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

Bakalářská práce

Automatická klasifikace textových dokumentů

Plzeň 2012 Veronika Černá

Prohlášení

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

V Plzni dne 9. května 2012

Veronika Černá

Abstract

This work deals with automatic text document classification. Text classification is a process of labelling documents with thematic categories from a predefined set of categories. Results of this work are supposed to be used by the Czech News Agency (ČTK). Three known classification techniques were chosen for experiments in this work: naive Bayes, support vector machines and maximum entropy. A lemmatizer and a POS–tagger were used for the text pre–processing. Four sets of documents were created based on the different feature selection criteria. All experiments were performed on the Czech corpus using the MinorThird toolkit. Experiments show that all classifiers perform well with slightly different recognition accuracy. The best obtained score is about 87 %. This score is sufficient for an application of the ČTK.

Obsah

1 Úvod

Automatická klasifikace textových dokumentů je problém založený na strojovém učení, u kterého došlo zejména v posledních letech k velkému rozvoji. Důvodem je především neustále se zvětšující množství dokumentů v elektronické podobě, které je potřeba nějakým způsobem strukturovat a organizovat. Jednou z možností je rozdělovat dokumenty do skupin na základě kategorie, do níž náleží. Přiřadit kategorii manuálně, jak tomu bylo v dřívějších dobách, dnes už však mnohdy není proveditelné, a je tedy potřeba tuto činnost nějakým způsobem zautomatizovat. A právě automatické přiřazování kategorie textovým dokumentům je oblast, které se budu v této práci věnovat.

Tato práce si klade za cíl prostudovat metody používané v oblasti automatické klasifikace textových dokumentů a vybrat z nich techniky, které budou otestovány na dodaných datech. K tomu je potřeba vyhledat vhodný nástroj pro klasifikaci textových dokumentů, což je další z cílů této práce. Prostřednictvím zvoleného nástroje budou provedeny experimenty a porovnány jejich výsledky.

Následující dvě kapitoly této práce jsou teoretické. Kapitola 2 je obecným úvodem do problematiky klasifikace textových dokumentů. Kapitola 3 je věnována popisu vybraných algoritmů, jimiž jsou naivní Bayesův klasifikátor, support vector machines a maximální entropie. V kapitole 4 lze nalézt výčet vyhledaných nástrojů pro klasifikaci včetně krátkého popisu každého z nich. Součástí této sekce je také srovnání těchto klasifikátorů a popis vybraného. Obsahem kapitoly 5 je popis úpravy zdrojových dat před jejich použitím pro klasifikaci, návrh a výsledky experimentů.

2 Rozbor úlohy

Tato kapitola je uvedením do problematiky automatické klasifikace textových dokumentů. Za definicí problému, která je obsahem první části, následuje shrnutí technik, jež se velmi často používají pro úpravy zdrojových dat. Třetí část pak představuje shrnutí důležitého kroku předcházejícímu samotné klasifikaci, kterým je výběr příznaků. V závěrečné sekci je popsáno, jak taková klasifikace probíhá a z jakých fází se skládá.

2.1 Definice klasifikační úlohy

Publikace [22] definuje kategorizaci textů jako úkol přiřadit booleovskou hodnotu každému páru $< d_j, c_i > \in D \times C$, kde D zastupuje soubor dokumentů a $C = \{c_1, c_2, ..., c_N\}$ je množina předdefinovaných kategorií. N zastupuje počet těchto kategorií. Hodnota T (*true* či 1) znamená rozhodnutí přiřadit dokumentu d_j kategorii c_i, zatímco hodnota F (*false* či 0) indikuje opak. Formálněji lze pak problém klasifikace dokumentů charakterizovat jako úkol nahradit neznámou cílovou funkci $\breve{\Phi} : D \times C \rightarrow \{T, F\}$, popisující jak by dokumenty měly být klasifikovány, funkcí $\Phi : D \times C \rightarrow \{T, F\}$, která je její aproximací. Funkci Φ nazýváme *klasifikátor* nebo také pravidlo, hypotéza či model.

Kategorie, respektive třídy, jsou pouhá symbolická označení a jsou definovány člověkem na základě toho, k jakému účelu bude klasifikace sloužit. V závislosti na využití pak lze také rozdělit kategorizaci do dvou základních skupin. První z nich zahrnuje případy, kdy je cílem přiřadit každému dokumentu právě jednu kategorii (tzv. případ *single–label*). Do druhé skupiny patří klasifikační úlohy s možností přiřadit každému dokumentu více kategorií (tzv. případ *multi–label*). Speciálním případem první skupiny je tzv. *binární klasifikace*, kde je potřeba vybrat právě jednu ze dvou tříd. Zdroj [22] poznamenává, že binární klasifikace a s ní i celá skupina *single–label* je obecnější než druhá možnost s překrývajícími se kategoriemi. Hlavním důvodem je fakt, že problém *multi–label* charakteru lze rozdělit na N nezávislých problémů binární klasifikace, obráceně to však nefunguje. Algoritmy pro multi–klasifikaci nelze použít, když je cílem vybrat pouze jednu třídu, protože nežádoucím výsledkem by mohlo být vybrání více kategorií nebo také žádné.

2.2 Parametrizace

Algoritmy strojového učení lze rozdělit do dvou základních skupin. Jednou z nich jsou metody učení s učitelem, druhou pak učení bez učitele. Liší se na základě toho, zda k natrénování používají označená vstupní data či nikoliv. Klasifikace textových dokumentů je typickým zástupcem metod učení s učitelem, která tato data vyžadují. Naším prvním krokem při vytváření jakéhokoli klasifikátoru by tedy mělo být opatření si korpusu, neboli souboru počítačově uložených dat, primárně sloužícímu k jazykovému výzkumu [12]. Pro účely klasifikace se jedná o soubor dokumentů, které již byly manuálně člověkem správně zařazeny do jedné z rozpoznávaných kategorií. Před použitím těchto dokumentů pro vytváření klasifikátoru je většinou nutné upravit je do podoby, kterou bude umět klasifikátor zpracovat. Tento proces je někdy nazýván parametrizace a jeho prvním krokem nejčastěji bývá tokenizace textu.

2.2.1 Tokenizace

Tokenizace označuje proces rozdělení textu na části nazývané *tokeny*. Těmi bývají nejčastěji slova, ale někdy také čísla, fráze, symboly či interpunkční znaménka. Je–li naším úkolem rozdělit text na jednotlivá slova, jistě nás napadne učinit tak na základě bílých znaků, tedy mezer, tabulátorů apod. Ovšem rozdělíme-li text podle mezer, zůstanou nám většinou součástí slov interpunkční znaky, které by měly být samostatně. Pokud tyto znaky přidáme k bílým znakům, spolu se kterými budou sloužit jako oddělovače, vznikne nám nový problém. Interpunkční znaky bývají totiž často součástí slov a místo jednoho tokenu tak vytvoříme více samostatných jednotek. Příkladem mohou být čísla, kde jsou tisíce odděleny mezerou (např. *100 000*), slova obsahující apostrof (*rock´n´roll*), označení skládající se z více slov (*New York*) a mnohé další případy. Úloha tokenizace textu je tedy mnohem komplikovanější, než by se mohlo na první pohled zdát, a vyžaduje hlubší analýzu konkrétního jazyka.

2.2.2 Lemmatizace a *stemming*

V textech se obvykle vyskytuje více tvarů stejného slova. V mnohých případech, včetně úlohy klasifikace textových dokumentů, může být užitečné převést všechny tyto tvary na jediný základní, což nám umožní například spočítat, kolikrát se dané slovo v textu vyskytuje, nebo provést některou další analýzu slov jako určení slovního druhu. K převodu slova do jeho základní podoby slouží proces lemmatizace

či *stemming*. Zdroj [20, s. 69] stemming popisuje jako „obvykle hrubý heuristický proces, jenž odsekne konce slov ve snaze o dosažení cíle“, kterým je zredukovat skloňované tvary (*am*, *are*, *is* $\rightarrow$ *be*) a případně odstranění derivačně souvisejících forem slova (*car*, *cars*, *car´s*, *cars´* $\rightarrow$ *car*). Lemmatizace pak dle téhož zdroje „dělá věci správně s využitím slovníku a morfologické analýzy slova s cílem odstranění pouze skloňovaných konců a vrácení slova do základního či slovníkového tvaru, známého jako *lemma*“. Tímto základním tvarem bývá např. pro podstatná jména 1. pád jednotného čísla, pro přídavná jména 1. pád jednotného čísla mužského rodu v kladném tvaru, u sloves infinitiv apod. Lemmatizátor je tedy nástroj, jenž při hledání lemmatu slova provádí morfologickou analýzu a k jeho činnosti je potřeba kompletní slovník. Stemmer oproti tomu vyžaduje méně znalostí. Řídí se pravidly pro stemming, která jsou pro každý jazyk specifická.

2.2.3 *POS–tagging*

Další důležitou technikou, jež má v oblasti zpracování přirozeného jazyka velmi široké využití, je *POS (part of speech) tagging*, jenž má za cíl pro každé slovo ve větě určit odpovídající jazykovou kategorii. Touto kategorií obvykle bývá slovní druh, jenž je často spojený s dalšími upřesňujícími informacemi, kterými jsou například typ nebo mluvnické kategorie. Znalost těchto informací pak lze uplatnit i pro potřeby klasifikace textových dokumentů, neboť umožňuje zredukovat objem zdrojových dat na základě daných kritérií.

V souvislosti s taggováním dokumentů v českém jazyce, které budou zdrojem pro experimenty této práce, je jistě na místě zmínit *Pražský závislostní korpus 2.0* (PDT 2.0) [13]. Jedná se o probíhající projekt pro ruční anotaci velkého množství českých textů bohatou lingvistickou informací, která zahrnuje oblast morfologie, syntax i sémantiku. Kromě toho, že obsahuje velké množství českých textů, disponuje softwarovými nástroji pro prohledávání korpusu, anotaci dat a jazykovou analýzu.

2.3 Výběr příznaků

Texty sloužící jako zdrojová data pro klasifikátory jsou obvykle reprezentovány jako vektory příznaků, někdy nazývaných *termy*. Konkrétní dokument d_j lze tedy vyjádřit jako $\overrightarrow{d}_j =< w_{1j}, ..., w_{Tj} >$, kde w jsou váhy termů a T je soubor termů, které se vyskytují minimálně jednou v alespoň jednom dokumentu. Příznaky zde

obvykle zastupují slova či lemmata, ale nemusí tomu tak být vždy. Ne všechna slova, která se v textu vyskytují, ovšem používáme jako příznaky. Některá slova, zejména ta, jež se v textu objevují velmi často, je obvykle možné vynechat. Takovým slovům říkáme *stop slova*.

K rozhodování, která slova použít jako příznaky a která je možné odfiltrovat jako stop slova, existuje celá řada strategií. Jednou z možností je aplikace výše zmíněného POS–taggingu, jenž nám umožňuje ponechat slova pouze určitých slovních druhů a zbytek odstranit. Jako užitečné se na první pohled jeví například podstatná a přídavná jména, slovesa či příslovce. Naopak se zdá, že často se v textu vyskytující slovní druhy jako předložky nebo spojky by pravděpodobně nepřispěly ke zvýšení úspěšnosti klasifikátoru.

Jako další často využívanou strategii, která již nevyžaduje použití POS–taggeru, lze uvést počítání četnosti jednotlivých slov a odstranění těch, která se ve třídě vyskytují nejčastěji. Tato četnost může být definována dvěma základními způsoby. První z nich je zjišťování počtu dokumentů ve třídě, které obsahují daný term (*document frequency*). Druhá varianta pak spočívá ve výpočtu, kolikrát se slovo nachází v dokumentech patřících do určité třídy (*collection frequency*).

Zjišťování četností slov pochopitelně není jediným možným měřením, které lze za účelem výběru příznaků provádět. Dalším příkladem může být měření vzájemné informace (*mutual information*), jejíž hodnota udává nakolik přítomnost či absence termu přispěje ke správnému rozhodnutí klasifikátoru. Jinak řečeno, vzájemná informace měří, kolik informace o třídě term obsahuje. Mezi další měření určené pro výběr příznaků pak patří např. X^2 (*chi-square*), DIA asociační faktor (*DIA association factor*), informační zisk (*informational gain*), poměr zvláštností (*odds ratio*) a jiné [22].

2.4 Průběh klasifikace

Po zpracování zdrojového korpusu a zredukování velikosti vektorů příznaků reprezentujích dokumenty lze přistoupit k samotnému vytváření klasifikátoru textových dat. Postup se vždy skládá ze dvou základních fází, kterými jsou trénování a testování. Výsledkem fáze trénování (učení) je vytvoření klasifikátoru, pro který je pak vhodné znát jeho úspěšnost. K tomuto účelu slouží fáze testovací. Za třetí fázi pak lze označit použití natrénovaného modelu v praxi pro klasifikaci dokumentů.

Aby bylo možné provést obě fáze vytváření klasifikátoru, je nutné rozdělit zdro-

jová data na dvě nepřekrývající se množiny. Na první z nich, na tzv. trénovacím souboru dat, probíhá učení klasifikátoru a druhá, testovací sada, je pak určena k jeho ohodnocení. Pro vytvoření co nejlepšího klasifikátoru je vhodné, aby objem trénovacích dat byl co největší. Testovací množina pak obvykle bývá mnohem menší. Poměr rozdělení zdrojových dat lze zvolit libovolně. Je na zvážení tvůrce klasifikátoru, jaký objem dat použije pro trénování a jaký si ponechá stranou pro testovací fázi.

Alternativním přístupem, jenž nevyžaduje rozčlenění zdrojových dat na trénovací a testovací sadu, je tzv. křížová validace (*k-fold cross-validation*). V tomto případě jsou zdrojová data rozdělena do k disjunktních částí. Pro každou je pak vytvořen klasifikátor, jenž je ohodnocen na principu trénovacích a testovacích dat, kde vybraná k-tá část zastupuje testovací množinu a zbytek dat, tedy všechny skupiny bez této k-té části, slouží jako data trénovací. Celková účinnost je spočtena jako průměr účinností jednotlivých klasifikátorů.

3 Klasifikační algoritmy

Přístupů ke klasifikaci dokumentů existuje celá řada. Mezi nejznámější skupiny klasifikátorů patří pravděpodobnostní metody, rozhodovací stromy, neuronové sítě, klasifikátory založené na rozhodovacích pravidlech, tzv. *líné klasifikátory*, různé *on–line* metody a další. Tato práce je zaměřena pouze na tři vybrané přístupy, které dle prostudované literatury pracují s velkou přesností: naivní Bayesův klasifikátor, support vector machines a maximální entropie. Popis těchto metod je obsahem této kapitoly.

3.1 Naivní Bayesův klasifikátor

Naivní Bayesův klasifikátor (NB) má dvě základní verze. Základní a známější variantou je tzv. multinomiální (*multinomial*) model, kterému je věnována sekce 3.1.1. Jeho modifikací je pak Bernoulliho model, jenž je podrobněji popsán v sekci 3.1.2.

3.1.1 Multinomiální model

Naivní Bayesův klasifikátor je pravděpodobnostní metoda založená na Bayesově teorému. Bayesova věta je jednou ze základních vět teorie pravděpodobnosti a statistiky. Zabývá se podmíněnými pravděpodobnostmi jevů, kdy dává do vztahu podmíněnou pravděpodobnost s opačnou podmíněnou pravděpodobností. Bayesovu větu lze formulovat takto:
Máme-li dva náhodné jevy A (v našem případě textový dokument, ke kterému chceme přiřadit nějakou kategorii) a B (hypotéza, že A patří k určité třídě) s pravděpodobnostmi $P(A)$ a $P(B)$ a pokud $P(B) > 0$, pak platí:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

$P(A|B)$ je podmíněná pravděpodobnost, též označována jako *aposteriorní*, jevu A za předpokladu, že nastal jev B, tedy pravděpodobnost platnosti B pro vzorek A. $P(B|A)$ je naopak pravděpodobnost jevu B podmíněná jevem A. $P(A)$ je v tomto případě tzv. *apriorní* pravděpodobnost a odpovídá znalostem o zastoupení jednotlivých hypotéz (tříd) [11].

Kategorií, mezi kterými se rozhoduje, bývá obvykle celá řada. Cílem je vždy vybrat tu nejvhodnější, což v případě naivního Bayesova klasifikátoru bude ta, jejíž pravděpodobnost nabude nejvyšší hodnoty. Konkrétní hodnota této pravděpodobnosti nás však nezajímá, a proto lze vzorec zjednodušit vynecháním jmenovatele. Pravděpodobnost, že dokument d bude ve třídě c, lze tedy spočítat jako

$$P(c|d) = P(c) \times \Pi_{1 \leq k \leq n_d} P(t_k|c).$$

Písmeno t zde zastupuje *token*, respektive příznak. $< t_1, t_2, ..., t_{n_d} >$ jsou tedy příznaky dokumentu d, n_d pak počet těchto příznaků. $P(t_k|c)$ je podmíněná pravděpodobnost, že se t_k vyskytuje v kategorii c. *P(c)* udává pravděpodobnost, že se jedná o kategorii c, tzn. apriorní pravděpodobnost, dle [20].

Důležitým rysem naivního Bayesova klasifikátoru je předpoklad nezávislosti jednotlivých příznaků. Realita je ovšem jiná. Slova ve větě na sobě nepochybně závisí. Stejně tak lze vyvrátit druhý předpoklad klasifikátoru, kterým je nezávislost slov na pozici ve větě. Přes tyto očividně naivní předpoklady, podle nichž je odvozen název klasifikátoru, však Bayesův klasifikátor pracuje uspokojivě. Jak je toto možné? Zdroj [20, s. 306] uvádí tento příklad: Máme dokument d s pravděpodobnostmi $P(c_1|d) = 0,6$ a $P(c_2|d) = 0,4$. Dále předpokládáme, že d obsahuje mnoho příznaků, které jsou pozitivními indikátory pro třídu c_1, a hodně příznaků, jež jsou negativními indikátory pro c_2. Potom bude platit, že hodnota *P($t_k|c_1$)* se bude blížit hodnotě 1.0, zatímco hodnota *P($t_k|c_2$)* bude téměř nulová. Rozdíl je velký a stejně tak tomu bývá ve většině případů. Správná třída mívá zpravidla mnohem větší pravděpodobnost než ostatní kategorie. Vzhledem k jeho efektivitě a jednoduchosti je naivní Bayesův klasifikátor jedním z nejoblíbenějších (ve smyslu nejpoužívanějších) klasifikátorů textových dokumentů.

3.1.2 Bernoulliho model

Bernoulliho model funguje na velmi podobném principu jako obecný naivní Bayesův klasifikátor popsaný v předchozí sekci. Pro tento tzv. multinomiální model je příznak ze slovníku generován pro každou jeho pozici. Alternativou k němu je vícerozměrný (*multivariate*) Bernoulliho model, jenž pro každý příznak ze slovníku vygeneruje binární hodnotu, která udává, jestli se daný příznak v dokumentu nachází či nikoliv. Hodnota 1 znamená, že se příznak v dokumentu vyskytuje, hodnota 0 indikuje opak. Jinými slovy, zatímco multinomiální model udržuje informaci o četnosti výskytu každého příznaku, Bernoulliho model tuto informaci ignoruje a zajímá ho jen, jestli se příznak v dokumentu objevuje nebo ne. Podle zdroje [20, s. 264] je toto, obzvláště u dlouhých dokumentů, často příčinou chybné klasifikace a jako příklad uvádí, že

dokument může být zařazen do třídy *Čína* pouze na základě jediného výskytu slova *Čína* v textu.

Oba modely se taktéž liší v tom, jak nepřítomnost určitého příznaku ovlivní klasifikaci. Zatímco u multinomiálního modelu nemá absence příznaku při rozhodování žádný vliv, Bernoulliho model je tímto faktorem ovlivněn, neboť každý příznak je součástí výpočtu pravděpodobnosti $P(c|d)$. Tyto a další rozdíly v obou modelech jsou shrnuty v tab. 3.1.

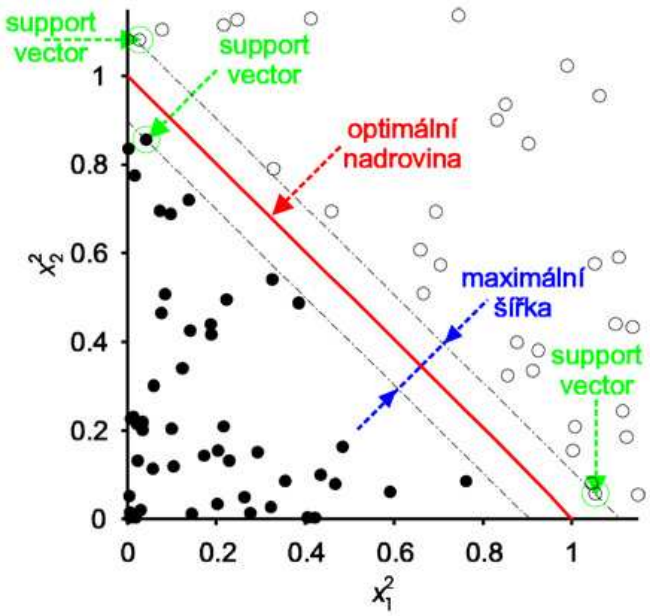
	multinomiální model	Bernoulliho model
činnost modelu	generování tokenů	generování dokumentu
náhodná proměnná	$X = t$ if t na dané pozici	$U_t = 1$ if t v dokumentu
reprezentace dokumentu	$d =< t_1, ..., t_k, ..., t_{n_d} >$, $t_k \in V$	$d =< e_1, ..., e_i, ..., e_M >$, $e_i \in \{0, 1\}$
odhad parametru	$\hat{\mathrm{P}}(X = t\|c)$	$\hat{\mathrm{P}}(U_i = e\|c)$
rozhodovací pravidlo	$\hat{\mathrm{P}}(\mathrm{c})\ \Pi_{1 \le k \le n_d} \hat{\mathrm{P}}(X = t_k\|c)$	$\hat{\mathrm{P}}(\mathrm{c})\ \Pi_{t_i \in V} \hat{\mathrm{P}}(U_i = e_i\|c)$
vícenásobný výskyt	brán v úvahu	ignorován
délka dokumentu	zpracuje delší dokumenty	pracuje lépe s menšími dokumenty
příznaky	zpracuje větší počet	nejlépe funguje jen s několika
odhad pro příznak *the*	$\hat{\mathrm{P}}(X = the\|c) \approx 0,05$	$\hat{\mathrm{P}}(U_{the} = 1\|c) \approx 1,0$

Tabulka 3.1: Srovnání multinomiálního a Bernoulliho modelu, zdroj: [20].

3.2 Support vector machines

Zástupcem relativně nových metod strojového učení je metoda *support vector machines* (SVM) patřící do kategorie tzv. jádrových algoritmů. Jedná se o binární metodu, jež rozhodne, do které ze dvou tříd bude objekt, v našem případě dokument, zařazen. SVM je založená na vektorovém prostoru a jejím cílem je najít ve vstupním prostoru příznaků lineární hranici oddělující pozitivní trénovací příklady od negativních. Tímto lineárním oddělovačem je nadrovina, a to konkrétně taková, která bude poskytovat co nejširší rozpětí mezi ní a pozitivními příklady na jedné straně a negativními na straně druhé. Tato rovina je označována jako

optimální. Body, které jsou nejblíže optimální nadrovině, se nazývají *podpůrné vektory* (support vectors), odkud plyne název metody. Jejich funkcí je totiž „podpora“ oddělovací nadroviny. Ostatní body pozici oddělovače nijak neovlivňují a není jich proto zapotřebí. Podpůrné vektory spolu s příslušnou optimální nadrovinou jsou znázorněny na obr. 3.1.



Obrázek 3.1: Pohled na optimální nadrovinu s podpůrnými vektory, zdroj: [8].

SVM hledá optimální nadrovinu pomocí metody kvadratického programování. Předpokládáme příklady x_i s klasifikací $y_i = +1$ pro pozitivní příklady či $y_i = -1$ pro negativní. Cílem je najít hodnoty parametru α_i, které maximalizují výraz

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)^2,$$

kde platí

$$\alpha_i \geq 0 \; a \sum_i \alpha_i y_i = 0.$$

Po spočítání optimálních parametrů α_i, je pak možné oddělovač najít pomocí rovnice

$$h(x) = sign\left(\sum_i \alpha_i y_i (x \cdot x_i)\right).$$

Ne vždy jsou však pozitivní a negativní příklady v původním vstupním prostoru **x** od sebe lineárně separovatelné. Tuto situaci lze řešit využitím techniky jádrové transformace. Jedním z jejích základních principů je převod originálního vstupního prostoru do vícedimenzionálního, kde už půjde od sebe třídy lineárně oddělit. Tímto způsobem, tedy mapováním do prostoru s dostatečným počtem dimenzí, je možné nakonec vždy najít optimální nadrovinu.

Ve vícerozměrném prostoru F(x) lze lineární oddělovač najít nahrazením členu $x_i \cdot x_j$ členem $F(x_i) \cdot F(x_j)$. Například pro tří–dimenzionální prostor platí

$$F(x_i) \cdot F(x_j) = (x_i \cdot x_j)^2,$$

přičemž výraz $(x_i \cdot x_j)^2$ je nazýván jádrovou funkcí $K(x_i, x_j)$. Nalezené oddělovače je pak možné zpětně mapovat do originálního prostoru, dle [8].

Ačkoliv je metoda support vector machines primárně binárním klasifikátorem, lze ji použít i v situaci, kdy je třeba rozhodnout mezi více třídami místo pouhých dvou. Principem je převést tento problém opět do binární podoby. Existují dvě základní varianty, jak toho docílit. Jednou z možností je řešit tento problém jako jedna třída versus zbytek (*jedna versus všechny*). Druhou variantou je nakombinovat třídy do všech možných dvojic (*jedna versus jedna*) a pro každou vytvořit speciální klasifikátor.

3.3 Maximální entropie

Maximální entropie (ME), známá také pod názvem multinomiální logistická regrese, je další metodou strojového učení. Stejně jako v případě naivního Bayesova klasifikátoru jde o pravděpodobnostní metodu, a je tak jednou z jeho alternativ. Její hlavní výhodou oproti Bayesově metodě je fakt, že netrvá na nezávislosti příznaků.

Podle [18, s. 227] maximální entropie funguje tak, že ze vstupních dat extrahuje soubor příznaků a lineárně je zkombinuje, respektive každý příznak je vynásoben jeho váhou (četnost výskytu) a následně přičten. Tato suma je pak použita jako exponent. Je-li naším cílem zvolit třídu např. pro dokument, maximální entropie vybere tu, která je nejpravděpodobnější, přičemž pravděpodobnost pro konkrétní kategorii c a vstup (dokument) d lze spočítat podle vzorce

$$p(c|d) = \frac{1}{Z} exp(\sum_i w_i \ f_i), \tag{3.1}$$

kde f_i je příznak, w_i jeho váha a Z normalizační faktor. Tento vztah představuje takový model rozdělení pravděpodobnosti, který maximalizuje hodnotu entropie. Na základě zdroje [15] lze rovnici 3.1 odvodit takto:

Trénovací data jsou tvořena N páry $(d_1, c_1), \ldots, (d_N, c_N)$, kde $c_i \in C$ jsou třídy a $d_i \in D$ dokumenty reprezentované jako vektor příznaků. Pro ně nás bude zajímat jejich „empirické“ pravděpodobnostní rozložení:

$$\tilde{p}(d|c) = \tfrac{1}{N} \times \text{ počet výskytů (d,c).}$$

Úkolem je vytvořit statistický model procesu, jenž generuje trénovací data $\tilde{p}(d, c)$. Očekávanou hodnotu příznaku f s ohledem na tuto empirickou distribuci lze vyjádřit jako

$$\tilde{p}(f) \equiv \sum_{d,c} \tilde{p}(d, c) \cdot f(d, c).$$

Pro model $p(c|d)$ je očekávaná hodnota příznaku

$$p(f) \equiv \sum_{d,c} \tilde{p}(d) p(c|d) \cdot f(d, c),$$

kde $\tilde{p}(d)$ představuje empirické rozložení d v trénovacích datech. Hodnoty $p(f)$ a $\tilde{p}(f)$ pokládáme za shodné a rovnici $p(f) = \tilde{p}(f)$ nazýváme *omezení*. Zkombinováním předchozích vztahů dostaneme

$$\sum_{d,c} \tilde{p}(d) p(c|d) \cdot f(d, c) = \sum_{d,c} \tilde{p}(d, c) \cdot f(d, c).$$

Může existovat nekonečně mnoho modelů, pro které platí soubor omezení. My vybereme ten nejjednotnější z nich. Tato jednotnost je měřena prostřednictvím entropie $p(c|d)$ takto:

$$H(p) \equiv -\sum_{d,c} \tilde{p}(d) p(c|d) log\ p(c|d).$$

Jak již bylo řečeno, vybíráme takový model $p_* \in C$, který maximalizuje $H(p)$. Vyjádříme-li p_* v exponenciální formě, dostaneme vztah 3.1.

Zdroj [21, s. 596–597] uvádí, že maximální entropie poskytuje pravděpodobnostní rámec pro integrování informací z heterogenních zdrojů. Za účelem podpoření správného rozhodnutí klasifikátoru lze totiž definovat libovolný soubor příznaků, který bude přidán jako další omezení modelu. Jinou předností maximální entropie je pak dle stejného zdroje skutečnost, že váha příznaků neovlivní výsledky klasifikace, což neplatí u většiny jiných metod.

4 Výběr klasifikátoru

Pro klasifikaci dokumentů bylo nutné nejprve najít vhodný nástroj a to takový, který bude splňovat dvě hlavní kritéria. Prvním z nich byla implementace v jazyce Java, popřípadě C++, druhým pak, aby byl nástroj zdarma. To výběr značně zúžilo. Podařilo se najít celkem 10 nástrojů pro klasifikaci textových dokumentů. Každému z nich je v kap. 4.1 věnován odstavec obsahující jeho základní charakteristiku včetně nejdůležitějších vlastností. V části 4.2 pak lze nalézt jejich srovnání spolu se závěrem, který z nich byl vybrán spolu s důvody proč. Veškeré informace o jednotlivých nástrojích jsou čerpány ze zdroje uvedeného v úvodu každé části. Sekce 4.3 obsahuje bližší popis vybraného klasifikátoru.

4.1 Popis klasifikátorů

4.1.1 Dragon Toolkit

Dragon Toolkit [3] je nástroj určený k akademickému použití. Jedná se o balík tříd napsaných v jazyce Java zaměřený na vyhledávání informací v textu a dolování dat zahrnující kategorizaci, shlukování a sumarizaci textů a modelování tématu. Je v něm integrována řada nástrojů pro zpracování přirozeného jazyka, což umožňuje indexovat množiny textů o rozdílných reprezentacích. Jeho důležitým rysem je škálovatelnost a fakt, že narozdíl od jiných podobných nástrojů umožňuje široké využití. Text je reprezentován pomocí řídké matice a není nutné načíst do paměti veškerá data, což může být výhodou zejména v případech, kdy máme omezenou paměť a zároveň velké množství dokumentů. Mezi základní trénovací algoritmy, které Dragon Toolkit implementuje, patří SVM, naivní Bayesův klasifikátor a Nigamovo aktivní učení.

4.1.2 Lingpipe

Lingpipe [6] je aplikací napsanou v jazyce Java s velmi širokým spektrem využití. Vedle klasifikace dokumentů do tříd patří mezi jeho funkce rozpoznávání pojmenovaných entit, shlukování (technika pro seskupování objektů na základě podobnosti), značkování částí promluv (přiřazení syntaktického označení), oprava pravopisu, porovnávání řetězců, vyhledávání syntakticky významných částí textu, iden-

tifikace jazyka a další. Pro účely klasifikace textových dokumentů je v něm implementována řada algoritmů. Mezi ně patří naivní Bayesův, Bernoulliho, *k–Nearest Neighbor* (KNN) a *Language Model* (LM) klasifikátor, vícevrstvý perceptron, logistická regrese a lineární klasifikátor pracující s vektory (tzv. *BigVector* klasifikátor), určený zejména pro případy velkého množství kategorií. Lingpipe je distribuován pod 4 různými licencemi, přičemž 1 verze produktu je volně ke stažení. Zbývající 3 jsou placené a určené ke komerčnímu využití.

4.1.3 Mahout

Mahout [1] je projektem, který si klade za hlavní cíl být škálovatelnou knihovnou strojového učení. Má zvládnout velký soubor dat a být k užitku komukoli, kdo o něj projeví zájem. Kdokoli se může k projektu připojit a stát se členem komunity, která se podílí na vývoji. Projekt má ambice obsáhnout celou řadu různých algoritmů z mnoha oblastí strojového učení včetně klasifikace, shlukování či dolování dat. Konkrétně z problematiky klasifikace dokumentů obsahuje naivní Bayesův klasifikátor, logistickou regresi, *Random Forests* a *Online Passive Aggressive*. Celá řada algoritmů je však právě ve vývoji, či je v plánu je v budoucnu začlenit. Z těchto lze jmenovat například podpůrné vektory, *Hidden Markov Models* (HMM), neuronové sítě a perceptrony.

4.1.4 Mallet

Mallet [7] je balíkem javovských tříd určených k aplikaci metod strojového učení na textová data. Mimo jiné obsahuje sofistikovaný nástroj pro klasifikaci dokumentů, mezi jehož součásti patří metody pro převod vstupních dat do požadovaného formátu, několik různých algoritmů pro trénování dat a nástroj pro vyhodnocení dosažených výsledků. Začleněnými algoritmy pro klasifikaci dokumentů jsou naivní Bayesův klasifikátor, maximální entropie a rozhodovací stromy. Na webových stránkách produktu lze nalézt přehledné a srozumitelné návody pro jeho použití.

4.1.5 MinorThird

MinorThird [16] je kolekcí javovských tříd pro strojové učení. Jedná se o aplikaci, kterou lze použít především pro ukládání a anotaci textu, extrahování pojmenovaných entit z textu či kategorizaci textových dokumentů. Pro tento účel je v něm

implementováno několik různých algoritmů, mezi které patří maximální entropie, naivní Bayesův klasifikátor, SVM, KNN, perceptrony, rozhodovací stromy a některé další. Více o tomto nástroji lze nalézt v kapitole 4.3.

4.1.6 OpenNLP

OpenNLP [2] je dalším z nástrojů založených na strojovém učení pro zpracování přirozeného jazyka. Mezi jeho funkce patří tokenizace, segmentace vět, značkování částí řeči, extrahování pojmenovaných entit, rozdělování textu do skupin slov na základě jejich syntaktického vztahu (tzv. *chunking*), parsování, řešení koreferencí[1] a samozřejmě také klasifikace textových dokumentů do předdefinovaných kategorií. Ta je zde realizována na principu algoritmu maximální entropie.

4.1.7 Stanford Classifier

Stanford Classifier [19], čili stanforský klasifikátor, byl vytvořen na Stanfordově univerzitě skupinou vědců zabývajících se zpracováním přirozeného jazyka. Napsán je v Javě. Je zaměřen výhradně na klasifikaci dat do tříd, k čemuž zde slouží metoda maximální entropie. Nejlepší výsledky podává pro textová data, ale je schopen zpracovat i numerické vstupy. Klasifikátor by měl být kompatibilní s dalším softwarem pro zpracování přirozeného jazyka, který byl vyvinut na této univerzitě (*Stanford Named Entity Recognizer*, *Stanford POS Tagger*, *Stanford Parser* a další).

4.1.8 SVMTorch II

SVMTorch II [9] je, jak už název napovídá, zaměřen výhradně na metodu podpůrných vektorů. Jako jediný zde uvedený program je implementován v jazyce C++. Lze ho použít pro účely klasifikace, ale i regrese. Soustřeďuje se zejména na problémy velkého rozsahu, do kterých, dle domovských stránek projektu, spadají případy s více jak 20000 příklady, a dokonce vstupy o více jak 100 dimenzích. Vývoj samotného SVMTorch byl pozastaven, neboť se stal součástí většího celku nazývaném *Torch* [4].

[1] Koreference je vzájemný vztah dvou nebo více výrazů vyskytujících se v textu a poukazujících k témuž předmětu řeči (osobě, předmětu, skutečnosti) [5].

4.1.9 TCatNG

TCatNG [10] je dalším nástrojem, jenž je tvořen souborem javovských tříd. Jeho zvláštností je reprezentace dat pomocí N-gramů, krátkých sekvencí bytů či písmen. N-gramy jsou jednou z možností, jak přistupovat ke kategorizaci textů. Jednotlivé řetězce jsou rozloženy na malé části, díky čemuž případná chyba ovlivní pouze malou část a zbytek zůstane nedotčen. S aplikací N-gramů navíc odpadá potřeba tokenizace a použití stemmeru či lemmatizátoru. Součástí TCatNG jsou také další algoritmy pro klasifikaci textů: SVM, bayesovská logistická regrese, bayesovské sítě a algoritmus založený na kompresi textu.

4.1.10 Weka

Weka [14] je velmi obsáhlým nástrojem pro strojové učení a dolování dat vyvinutý v Javě. Jeho hlavní síla leží v oblasti klasifikace. Měly by být implementovány všechny současné metody strojového učení včetně některých starších. Kromě toho obsahuje algoritmy pro regresi, asociační pravidla a shlukování. Ze začleněných klasifikátorů lze jmenovat zejména rozhodovací stromy, naivní Bayesův klasifikátor, lineární a logistickou regresi či SVM. Weka díky své obsáhlosti není nástrojem, jenž by bylo snadné zvládnout, k čemuž nepřispívá, dle mého názoru, velmi stručná dokumentace.

4.2 Srovnání klasifikátorů

Nejdůležitější údaje k jednotlivým klasifikátorům jsou zobrazeny v tab. 4.1. Ta obsahuje celkem 5 sloupců. V prvním je uveden název nástroje pro klasifikaci a v druhém jazyk, ve kterém je implementován. Následuje časový údaj o tom, kdy byly provedeny poslední úpravy nástroje. Tato informace může být velmi užitečnou, neboť si z ní lze udělat představu, zdali byl vývoj klasifikátoru již ukončen, či je stále upravován nebo rozšiřován. Dále je v tabulce vypsán typ licence, pod kterou je distribuován, pokud byla na webových stránkách či v dokumentaci nalezena. V posledním sloupci tabulky jsou vypsány typy algoritmů, které jsou v každém nástroji implementovány.

Implementované algoritmy byly tím nejdůležitějším, co rozhodovalo o tom, jaký nástroj bude použit pro klasifikaci dat. Hledán byl takový, který má implementovány minimálně 3 různé metody pro klasifikaci dokumentů, nejlépe naivní Bayesův

klasifikátor, maximální entropii a podpůrné vektory (SVM). Jediný z nalezených nástrojů, jenž toto splňuje, je MinorThird. V jeho prospěch mluvila také skutečnost, že se nejedná o příliš rozsáhlou aplikaci, kterou by bylo složité zvládnout, a fakt, že jako jeden z mála výše uvedených nástrojů disponuje vcelku přehlednými návody, jež usnadňují jeho použití. Na základě těchto důvodů byl MinorThird vybrán pro další experimenty s klasifikací textových dokumentů.

4.3 Popis nástroje MinorThird

4.3.1 Struktura

Rozhraní nástroje MinorThird je rozděleno do 4 hlavních částí:

1. `edu.cmu.minorthird.classify`
2. `edu.cmu.minorthird.text`
3. `edu.cmu.minorthird.ui`
4. `edu.cmu.minorthird.util`

Část **classify** obsahuje algoritmy strojového učení pro extrahování a klasifikaci, struktury pro ukládání netextových dat, klasifikátory a třídy pro vyhodnocení experimentů. Tento balík lze stáhnout a následně použít samostatně. Hodí se však spíše jen pro binární klasifikaci a menší objemy dat. V části **text** jsou zahrnuty třídy určené pro zpracování textových dat. Také je zde začleněn tzv. `Mixup` (*My Information eXtraction and Understanding Program*). Jedná se o speciální jazyk nástroje MinorThird pro manipulaci s textem. Balík **ui** představuje uživatelské rozhraní pro spouštění experimentů s textovými daty a v části **util** se nachází utility pro příkazovou řádku a grafické uživatelské rozhraní. Pro potřeby klasifikace jsou důležité zejména 2 součásti nástroje MinorThird a to `TextBaseLabeler` a `TrainTestClassifier`, jež jsou součástí balíku `ui`.

4.3.2 TextBaseLabeler

TextBaseLabeler je nástroj, jenž převádí data do formátu, který je MinorThird schopen zpracovat. Jeho úkolem je přiřadit textovým dokumentům třídu, do níž

	jazyk	**poslední úpravy**	**licence**	**algoritmy**
Dragon Toolkit	Java	leden 2008	–	NB, SVM, Nigamovo aktivní učení
Lingpipe	Java	2011	*Royalty Free License Version 1*	NB, Bernoulliho model, KNN, LM, perceptrony, logistická regrese, *BigVector*
Mahout	Java	únor 2012	*Apache License*	NB, logistická regrese, *Random Forests*, *Online Passive Aggressive*
Mallet	Java	září 2011	CPL licence	NB, maximální entropie, rozhodovací stromy
MinorThird	Java	listopad 2008	BSD licence	NB, SVM, maximální entropie, KNN, rozhodovací stromy, perceptrony
OpenNLP	Java	listopad 2011	*Apache License*	maximální entropie
Stanford Classifier	Java	březen 2012	GNU licence	maximální entropie
SVMTorch II	C++	2001	BSD licence	SVM
TCatNG	Java	leden 2005	BSD licence	N-gramy, logistická regrese, SVM, bayesovské sítě, algoritmus založený na kompresi textu
Weka	Java	říjen 2011	GNU licence	NB, lineární a logistická regrese, SVM, rozhodovací stromy

Tabulka 4.1: Srovnání nalezených nástrojů pro klasifikaci textových dokumentů.

náleží, a uložit ji a další informace o dokumentu do speciálního souboru, pomocí kterého bude možné načíst do MinorThirdu tyto dokumenty jako zdrojová data. Struktura výsledného souboru (tzv. *labels file*) je následující:

```
addToType zprava1.txt 1 34 meteo
addToType zprava2.txt 1 13 vlady
addToType zprava3.txt 1 10 kultura
```

V druhém sloupci je uvedeno jméno souboru, ve třetím je pořadí příznaku, na kterém zpráva začíná. Na čtvrté pozici v řádku se nachází počet příznaků zprávy a na posledním místě pak třída, do níž zpráva náleží.

Nejsnazší způsob, jak potřebný soubor vytvořit, je mít každou zprávu ohraničenou XML elementem, jehož název odpovídá příslušné kategorii dokumentu, tedy

`<kategorie> vybrané_příznaky_zprávy </kategorie>`.

Nemají–li dokumenty tuto strukturu, je nutné pro každý kategorii „ručně“ nastavit, což lze realizovat pouze pro velmi malý objem zdrojových dat.

Jednotlivé parametry lze dokumentům přiřadit pomocí přehledného grafického uživatelského rozhraní (GUI), pomocí kterého lze TextBaseLabeler ovládat. Stručný návod je součástí přílohy A.2.

4.3.3 TrainTestClassifier

TrainTestClassifier slouží k samotné klasifikaci textových dokumentů. Nejprve proběhne proces učení na základě zvolených trénovacích dat a následně pak otestování vytvořeného modelu. Množinu testovacích dat si může přímo vybrat uživatel, či lze použít některou z funkcí nástroje, která určí testovací data za nás, přičemž jednou z možností je náhodné rozdělení dat na trénovací a testovací množinu a druhou variantou je křížová validace. Začleněny jsou i třídy pro vyhodnocení úspěšnosti klasifikace včetně přehledného grafického uživatelského rozhraní pro zobrazení výsledků. Jak použít TrainTestClassifier pro natrénování vlastních dat je obsahem přílohy A.3.

Trénování modelu a testování dat lze nástrojem MinorThird provést i odděleně. K samotnému trénování slouží nástroj **TrainClassifier**. Stručný návod k jeho použití lze nalézt v příloze A.4. Otestovat natrénovaný model je pak možné nástrojem **TestClassifier**, viz příloha A.5.

5 Řešení

Tato kapitola se skládá ze tří základních částí. V první z nich, v sekci 5.1, jsou popsány úpravy, které bylo nutné provést na zdrojovém korpusu. Následující část 5.2 je věnována návrhu experimentů a součástí sekce 5.3 jsou pak výsledky těchto experimentů a jejich zhodnocení.

5.1 Předzpracování dat

Obsahem této části práce je popis zdrojových dat, jejich úprav a způsobu, jakým byly vybrány dokumenty pro experimenty.

5.1.1 Zdrojová data

Zdrojová data potřebná pro trénování jednotlivých algoritmů a následné testování modelů poskytla Česká tisková kancelář (ČTK). Jednalo se o zpravodajství ČTK za leden 2011 ve formátu jednoho XML souboru. V něm bylo obsaženo celkem 20472 zpráv. Každá zpráva byla ohraničená elementem `<radek>`, v němž byly vnořené elementy `<titulek>`, `<datum>`, `<cas>`, `<lokalita>`, `<kw>`, `<hlavni_kategorie>`, `<kategorie>`, `<priorita>`, `<servis>` a `<zprava>`. Ukázka jedné zprávy je uvedena v příloze B.

5.1.2 Úpravy zdrojového souboru

Před použitím dat obsažených ve zdrojovém souboru bylo potřeba provést některé jeho úpravy. Pro tento účel bylo napsáno několik tříd v jazyce Java, či byly v editoru *Notepad++* pomocí regulárních výrazů upraveny některé nedostatky ručně.

V prvé řadě bylo třeba v elementech `<zprava>`, které jsou nejpodstatnějšími částmi souboru, odstranit některé nevhodné sekvence. Na začátku každé zprávy se vyskytovala krátká neúplná věta zakončená třemi tečkami kopírující titulek a zároveň první věta samotného textu zprávy. Dvojnásobný výskyt této části by mohl při klasifikaci zkreslit výsledky, a proto byla tato neúplná věta odstraněna. Dále se v textu nacházely XML entity jako například `&lt;`, `&gt;` či `&amp;`. Ty velmi často

zastupovaly HTML značky, které pro další použití nejsou významné. Vymazány byly také různé popisky, zkratky nebo oddělující čáry, jež nebyly součástí samotného textu.

Ve zdrojovém souboru bylo možné najít i zprávy v anglickém jazyce. Vzhledem k tomu, že bylo cílem pracovat výhradně s češtinou, bylo nutné je vynechat. Většina těchto zpráv spadala pod kategorie *Dayly News* se zkratkou *eng* a *Business News* se zkratkou *bns*, proto nebylo obtížné je vyhledat a následně odstranit. Jejich vymazáním se celkový počet zpráv souboru zredukoval na 18529.

5.1.3 Výběr trénovacích dat

Pro účely klasifikace byly důležité pouze samotné texty zpráv, tedy obsahy elementů `<zprava>`, spolu s kategorií, do které zpráva náleží. Kategorie byla určena dvěma elementy. Jedním z nich byl element `<hlavni_kategorie>`, v němž byla uvedena vždy právě jedna hlavní kategorie, druhým elementem potom `<kategorie>`, jenž obsahoval od jedné do pěti vedlejších kategorií. Vzhledem k jejich velkému počtu a skutečnosti, že MinorThird je stejně jako mnohé podobné nástroje zaměřen především na typ klasifikace, kdy dokument řadíme do jedné z n tříd, rozhodla jsem se po domluvě s vedoucím bakalářské práce s vedlejšími kategoriemi nepracovat a využívat pouze obsah elementu `<hlavni_kategorie>`.

Texty zpráv byly dále rozděleny do jednotlivých souborů na základě kategorií tak, aby každá zpráva byla uložená v samostatném souboru umístěném ve složce, jejíž název odpovídá zkratce pro danou třídu. Hlavních kategorií se v upraveném zdrojovém souboru vyskytovalo celkem 37, což tedy odpovídá počtu vytvořených adresářů. Jednotlivé třídy jsou spolu s počtem zpráv, které do nich náleží, zobrazeny v tab. 5.1.3. V tabulce není zahrnuta kategorie označena zkratkou *xhs*, která obsahovala zprávy skládající se většinou z jedné či dvou vět, velmi často kopírujícími nadpis. Tématika těchto zpráv byla navíc velmi široká, proto byla celá kategorie vyřazena, neboť pro účely klasifikace ji nelze použít.

Vzhledem k tomu, že v některých kategoriích byl nedostatečný počet zpráv pro úspěšné natrénovaní klasifikátoru, omezila jsem se na třídy s větším počtem než 200 dokumentů. Po domluvě s vedoucím bakalářské práce jsem dále klasifikovala do 10 tříd. 10 vybraných kategorií je spolu s jejich popisem vypsáno v tab. 5.2.

Název kategorie	**Zkratka**	**Počet zpráv**
Burzy	bur	698
Cestovní ruch	tur	75
Doprava	dpr	474
Energie	ene	189
Finanční služby	fin	223
Chemický a farmaceutický průmysl	che	43
Kriminalita a právo	zak	1493
Kultura	kul	620
Lehký průmysl	prm	55
Magazínový výběr	mag	412
Makroekonomika	mak	641
Média a reklama	med	89
Monitor	mnt	55
Náboženství	nab	56
Neštěstí a katastrofy	kat	26
Obchod	obo	121
Parlamenty a vlády	for	699
Plány a zpravodajství ČTK	pla	4
Počasí	met	814
Politika	pol	1583
Potravinářství	ptr	78
Práce a odbory	odb	26
Služby	slz	174
Sociální problematika	sop	179
Souhrn ekonomického zpravodajství	sue	119
Sportovní zpravodajství	spo	4287
Stavebnictví a reality	sta	149
Strojírenství	str	246
Školství	sko	100
Telekomunikace a IT	pit	198
Věda a technika	vat	86
Zdravotnictví	zdr	359
Zemědělství	zem	183
Zpravodajské deníky	den	926
Životní prostředí	ekl	127
Životní styl	spl	184

Tabulka 5.1: Všechny hlavní kategorie vyskytující se v již upraveném zdrojovém XML souboru.

Název kategorie (*zkrácený název*)	**Popis kategorie**
Doprava (*doprava*)	Informace o dopravě automobilové, železniční, letecké, lodní a městské hromadné a o budování dopravních staveb.
Kriminalita a právo (*krimi*)	Informace o právních normách a jejich dodržování. Zejména jde o zprávy o kriminalitě, jejím vyšetřování, souzení a trestání, dále o dopravních nehodách a různých neštěstích.
Kultura (*kultura*)	Informace o filmu, divadlu, hudbě, literatuře, výtvarném umění, architektuře, historických památkách, muzeích a kultuře obecně.
Magazínový výběr (*magazín*)	Zpravodajství ze světa zábavního průmyslu a módy, informace o celebritách, trendech a zajímavostech z různých oborů a populárně–naučné zdravotnické zprávy.
Makroekonomika (*makro*)	Informace o státních rozpočtech, dluzích, měnách, HDP, úrokových mírách a sazbách a o mezinárodních ekonomických institucích.
Počasí (*počasí*)	Informace o počasí, o sjízdnosti silnic, sněhových podmínkách, ale i záplavách, zemětřeseních apod.
Sportovní zpravodajství (*sport*)	Informace o sportovních soutěžích doma i v zahraničí. Obsahuje i výsledky tipovacích soutěží.
Strojírenství (*strojírenství*)	Informace o strojírenství, automobilovém, leteckém, lodním, zbrojním a elektrotechnickém průmyslu a o železárnách a hutích.
Parlamenty a vlády (*vlády*)	Informace o činnosti hlav států, parlamentů a vlád.
Zdravotnictví (*zdravotnictví*)	Informace o zdraví, léčení chorob, lékárnách, lécích a o zdravotnických systémech včetně zdravotních pojišťoven a zdravotnických profesních a odborových organizací.

Tabulka 5.2: Kategorie vybrané pro testování klasifikátoru.

5.1.4 CoNLL

Aby bylo možné jednotlivá slova tvořící zprávy později lemmatizovat a určit slovní druh, bylo nutné zprávy převést do formátu CoNLL–2009. Ten vyžaduje, aby každé slovo bylo umístěno na samostatném řádku, který začíná číslem označujícím pořadí slova ve větě. V každé větě se tedy začíná číslovat znovu od jedničky. Následuje samotné slovo a za ním pak 13 podtržítek. Vše je odděleno tabulátorem. Mezi jednotlivými větami musí být prázdný řádek. Například zpráva „*Zítra se výrazně ochladí. Na většině území bude sněžit.*“ by vypadala takto:

```
1 Zítra _ _ _ _ _ _ _ _ _ _ _ _ _
2 se _ _ _ _ _ _ _ _ _ _ _ _ _
3 výrazně  _ _ _ _ _ _ _ _ _ _ _ _ _
4 ochladí _ _ _ _ _ _ _ _ _ _ _ _ _
5 . _ _ _ _ _ _ _ _ _ _ _ _ _

1 Na _ _ _ _ _ _ _ _ _ _ _ _ _
2 většině _ _ _ _ _ _ _ _ _ _ _ _ _
3 území _ _ _ _ _ _ _ _ _ _ _ _ _
4 bude _ _ _ _ _ _ _ _ _ _ _ _ _
5 sněžit _ _ _ _ _ _ _ _ _ _ _ _ _
6 . _ _ _ _ _ _ _ _ _ _ _ _ _
```

5.1.5 Lemmatizátor a POS–tagger

Jako vstupní data pro klasifikátor jsem se na základě dobrých výsledků v literatuře [24] rozhodla použít lemmata místo slov, proto bylo potřeba provést lemmatizaci. Dále jsem předpokládala, že pro klasifikaci dokumentů nebude nutné pracovat se všemi slovy obsaženými v textu, ale pouze s vybranými slovními druhy [23]. Pro účel přiřazení slovního druhu lemmatům slouží POS–tagger. Oba nástroje, lemmatizátor i POS–tagger s již natrénovaným modelem pro češtinu, poskytl vedoucí bakalářské práce včetně spouštěcích skriptů.

Vstupem lemmatizátoru jsou soubory ve formátu CoNLL, který byl představen v sekci 5.1.4. Lemmatizátor do tohoto formátu doplní lemmata slov a jeho výsledek pak slouží jako vstup pro POS–tagger, který dodá zkratku slovního druhu. Přehled zkratek pro jednotlivé slovní druhy je zobrazen v tab. 5.1.5. Po lemmatizaci a otaggování vypadá výše zmíněná vzorová zpráva „*Zítra se výrazně ochladí. Na většině území bude sněžit.*“ takto:

```
1 Zítra zítra _ _ D _ _ -1 _ _ _ _ _
2 se se_^(zvr._zájmeno/částice) _ _ P _ _ -1 _ _ _ _ _
3 výrazně  výrazně _^(*1ý) _ _ N _ _ -1 _ _ _ _ _
4 ochladí ochladit_:w _ _ V _ _ -1 _ _ _ _ _
5 . . _ _ Z _ _ -1 _ _ _ _ _

1 Na na-1 _ _ R _ _ -1 _ _ _ _ _
2 většině většina _ _ N _ _ -1 _ _ _ _ _
3 území území _ _ N _ _ -1 _ _ _ _ _
4 bude být _ _ V _ _ -1 _ _ _ _ _
5 sněžit sněžit_:w _ _ V _ _ -1 _ _ _ _ _
6 . . _ _ Z _ _ -1 _ _ _ _ _
```

První sloupec označuje pořadí ve větě. Druhý sloupec obsahuje slovo v původním tvaru nebo interpunkční znak. Ve třetím sloupci najdeme lemma. Zkratka slovního druhu se nachází v šestém sloupci. Ostatní sloupce nejsou využity.

Zkratka	**Význam**
N	podstatné jméno
A	přídavné jméno
P	zájmeno
C	číslovka
V	sloveso
D	příslovce
R	předložka
J	spojka
T	částice
I	citoslovce
Z	interpunkční znaménko

Tabulka 5.3: Zkratky vyskytující se v otaggovaném CoNLL souboru.

5.1.6 Finální úpravy

Na základě zvolených kritérií pro výběr příznaků, která jsou dále popsána v sekci 5.2, byla z otaggovaných souborů extrahována daná lemmata či původní slova a umístěna do nových souborů, které byly ukládány do složek podle kategorií.

V zápětí se ukázalo, že pro použití nástroje *TextBaseLabeler*, jenž jednotlivým zprávám přiřadí správnou kategorii, je vhodné, aby byl text každé zprávy ohraničen elementem, jehož název odpovídá názvu třídy, do které zpráva náleží. Na začátek každého souboru tak byl přidán příslušný otevírací tag a na jeho závěr tag ukončovací. Nakonec bylo z každé kategorie náhodně vybráno 200 souborů určených pro trénování a umístěno do jednoho adresáře, čímž byla příprava trénovacích dat pro nástroj MinorThird hotova a bylo možné přistoupit k experimentům.

5.2 Návrh experimentů

Pro experimenty bylo vedle výběru klasifikačních algoritmů patrně nejdůležitějším krokem zvolit kritéria pro výběr příznaků a na jejich základě pak vytvořit několik různých skupin zdrojových dokumentů, které budou použity jako trénovací data pro klasifikátor.

Lemmatizátor a POS–tagger byly použity, aby bylo možno jako trénovací data vybrat pouze určité slovní druhy. Z těch jsem zvolila podstatná a přídavná jména, slovesa a příslovce. Tyto slovní druhy patří mezi základní, neboť je lze odvodit přímo z pojmů našeho myšlení [17]. Naopak jsem předpokládala, že bude možné vynechání zájmen, předložek, spojek, částic, citoslovcí a také interpunkčních znamének. Ke zvážení, zdali je začlenit či nikoliv, zbývaly pouze číslovky. Zahrnutí konkrétních číslovek by nejspíše bylo zbytečné, neboť není pravděpodobné, že by ojedinělý výskyt některého čísla přispěl k lepšímu natrénování klasifikátoru. Na druhou stranu počet číslovek ve zprávě by úlohu hrát mohl. Z toho důvodu jsem dospěla k závěru, že vyzkouším obě varianty. Vytvořila jsem tedy dvě skupiny trénovacích dat, přičemž v jedné jsem číslovky vynechala zcela a do druhé jsem je začlenila. Konkrétní číslovky jsem však nahradila slovem *číslovka*. Z vícenásobného výskytu tohoto slova ve zprávě tak bude jasné, kolik čísel text obsahoval. Jako příznaky pro tyto dvě skupiny jsem se rozhodla použít lemmata slov, neboť jsem předpokládala, že takto bude klasifikátor podávat lepší výsledky než s použitím slov v jejich původním tvaru.

Aby bylo možné ověřit, zdali použití pouze určitých slovních druhů zlepšuje úspěšnost klasifikátoru, bylo vhodné, aby další vytvořená skupina dokumentů naopak obsahovala všechny slovní druhy. Třetí soubor dat je tedy složen z dokumentů, které jsou reprezentovány všemi slovy nacházejícími se v originálním textu, pouze jsou na místo původních tvarů slov použita jejich lemmata. Pro čtvrtou, poslední, skupinu dokumentů pak byla použita slova v původních tvarech, což by mělo poskytnout dobré srovnání s předchozí skupinou a zároveň umožnit zhodnocení významu lemmatizace a POS–taggingu.

Vznikly tedy celkem 4 skupiny dokumentů lišící se vybranými příznaky a určené pro klasifikaci nástrojem MinorThird. Cílem práce bylo použít k tomu tři různé algoritmy, přičemž vybrány byly naivní Bayesův klasifikátor, support vector machines a maximální entropie. Úspěšnost vytvořeného klasifikátoru pro tyto metody a pro všechny soubory dokumentů jsou obsaženy v kap. 5.3.

5.3 Dosažené výsledky

Pro testování natrénovaných modelů byla použita křížová validace, která je součástí implementace nástroje MinorThird. Důležitým parametrem pro tuto metodu je počet částí, do kterých se mají zdrojová data rozdělit. Zvolena byla hodnota 10. Výsledky pro jednotlivé skupiny vstupních dokumentů jsou zobrazeny v tabulkách podle použitého algoritmu pro trénování. Tab. 5.4 obsahuje výsledky pro naivní Bayesův klasifikátor, tab. 5.5 pro metodu SVM a tab. 5.6 pro algoritmus založený na maximální entropii. V těchto tabulkách jsou začleněny výsledky pro jednotlivé kategorie i celkové. Číselné hodnoty udávají, pro kolik procent dokumentů určil klasifikátor třídu správně. Řádky tabulek odpovídají rozpoznávaným kategoriím a sloupce jednotlivým sadám dokumentů lišících se výběrem příznaků. První sloupec s hodnotami obsahuje výsledky pro skupinu dokumentů, kde byla použita všechna slova. V druhém sloupci se nachází výsledky pro skupinu dokumentů, které jsou reprezentovány všemi lemmaty. Pro sadu dokumentů, jejíž výsledky jsou ve třetím sloupci tabulky, byly vybrány pouze určité slovní druhy. Výsledky poslední skupiny dokumentů, ve které byly k vybraným slovním druhům přidány číslovky, se nachází ve čtvrtém sloupci.

Z tab. 5.4 vyplývá, že nejlepších výsledků dosahoval naivní Bayesův klasifikátor, a to s výraznějším průměrným odstupem 3,9 %. Na druhém místě je pak algoritmus support vector machines a nejhorších výsledků dosahoval algoritmus maximální entropie. Rozdíl výsledných průměrných úspěšností je zde však už jen 2 %. Pořadí algoritmů zůstane stejné i v případě, že porovnáme jejich výsledky pro každou skupinu dokumentů zvlášť. Jedinou výjimkou je skupina, ve které jsou jako příznaky použita všechna slova. Zde dopadla maximální entropie lépe než SVM. Hlavním důvodem vysoké úspěšnosti naivního Bayesova klasifikátoru oproti metodě support vector machines a algoritmu maximální entropie je pravděpodobně nedostatečná velikost trénovacích dat. Zatímco pro Bayesovu metodu byla velikost trénovacích dat nejspíše dostačující, pro další dva algoritmy byla pravděpodobně příliš malá.

Zajímavá zjištění vyplývají ze srovnání jednotlivých skupin dokumentů. Předpokladem bylo, že soubor dokumentů, kde jsou jako příznaky zvolena všechna slova

	všechna slova	**všechna lemmata**	**lemmata vybraných slovních druhů**	**lemmata vybraných slovních druhů včetně číslovek**
doprava	79,5	78,0	78,5	79,0
krimi	83,0	82,0	82,0	82,0
kultura	89,5	91,5	92,0	91,0
magazin	66,0	69,5	67,5	70,0
makro	87,0	86,0	85,5	86,5
meteo	97,0	95,5	96,5	97,0
sport	96,0	92,0	93,5	93,0
strojírenství	94,5	95,5	94,5	96,5
vlády	85,5	84,0	87,0	84,0
zdravotnictví	91,0	91,5	92,5	93,5
celkem	**86,9**	**86,6**	**87,0**	**87,2**

Tabulka 5.4: Výsledky pro NB (úspěšnost klasifikátoru v %).

	všechna slova	**všechna lemmata**	**lemmata vybraných slovních druhů**	**lemmata vybraných slovních druhů včetně číslovek**
doprava	81,5	78,5	79,5	82,0
krimi	83,5	75,5	80,5	80,0
kultura	80,0	79,0	81,0	79,5
magazin	76,0	75,0	76,0	75,0
makro	81,5	78,0	81,0	86,0
meteo	93,5	94,5	95,0	95,5
sport	89,5	89,5	89,0	89,5
strojírenství	89,0	91,5	89,5	90,5
vlády	77,5	76,5	79,0	74,0
zdravotnictví	81,5	77,5	83,0	82,5
celkem	**83,4**	**81,6**	**83,4**	**83,5**

Tabulka 5.5: Výsledky pro SVM (úspěšnost klasifikátoru v %).

	všechna slova	**všechna lemmata**	**lemmata vybraných slovních druhů**	**lemmata vybraných slovních druhů včetně číslovek**
doprava	81,5	68,0	78,0	73,0
krimi	81,5	71,0	80,5	81,0
kultura	80,5	67,5	80,0	81,0
magazin	74,5	59,5	72,5	70,5
makro	83,0	78,0	82,5	83,0
meteo	96,5	90,5	92,0	91,0
sport	93,5	87,5	92,5	87,5
strojírenství	92,5	84,0	92,0	92,5
vlády	80,0	68,0	75,0	76,0
zdravotnictví	81,5	75,0	83,5	82,0
celkem	**84,5**	**74,9**	**82,9**	**81,8**

Tabulka 5.6: Výsledky pro ME (úspěšnost klasifikátoru v %).

v původním tvaru, si povede nejhůře a že v případě použití lemmat slov, bude mít klasifikátor úspěšnost větší. Nejlepší výsledky měl podávat klasifikátor natrénovaný jen na lemmatech vybraných slovních druhů. Tyto předpoklady se však nepotvrdily. Nejhorších výsledků ve všech třech případech dosáhla skupina dokumentů, která má jako příznaky všechna lemmata. To lze vidět zejména v případě maximální entropie, kde byla úspěšnost klasifikátoru až překvapivě malá oproti ostatním. Nejlepších výsledků zde naopak dosáhla skupina používající všechna slova. U dalších dvou algoritmů, NB a SVM, pak skutečně sady souborů s pouze určitými slovními druhy měly nejvyšší úspěšnost, přičemž soubor obsahující navíc číslovky dosáhl ještě nepatrně lepších výsledků. Rozdíly v hodnotách celkové úspěšnosti jsou však příliš malé. Výsledky tedy nepotvrdily předpoklady, že použití POS–taggeru a výběr jen některých slovních druhů přispějí k vytvoření lepšího klasifikátoru. Hlavní příčinou je pravděpodobně skutečnost, že byla použita nejjednodušší parametrizační technika (*document frequency*). Použití sofistikovanější metody by pravděpodobně vedlo k tomu, že by se výsledky experimentů více přiblížily očekávaným výsledkům.

Podíváme–li se na výsledky jednotlivých kategorií, nejlepších dosáhly klasifikátory pro kategorii *meteo*, což je pravděpodobně dáno tím, že disponuje specifickou slovní zásobou, jež se pouze minimálně překrývá se zbylými třídami. Velmi dobré výsledky podávaly klasifikátory také pro kategorie *sport* a *strojírenství*. Naopak nejhůře rozpoznatelná byla třída *magazín*, pro kterou je až na jedinou výjimku

úspěšnost klasifikátoru vždy nejnižší ze všech kategorií. Při bližším prozkoumání bylo zjištěno, že tato kategorie byla velmi podobná zejména kategorii *kultura*, která z toho důvodu byla často zvolena namísto správné třídy *magazín*.

6 Závěr

Cílem této práce bylo nastudovat používané techniky pro automatickou klasifikaci textových dokumentů a zvolit alespoň dvě metody, které budou otestovány na českém korpusu, jenž dodala ČTK. K tomu bylo potřeba vyhledat vhodný klasifikační nástroj, což tvořilo další cíl této práce.

Na základě předchozí analýzy existujících přístupů byly pro vlastní experimenty jako metody zvoleny naivní Bayesův klasifikátor, support vector machines a algoritmus maximální entropie. K této volbě pak bylo nutné přihlédnout při výběru nástroje, který bude použit pro vlastní klasifikaci dokumentů. Na internetu jich bylo nalezeno celkem deset, jež vyhovovaly zadaným kritériím. Mezi kritéria patřilo především, aby se jednalo o volně dostupný software, který je implementován v jazyce Java, případně C++. Z nalezených nástrojů byl nakonec zvolen MinorThird především proto, že jsou v něm zahrnuty všechny tři vybrané algoritmy a že disponuje kvalitní dokumentací.

Zdrojové texty byly upraveny do potřebné podoby pro nástroj MinorThird a byly vytvořeny čtyři skupiny dokumentů, které se lišily typem zvolených příznaků. První sada obsahovala dokumenty, které měly jako příznaky všechna slova původního textu v jejich originálním tvaru. Pro druhý soubor byla použita také všechna slova, ovšem jejich původní tvar byl nahrazen jejich lemmatem. Lemmata byla součástí i dalších dvou souborů, jež obsahovaly vybrané slovní druhy, kterými byly podstatná a přídavná jména, slovesa a příslovce. V jedné sadě se ve speciálním tvaru nacházely i číslovky.

Pro všechny čtyři soubory dokumentů a všechny tři zvolené algoritmy byly jednotlivě provedeny experimenty a v závěru tak vzniklo celkem dvanáct sad výsledků. Z výsledků vyplynulo, že jako nejlepší metoda se jeví naivní Bayesův klasifikátor, který přiřadil správnou kategorii pro nejvyšší procento dokumentů. S mírným odstupem skončil na druhém místě algoritmus SVM a nejnižší úspěšnost zaznamenala metoda maximální entropie.

Výsledky pro jednotlivé sady dokumentů nebyly ve shodě s prvotními předpoklady. Očekávalo se, že soubor obsahující jako příznaky všechna slova dosáhne nejhorších výsledků. Nejnižší úspěšnost měly natrénované klasifikátory ovšem pro skupinu, která obsahovala jako příznaky všechna lemmata. Dalším předpokladem bylo, že skupiny, kde byly začleněny jen některé slovní druhy dosáhnou nejlepších výsledků. Tak tomu bylo pouze v případě naivního Bayesova klasifikátoru a SVM. Rozdíl oproti ostatním skupinám byl však minimální. Pro algoritmus maximální

entropie měla nejvyšší úspěšnost skupina obsahující všechna slova původního textu.

Výsledky experimentů potvrdily, jak důležitým krokem je výběr příznaků. Vyzkoušení dalších metod pro volbu příznaků s cílem najít tu nejvýhodnější je jedna z variant budoucího vývoje. Dalšími možnostmi jsou ověření výsledků s rozsáhlými daty a automatizace procesu parametrizace. Kromě zdokonalení systému by pak bylo možné jeho rozšíření tím způsobem, že by jednomu dokumentu mohlo být přiřazeno více kategorií, což by umožnilo aplikaci v těch oblastech, kde se témata mohou překrývat.

Přehled zkratek

BSD	*Berkeley Software Distribution* (licence)
CoNLL	*Computer Natural Language Learning*
CPL	*Common Public License*
ČTK	Česká tisková kancelář
GNU	*General Public License*
GUI	Grafické uživatelské rozhraní
HMM	*Hidden Markov Models*
KNN	*k-Nearest Neighbor* (algoritmus)
LM	*Language Model* (algoritmus)
ME	Maximální entropie
NB	Naivní Bayesův klasifikátor
PDT	Pražský závislostní korpus
POS	*Part of Speech*
SVM	*Support Vector Machines*

Literatura

[1] *Apache Mahout: Scalable machine learning and data mining* [online]. c2011, [cit. 2012–04–11]. URL <http://mahout.apache.org>.

[2] *Apache OpenNLP – Welcome to Apache OpenNLP* [online]. c2010, [cit. 2012–04–11]. URL <http://opennlp.apache.org/>.

[3] *Dragon Toolkit* [online]. [cit. 2012–04–11]. URL <http://dragon.ischool.drexel.edu>.

[4] *index [Torch7]* [online]. [cit. 2012-05-06]. URL <http://www.torch.ch>.

[5] Kapitola 8. Koreference. *Anotace na tektogramatické rovině Pražského závislostního korpusu: Anotátorská příručka* [online]. [cit. 2012–05–06]. URL <http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/cz/t-layer/html/ch08.html>.

[6] *LingPipe Home* [online]. c2003-2011, [cit. 2012–04–11]. URL <http://alias-i.com/lingpipe/>.

[7] *MALLET homepage* [online]. c2011, [cit. 2012–04–11]. URL <http://mallet.cs.umass.edu>.

[8] *Support vector machines (SVM): Algoritmy podpůrných vektorů* [online]. posl. revize 9. 12. 2004 [cit. 2012–04–25]. URL <http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf> .

[9] *SVMTorch* [online]. [cit. 2012–04–11]. URL <http://bengio.abracadoudou.com/SVMTorch.html>.

[10] *TCatNG Toolkit:: Text Categorization via N–Grams* [online]. c2004, [cit. 2012–04–11]. URL <http://tcatng.sourceforge.net>.

[11] *Bayesovská klasifikace* [online]. [cit. 2012-04-25]. URL <http://sorry.vse.cz/~berka/docs/izi456/kap_5.6.pdf>.

[12] *Český národní korpus* [online]. [cit. 2012–04–25]. URL <http://ucnk.ff.cuni.cz/>.

[13] *Pražský závislostní korpus 2.0.* [online]. c2006, [cit. 2012–04–25]. URL <http://ufal.mff.cuni.cz/pdt2.0/>.

[14] *Weka* [online]. [cit. 2012–04–11]. URL <http://weka.wikispaces.com>.

[15] BERGER, Adam L. - PIENTRA, Vincent J. Della - PIENTRA, Stephen A. Della. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, March 1996. ISSN 0891-2017.

[16] COHEN, William W. *MinorThird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data*, 2004. URL <http://minorthird.sourceforge.net>.

[17] ČECHOVÁ, Marie, a kol. *Čeština – řeč a jazyk*. 2. vyd. Praha: ISV nakladatelství, 2000. ISBN 80-85866-57-9.

[18] JURAFSKY, Daniel - MARTIN, James H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2. vyd. Upper Saddle River: Prentice Hall, 2009. ISBN 0135041961.

[19] MACCARTNEY, Bill. *The Standford NLP (Natural Language Processing) Group* [online]. [cit. 2012–04–11]. URL <http://nlp.stanford.edu/software/classifier.shtml>.

[20] MANNING, Christopher D. – RAGHAVAN, Prabhakar – SCHÜTZE, Hinrich. *Introduction to Information Retrieval*. New York: Cambridge University Press, 2008. ISBN 0521865719.

[21] MANNING, Christopher D. – SCHÜTZE, Hinrich. *Foundations of Statistical Natural Language Processing*. 2. vyd. Cambridge (Massachusetts): MIT Press, 1999. ISBN 0–262–13360–1.

[22] SEBASTIANI, Fabrizio. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. ISSN 0360-0300.

[23] SHAH, Chirag – BHATTACHARYYA, Pushpak. A Study for Evaluating the Importance of Various Parts of Speech (POS) for Information Retrieval (IR). In *Proceedings of International Conference on Universal Konwledge and Language (ICUKL)*, 2002.

[24] ZELAIA, A. – ALERGIA – I. – ARREGI, O. – SIERRA, B. Analying the Effect of Dimensionality Reduction in Document Categorization for Basque. In *Proceedings of L&TC´05: 2nd Language & Technology Conference*, 72–75, 2005.

A Uživatelská dokumentace

A.1 MinorThird - kompilace

Pro používání nástroje MinorThird je nutné mít nainstalovanou javu verze 1.5.0 a vyšší. Pro kompilaci kódu je navíc potřeba *Ant* verze 1.6.5 nebo pozdější.

Postup instalace:

1. Stáhnout MinorThird z adresy uvedené u zdroje [16].
2. Pomocí příkazu `cd` se přesunout do adresáře, do kterého byly rozbaleny stažené soubory.
3. Spustit instalační skript pro daný operační systém pro nastavení CLASSPATH.
   - pro Linux: `source script/setup.linux`
   - pro Windows: `script\setup`
4. Aby bylo možné používat veškeré součásti nástroje MinorThird, je nutné provést drobnou úpravu zdrojového kódu. Jedná se o doplnění českých znaků, jež by jinak MinorThird považoval za bílé znaky a v důsledku toho by docházelo mimo jiné ke špatnému dělení jednotlivých slov.
   V adresáři `src/edu/cmu/minorthird/text/` vyhledáme třídu

   `RegexTokenizer.java`

   a zobrazíme ji v libovolném textovém editoru. Zde najdeme globálně deklarovaný řetězec

   `TOKEN_REGEX_DEFAULT_VALUE`,

   jenž je inicializován na hodnotu

   `"\\s*([0-9]+|[a-zA-Z]+|\\W)\\s*"`.

   Do toho je nutné začlenit veškerá písmena obsahující háčky a čárky vyskytující se v české abecedě a to v jejich malé i velké variantě. Výsledek může vypadat např. takto:

   `"\\s*([0-9]+|[a-zA-Zěščřžýáíéňd’t’úůĚŠČŘŽÝÁÍÉŇĎŤÚŮ]+|\\W)\\s*"`

5. Po uložení a přesunutí se zpět do hlavního adresáře lze kód zkompilovat příkazem:

   ```
   ant build-clean
   ```

6. Dokumentaci lze vygenerovat příkazem:

   ant javadoc

7. Otestovat, zdali kompilace byla úspěšná, lze příkazem:

   ```
   ant test
   ```

V případě problémů s kompilací je možné z uvedeného zdroje stáhnout také spustitelnou, tedy již zkompilovanou, verzi. Je však nutné pamatovat na fakt, že bez výše znázorněných úprav třídy `RegexTokenizer.java` nebude nástroj pro češtinu spolehlivě fungovat.

A.2 TextBaseLabeler

Postup vytvoření souboru **DATA.labels** potřebného pro načtení zdrojových dokumentů do nástroje MinorThird je následující:

1. Na příkazové řádce se přesuneme do složky, kam jsme nainstalovali MinorThird. TextBaseLabeler spustíme příkazem

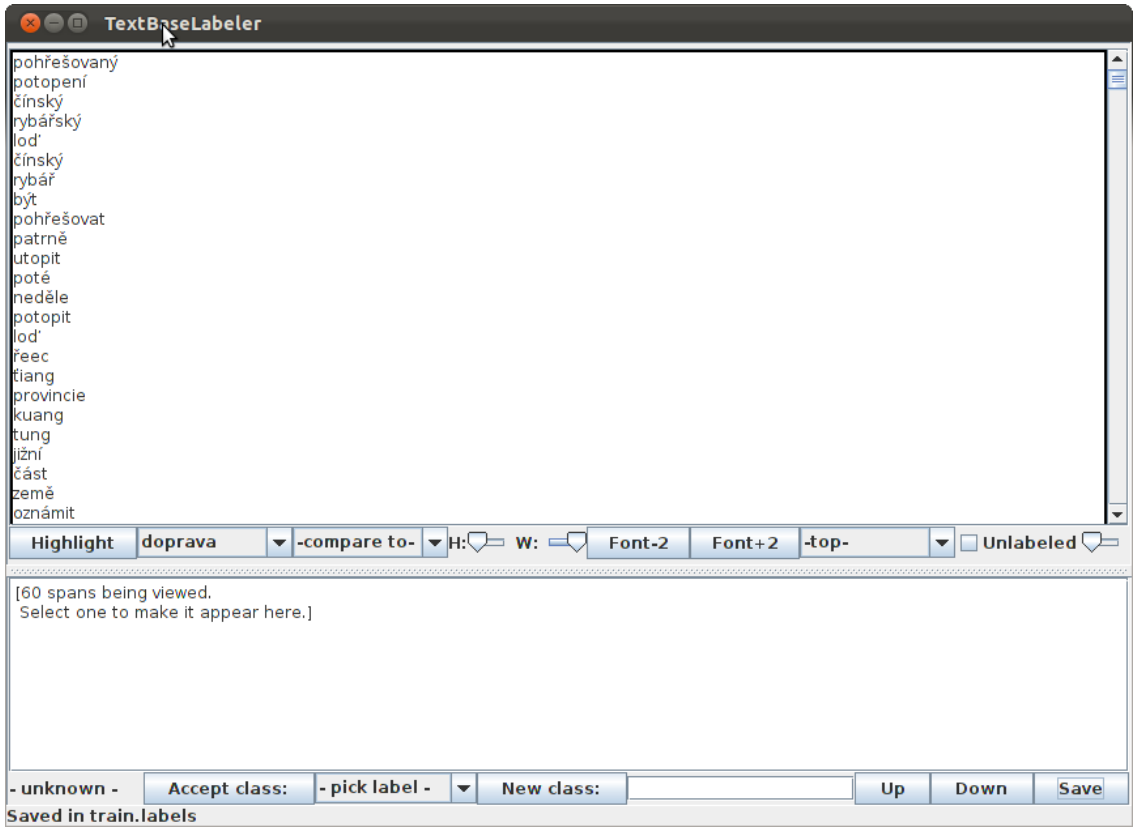
   ```
   java -Xmx500M edu.cmu.minorthird.text.gui.TextBaseLabeler DATA
   DATA.labels
   ```

   kde `DATA` je název adresáře obsahujícího zdrojové dokumenty. Je vhodné pojmenovat výstupní soubor shodně jako adresář s daty, protože tak bude MinorThird načítat zdrojové dokumenty automaticky.

2. Otevře se nové okno, viz obr. A.1, kde klikneme na tlačítko **Save** v pravém dolním rohu, čímž je soubor **DATA.labels** hotov.

3. Pokud nemáme zprávy ohraničené počátečními a ukončovacími XML značkami, viz kap. 4.3.2, musíme kategorii zvolit pro každý dokument zvlášť:

   - vybereme dokument (klikneme na něj v horní části okna)
   - zvolíme kategorii v menu **pick label** (dolní lišta)
   - není-li kategorie v nabídce, napíšeme ji do pole **New class** a klikneme na tlačítko **Accept class**
   - stejným způsobem vybereme třídu pro další dokumenty a nakonec práci uložíme volbou **Save**



Obrázek A.1: GUI nástroje TextBaseLabeler.

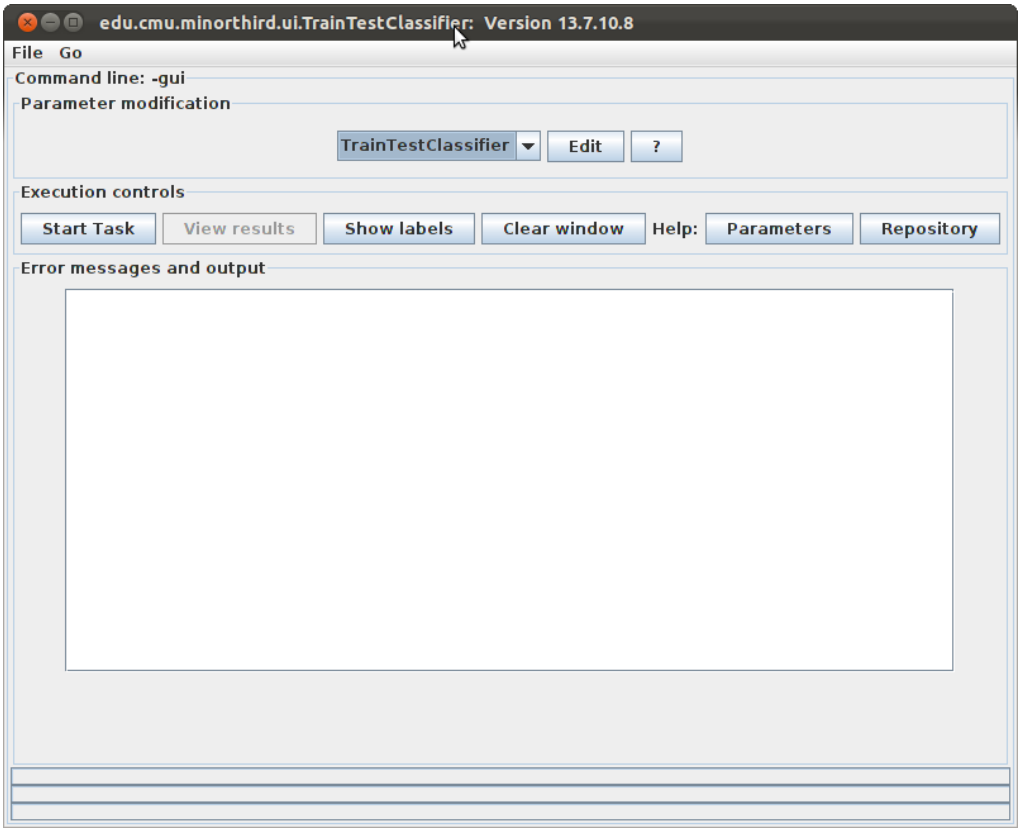
A.3 TrainTestClassifier

Postup pro natrénování modelu a jeho následné otestování pomocí GUI je následující:

1. Na příkazové řádce se přesuneme do složky, kam jsme nainstalovali Minor-Third. TrainTestClassifier spustím příkazem:

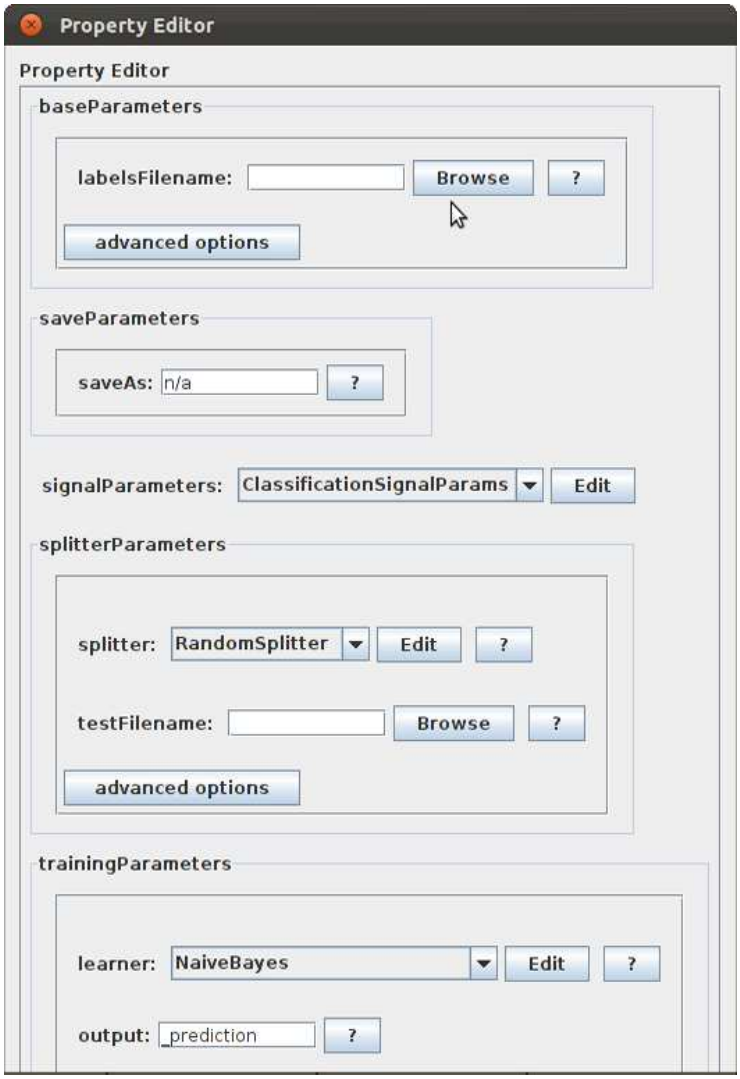
   ```
   java -Xmx500M edu.cmu.minorthird.ui.TrainTestClassifier -gui
   ```

2. Otevře se nové okno, viz obr. A.2, kde klikneme na tlačítko **Edit** umístěné v horní části.



Obrázek A.2: GUI nástroje TrainTestClassifier.

3. V dalším okně, které vidíme na obr. A.3 nastavíme veškeré parametry pro trénování a testování dat:



Obrázek A.3: Okno pro nastavení parametrů.

- **labelsFilename**: Kliknutím na tlačítko **Browse** vybereme soubor s koncovkou **.labels**, který jsme vytvořili pomocí nástroje TextBaseLabeler a který specifikuje dokumenty pro trénování (a případně i testování). Tyto dokumenty musí být umístěny v jedné složce, která se nachází ve stejném adresáři jako tento soubor.
- **signalParameters**: Klikneme na tlačítko **Edit** a jako **spanProp** nastavíme výběrem v menu možnost **userLabel**. Tato volba upřesňuje kategorie, pro které má být klasifikátor natrénován.
- **splitter**: Zde specifikujeme, jak budou rozdělena naše data na trénovací a testovací množinu. Jinou variantou je v poli **testFilename** vybrat testovací množinu přímo. V tomto případě bude splitter ignorován.
- **learner**: Pro případ multiklasifikace je nutné zvolit možnost **OneVsAllLearner** a kliknutím na tlačítko **Edit** v poli **innerLearner** zvolit požadovaný algoritmus.

4. Stisknutím klávesy Enter se okno *Property Editor* uzavře a my se dostaneme zpět do původní nabídky. Zde kliknutím na tlačítko **Start Task** zahájíme klasifikaci a trénování.

5. Po natrénování a otestování modelu si můžeme prohlédnout výsledky kliknutím na tlačítko **View results**.

Nechceme-li či nemůžeme-li použít GUI, je možné spustit TrainTestClassifier s využitím příkazové řádky. V tomto případě zadáme veškeré naše volby jako parametry. Nápovědu, která obsahuje přehled možností, lze vypsat příkazem:

```
java -Xmx500M edu.cmu.minorthird.ui.TrainTestClassifier -help
```

Například spustit TrainTestClassifier pomocí příkazové řádky s volbami

- **labelsFilename**: data.labels (včetně cesty)
- **spanProp**: userLabel
- **splitter**: CrossValSplitter (zdrojová data rozdělíme na 10 skupin)
- **learner**: OneVsAllLearner
- **innerLearner**: MaxEnt (maximální entropie - defaultní)

lze příkazem

```
java -Xmx500M edu.cmu.minorthird.ui.TrainTestClassifier -labels data
-spanProp userLabel -splitter "new CrossValSplitter()" -SplitterOp
numberOfFolds=10 -learner "new Recommended.OneVsAllLearner()"
```

Výsledkem bude stručný výpis výsledků zahrnující spočtenou chybovost celkově a pro každou třídu zvlášť. Pro podrobnější výpis ve speciálním okně je nutné při spouštění přidat ještě parametr `-showResult`.

A.4 TrainClassifier

Pro natrénování klasifikátoru je nutné zadat tyto parametry:

- **learner**: Pro klasifikaci do více než dvou tříd vybereme *OneVsAllLearner* a algoritmus upřesníme volbou **innerLearner**.
- **labels**: Název souboru s koncovkou *.labels* specifikující dokumenty pro trénování.
- **spanProp**: Zvolíme možnost *userLabel* pro trénování námi určených kategorií.
- **saveAs**: Název, pod kterým se má uložit natrénovaný model.

TrainClassifier můžeme spustit pomocí příkazové řádky nebo pomocí GUI.

Pomocí **příkazové řádky** můžeme trénování spustit např. tímto příkazem:

```
java -Xmx500M edu.cmu.minorthird.ui.TrainClassifier -labels data
-spanProp userLabel -learner "new Recommended.OneVsAllLearner()"
-LearnerOp innerLearner=ui.Recommended.NaiveBayes -saveAs model.ann
```

Pro spuštění pomocí **GUI** zadáme:

```
java -Xmx500M edu.cmu.minorthird.ui.TrainClassifier -gui
```

Další postup je pak shodný jako pro nástroj TrainTestClassifier, viz. příloha A.3. Liší se pouze množství zadávaných parametrů.

A.5 TestClassifier

Pro testování klasifikátoru je potřeba upřesnit tyto parametry:

- **labels**: Název adresáře obsahující dokumenty pro testování.
- **spanProp**: Zvolíme možnost *userLabel* pro klasifikaci do námi určených kategorií.
- **loadFrom**: Parametr pro upřesnění natrénovaného modelu, který se má načíst.
- **saveAs**: Jméno souboru pro uložení výsledku testování, nepovinný parametr.

TestClassifier můžeme spustit pomocí příkazové řádky nebo pomocí GUI.

Pomocí **příkazové řádky** můžeme testování spustit např. tímto příkazem:

```
java -Xmx500M edu.cmu.minorthird.ui.TestClassifier -labels data
-spanProp userLabel -loadFrom model.ann -saveAs data.labels
```

Pro spuštění pomocí **GUI** zadáme:

```
java -Xmx500M edu.cmu.minorthird.ui.TestClassifier -gui
```

Další postup je pak shodný jako pro nástroj TrainTestClassifier, viz. příloha A.3. Liší se pouze množství zadávaných parametrů.

B Ukázka zprávy ze zdrojového XML souboru

<radek id=”T201101010012301”>
<titulek>Tornádo v USA zabilo šest lidí</titulek>
<datum>01.01.2011</datum>
<cas>00:09</cas>
<lokalita>USA </lokalita>
<kw>USA počasí tornádo </kw>
<hlavni_kategorie>met </hlavni_kategorie>
<kategorie>met </kategorie>
<priorita>4</priorita>
<servis>mus</servis>
<zprava> Tornádo v USA zabilo šest lidí <p>Washington 1. ledna (ČTK) - Šest lidí zahynulo v pátek v USA, když se přes státy Arkansas, Missouri a Illinois přehnalo tornádo. Desítky osob utrpěly zranění. Vítr také zničil několik budov.</p> <p>Tornádo se přehnalo přes arkansaský okres Washington County, kde smetlo jeden zděný dům, stodolu a jeden montovaný domek. V troskách zahynuli tři lidé a několik dalších bylo zraněno. Vítr také zpřetrhal elektrické vedení.</p> <p>Další tři oběti jsou z oblasti Rolla ve státě Missouri. Dvě ženy zahynuly na jedné farmě složené z mobilních domů.</p> <p>Tornádo vzniklo na okraji frontálního systému, který ve čtvrtek přinesl sněhovou bouři do západních amerických států a pohybuje se k severovýchodu. Varování před dalšími větrnými smrštěmi dnes vyhlásily státy Arkansas, Missouri a Oklahoma.</p> <p>rv str</p> </zprava>
</radek>