo je na samostatnom riadku.

predictFile (predict.cls) – Obsahuje označenia tried priradených klasifikátorom klasifikovaným dokumentom. Poradie zodpovedá poradiu v `classifiedFile (files.cls)`.

classifiedFile (files.cls) – Zoznam názvov klasifikovaných súborov. Poradie zodpovedá poradiu tried v súbore `predictFile (predict.cls)`.

Ďalšie možnosti nastavenia sú popísané priamo v komentároch konfiguračného súboru. Stručný popis činnosti jednotlivých modulov je možné nájsť v hlavičkách skriptov. Podrobnejšie informácie o tom, ako moduly použiť samostatne, možno nájsť v priloženom `README` súbore.