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

Diplomová práce

Automatická klasifikace dokumentů s podobným obsahem

Plzeň, 2012 Michal Hrala

Poděkování

Děkuji Ing. Pavlu Královi, Ph.D. vedoucímu této diplomové práce za jeho ochotu, čas a cenné připomínky k obsahu i zpracování. Dále mé díky patří všem, kteří byli ochotni poskytnou výpočetní prostředky, bez kterých by tato práce těžko mohla vzniknout.

V neposlední řadě bych také rád poděkoval rodičům za jejich nekonečnou podporu při studiu a kamarádům, kteří mi byli oporou a motivovali mě.

Prohlášení

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

V Plzni dne 11. května 2012 .....

Michal Hrala

Abstract

The main goal of this work is to study methods for a multi-label document classification and to propose a user friendly software solution for Czech News Agency (ČTK). Multi-label classification is a task, where document is classified in to more than one class. Based on the literature, we have chosen three classifiers that are successfully used in the document classification field: Naive Bayes (NB), Support Vectors Machine (SVM) and Maximum Entropy classifier.

We also study the possibility to use Part of Speech (POS) tagging for document word filtration and lemmatization to improve classification accuracy. For the feature selection, five methods are compared: Document Frequency (DF), Information Gain (IG), Mutual Information (MI), Chi-square and GSS methods.

All methods are evaluated on the Czech corpus of ČTK newspapers articles. An optimal classifier setting is proposed based on these results. The proposed software solution uses the MinorThird classification tool package as an implementation of the classification methods. We used the Mate tool for lemmatization and POS tagging.

Keywords: feature selection, lemmatization, Maximum Entropy, Multi-label document classification, Naive Bayes, POS tagging, Support Vector Machine, text classification

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní digitální době jsme každý den zahrnovány stovkami nových dokumentů. Může se jednat například o digitalizované archivní materiály, novinové články, příspěvky na diskusních fórech, elektronickou poštu, obsah webových stránek či elektronické knihy. Roste tak potřeba tyto dokumenty zpracovávat a označovat *metadaty*[1], například pro usnadnění jejich vyhledávání. Jedním z procesů, který lze řešit programově je třídění dokumentů do kategorií, ať už podle jejich významu, autora, žánru, či jiných kritérií.

Těmito problémy se zabývá několik oblastí umělé inteligence. Jedná se např. o zpracování přirozeného jazyka (NLP – Natural Language Procesing) využívající obvykle metod strojového učení (Machine Learning) a o metody dolování dat (Data Mining). Všechny tyto oblasti pracují v součinnosti za jediným účelem a tím je nalezení určitých vzorů či podobností v datech, na základě kterých je poté text tříděn do kategorií.

Tato práce se zabývá klasifikací novinových článků podle obsahu do jedné hlavní kategorie (*burzy*, *politika*, ...) a několika vedlejších kategorií, které blíže článek upřesňují. Mezi vedlejší kategorie mohou patřit například: *burzy akciové*, *burzy komoditní*, *domácí politika*, *zahraniční politika, atd.* Práce je vytvořena na základě potřeb České tiskové kanceláře (dále jen ČTK) a předpokládá se její další rozšiřování na základě požadavků z testovacího provozu.

Čtenář by po přečtení této práce měl porozumět hlavnímu problému klasifikace dokumentů, metodám výběru příznaků pro klasifikaci a být schopen využít tyto informace pro nasazení v jiné oblasti (např. detekce spamu v emailové poště, detekce sentimentu, či blokování pornografie na webu). Dále by měl být schopen konfigurovat vytvořenou aplikaci, trénovat klasifikátor a provádět tak vlastní experimenty.

[1] Strukturované informace o datech.

1.1 Stručný přehled kapitol

V úvodní kapitole je čtenář seznámen s důvody, proč byla tato práce řešena a dále dostane informace o jejím využití. Následující kapitola objasňuje problém klasifikace dokumentů a seznamuje čtenáře s metodami výběru příznaků a klasifikátorů použitých v této práci. Třetí kapitola popisuje klasifikační nástroje a srovnává je podle implementovaných klasifikátorů. Dále se seznámíme s databázemi ČTK, struktuře zpracovávaných souborů a metrik pro vyhodnocení úspěšnosti klasifikace. Další kapitola popisuje řešení úlohy a grafické prostředí vytvořené aplikace. Čtenář zde také dostane informace o struktuře vstupních a výstupních souborů pro jednotlivé fáze klasifikace. Předposlední kapitola zobrazuje výsledky experimentů v přehledných tabulkách a grafech, srovnává obdržené výsledky s teorií a navrhuje nejlepší řešení pro praktické využití. Kapitola závěr shrnuje dosažené výsledky a navrhuje případná další rozšíření.

V přílohách je obsažena struktura přiloženého DVD, UML diagram vytvořené aplikace, konfigurace nástroje s příklady spuštění, struktura textových databází ČTK a přehled kategorií, do kterých budeme články zařazovat.

Kapitola 2

Klasifikační úloha

Tato kapitola si klade za cíl objasnit problémy klasifikace textů. Čtenář se dozví, jaké metody předzpracování dat se používají pro tuto úlohu a jak optimalizovat příznakový vektor. Dále je stručně vysvětlen princip klasifikátorů, které se běžně používají k řešení této úlohy.

2.1 Popis problému

Hlavním problémem při *klasifikaci textů*[1] je výběr vhodných příznaků tak, aby dokument co nejlépe vystihovaly. Samozřejmě můžeme jako příznakový vektor použít všechna slova, která dokument obsahuje. Obecně ale může mít klasifikovaný dokument různou velikost a příznakový vektor by tak byl různě velký. Rozsáhlé dokumenty by bylo časově náročné zpracovat. Trénování klasifikátoru by se tak mohlo prodloužit řádově o hodiny s nejistým výsledkem. Z tohoto důvodu se provádí předzpracování dokumentu, a to nejen kvůli redukci příznakového vektoru, ale také kvůli lepšímu popisu dokumentu, kdy jsou filtrovány nepodstatné informace (slova).

2.1.1 Typy klasifikačních problémů

Samotnou klasifikaci dokumentů můžeme rozdělit na tři podmnožiny [15] podle toho, do kolika kategorií dokument zařazujeme.

Binární klasifikace

Pokud dokument patří přesně do dvou kategorií, provádíme binární klasifikaci. Může se například jednat o klasifikaci typu *je spam/není spam*.

Multi-class klasifikace

Pokud předdefinovaných kategorií, do kterých článek může patřit, je více a každý článek patří přesně do jedné z těchto kategorií, jedná se o tzv. *Multi-class* klasifikaci.

[1] Též kategorizace/klasifikace textů, dokumentů, článků, ...

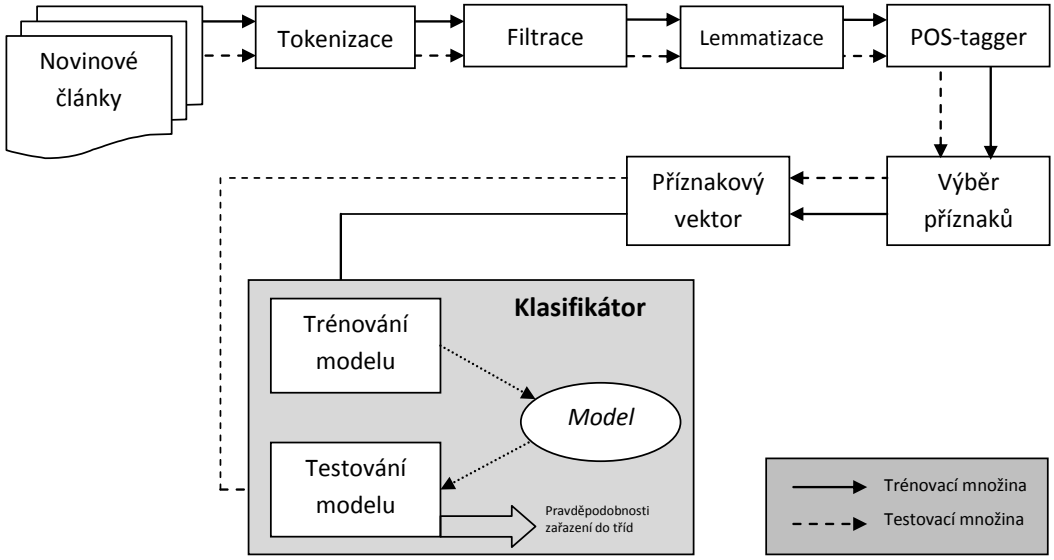
Multi-label klasifikace

Ve většině případů ale dokument nemá stanovenu jednu kategorii do které patří. Obzvlášť u novinových článků je pravděpodobné zařazení do více předddefinovaných kategorií. Tato kategorizace se nazývá *Multi-label* klasifikace a je hlavním cílem této práce.

2.2 Předzpracování dokumentu

Předzpracování dokumentu můžeme rozdělit do několika kroků, jak je vidět na obrázku 2.1 s diagramem funkce klasifikačního systému. Načtený dokument je třeba tokenizovat, tedy rozdělit na jednotlivá slova, termy. V této fázi je prováděna pomocí regulárního výrazu i filtrace případných HTML tagů či vyjadřovacích prostředků jiného jazyka (XML značek) vyskytujících se v dokumentech. Víme-li, že v dokumentech existují slova, která nemají na klasifikaci vliv, je možné tato slova explicitně odstranit aby neovlivňovala klasifikaci.

Po tomto kroku můžeme provést morfologickou analýzu, v našem případě se bude jednat o určení slovních druhů, tzv. *POS-tagging* (viz kapitola 2.2.1) a *lemmatizaci* (viz kapitola 2.2.2).



Obrázek 2.1: Diagram funkce klasifikačního systému

Provedením *lemmatizace* se zbavíme stejných slov v různé osobě, v různém čase a pádě, jinými slovy převedeme slova na základní tvar, což bude vhodné při tvorbě frekvenčního slovníku. Filtraci slov nám výrazně usnadní *POS-tagging*. Umožní

nám vyfiltrovat slova, která mají stejné pravděpodobností rozdělení ve všech dokumentech a ve všech třídách, do kterých klasifikujeme. Jedná se zejména o interpunkční znaky, spojky a předložky. Provedením *POS-taggingu* nám odpadá programová filtrace podle předem určeného vektoru (seznamu) slov tzv. *STOP listu*.

2.2.1 POS-tagging

Part of Speech (POS) tagování, je metoda, která každému tokenu (dále jen slovu) přiřadí slovní druh. Tímto si zjednodušíme filtraci slov pro klasifikaci, protože například předložky a spojky jsou pro klasifikaci většinou nepodstatné.

Přiřazení slovního druhu bude prováděno nástrojem *Mate tool* [4], který je distribuován pod licencí *GNU GPL* a je tudíž možné ho využít zdarma i pro komereční účely. Tento nástroj byl použit jednak z licenčních důvodů, byl mi doporučen a jednak protože dle literatury dosahoval dobrých výsledků.

Označkování slov je provedeno zkratkami z tabulky 2.1 představující jednotlivé slovní druhy.

POS tag	**Anglický význam**	**Český význam**
N	noun	podstatné jméno
A	adjective	přídavné jméno
P	pronoun	zájmeno
C	numeral	číslovka
V	verb	sloveso
D	adverb	příslovce
R	preposition	předložka
J	conjunction	spojka
I	interjection	citoslovce
T	particle	částice
Z	punctuation, numeral figures, root of the tree	čárky, tečky
X	(unknown, unidentified)	neznámé

Tabulka 2.1: Význam POS tagů

2.2.2 Lemmatizace

Lemmatizace označuje proces převodu slov v dokumentu na jejich základní tvar, taktéž slovníkový tvar. Podobným procesem je *Stemming*, který určuje u slov jejich kořen.

Lemmatizace bude provedena taktéž nástrojem *Mate tool* [4] ze stejného důvodu jako u POS-taggingu.

Po provedení lemmatizace například slova *pracující*, *pracovní* budou nahrazena slovem *práce*. Zároveň se odstraní negativní formy slov, takže například slovo *neplatí* bude nahrazeno slovem *platit*.

Výsledkem jsou všechna slova převedena na stejný tvar pro všechny dokumenty. Celková množina unikátních slov napříč třídami se zmenší, výběr slov pro klasifikaci tak bude efektivnější a úspěšnost klasifikace by měla vzrůst za předpokladu, že různé tvary slov se nevyskytují v různých třídách.

2.3 Volba příznaků

Jak jsem již naznačil v předchozí kapitole, hlavním problémem klasifikace dokumentů je vysoká dimenze příznakového vektoru [22]. Z tohoto důvodu se používají metody, které tento příznakový vektor redukují jen na slova, která jsou relevantní pro úlohu klasifikace a nějakým způsobem ji ovlivňují.

Samotné získání příznakového vektoru lze provést několika způsoby. Nejčastěji je používaná metoda, kdy každému slovu v dokumentu je přiřazena frekvence jeho výskytu. Podle [24] je ale mnohem lepší použít funkci inverzní dokumentové frekvence, která je počítána podle vztahu 2.1 a následně normována kosinovou transformací podle vztahu 2.2,

$$tfidf(t_k, d_j) = \varphi(t_k, d_j).\log\frac{|T_r|}{\varphi T_r(t_k)} \tag{2.1}$$

$$t_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{i=1}^{|t|} tfidf(t_i, d_j)^2}} \tag{2.2}$$

kde t_k je k-té slovo (dále jen term) dokumentu d_j, $\varphi(t_k, d_j)$ je frekvence výskytu termu t_k v dokumentu, $|T_r|$ je velikost trénovací množiny, tedy počet dokumentů a $\varphi T_r(t_k)$ je dokumentová frekvence termu t_k, tedy počet dokumentů v trénovací množině ve kterých se daný term vyskytuje.

Ale protože i po výpočtu inverzní dokumentové frekvence je vektor termů příliš dlouhý a čas klasifikace by rostl, provádí se jeho redukce metodami výběru příznaků, které vyberou jen termy, které dokáží dokumenty v jednotlivých kategoriích odlišit.

Podle [29] jsou nejpoužívanější následující metody pro výběr příznaků, proto jsou v této práci testovány.

2.3.1 Dokumentová frekvence (DF)

Dokumentová frekvence udává, v kolika dokumentech se daný term vyskytuje [29, 12, 24]. Je nutné pro každý term projít všechny dokumenty a spočítat frekvenci jeho výskytů. Následně je vybráno jako příznakový vektor k termů s největší hodnotou frekvence výskytů.

Vycházíme z předpokladu, že slova s malou četností výskytů nejsou pro správné určení třídy relevantní. Jedná se o nejjednoduší formu redukce příznakového vektoru. Nevýhodou této metody je paradoxně ignorování méně častých slov, které mohou dokumenty odlišovat.

2.3.2 Information Gain (IG)

Před samotným vysvětlením této metody je nutno definovat termín *entropie*. Entropii můžeme definovat jako míru neurčitosti. Také můžeme použít definici z informatiky, která říká, že entropie je minimální počet bitů potřebných k zakódování informace při optimálním kódování. Obecný vzorec 2.3 pro výpočet entropie je

$$H(s) = \sum -p_i \cdot log_2\ p_i \tag{2.3}$$

kde p_i je pravděpodobnost i-tého mikrostavu. Metoda měří snížení entropie způsobenou přítomností či nepřítomností slova [18, 29] a je možno ji vypočítat dle vztahu 2.4

$$\begin{aligned} IG(t_k) = &- \sum_{j=1}^{j=m} P(c_j)\ log\ P(c_j) + \\ &P(t_k) \sum_{j=1}^{j=m} P(c_j|t_k)\ log\ P(c_j|t_k) + \\ &P(\bar{w}) \sum_{j=1}^{j=m} P(c_j|\bar{t_k})\ log\ P(c_j|\bar{t_k}) \end{aligned} \tag{2.4}$$

kde c značí třídu, $m = |C|$ počet tříd, t_k term a $P(\bar{t_k}) = 1 - P(t_k)$ inverzní pravděpodobnost výskytu slova.

Pokud je hodnota Information Gain vysoká, je příznak považován za důležitý a dané slovo se vztahuje k zařazované kategorii. V ostatních případech je příznak ignorován.

2.3.3 χ^2 test

Jedná se o rozšířený statistický test, který měří závislost třídy c_j na výskytu termu t_k za předpokladu, že výskyt termu je nezávislý na třídě, do které klasifikujeme [27].

Protože se jedná o statistický test, jeho chování je nevyzpytatelné pro slova s málo častým výskytem v dokumentech [12]. Vypočítá se podle vzorce 2.5 uvedeného například v [13, 27],

$$\chi^2(t_k, c_j) = \frac{|Tr|(P(c_j, t_k)P(\bar{c_j}, \bar{t_k}) - P(\bar{c_j}, t_k)P(c_j, \bar{t_k}))}{\sqrt{P(t_k)P(\bar{t_k})P(c_j)P(\bar{c_j})}} \tag{2.5}$$

kde $P(c_j)$ je pravděpodobnost výskytu třídy c_j a $P(c_j, t_k)$ je pravděpodobnost, že se vyskytne term t_k pokud máme třídu c_j. $|Tr|$ označuje velikost trénovací množiny, tedy celkový počet dokumentů.

2.3.4 Mutual Information (MI)

Další rozšířenou metodou pro výběr příznaků je metoda *vzájemné informace* (Mutual information) [18, 7]. Tato metoda také zahrnuje pravděpodobnosti vzhledem ke kategoriím. Mutual Information obecně určuje vzájemnou závislost mezi náhodnými proměnými. V našem případě pro kategorizaci textů určuje vzájemnou závislost třídy a slova.

Pro výpočet $MI(t_k, c_j)$ tedy že slovo t_k odpovídá třídě c_j můžeme použít vztah 2.6 uvedený například v [7] nebo v [13],

$$MI(t_k, c_j) = log\frac{P(t_k, c_j)}{P(t_k)P(c_j)} \tag{2.6}$$

kde $P(t_k, c_j)$ je sdružená distribuční funkce. Stejně jako u Information Gain jsou pro nás zajímavé vyšší hodnoty $MI(t_k, c_j)$.

2.3.5 GSS koeficient

Tato metoda výběru příznaků je pojmenována podle počátečních jmen tvůrců, kterými jsou pánové *Gallavotti, Sebastiani a Simi*. Podrobně je metoda popsána v [13]. Vypočtena je podle vzorce 2.7 a jedná se v podstatě o zjednodušenou formu metody χ^2, ze které je odstraněn jmenovatel. Odstraněné hodnoty $\sqrt{P(c_j)P(\bar{c_j})}$ a $\sqrt{P(t_k)P(\bar{t_k})}$ ze jmenovatele pouze zdůrazňovaly vzácně se vyskytující termy nebo málo časté třídy.

$$GSS(t_k, c_j) = P(t_k, c_j)P(\bar{t_k}, \bar{c_j}) - P(t_k, \bar{c_j})P(\bar{t_k}, c_j) \tag{2.7}$$

2.4 Klasifikační metody

V této kapitole představím tři klasifikační metody vybrané pro danou úlohu. Na základě přečtené literatury jsem se rozhodl použít *Naivní Bayesův klasifikátor*, klasifikátor *Maximální entropie* a *metodu podpůrných vektorů* (SVM - Support Vector Machine).

Nejdříve ale v následující kapitole uvedu, jaké typy učení se v této úloze používají.

2.4.1 Typy učení

Učení s učitelem *(Supervised learning)*

Učení s učitelem, neboli *Supervised Learning* je metoda, při které se ke trénování klasifikátoru používají data, která mají již označenou třídu, do které patří. V našem případě to znamená, že klasifikátor se natrénuje dokumenty označenými kategorií, do které patří.

Klasifikace probíhá přivedením testovací množiny na natrénovaný model klasifikátoru, který určí pravděpodobnosti zařazení dokumentu do jednotlivých tříd. Tato metoda bude použita pro tuto práci.

Učení bez učitele *(Unsupervised learning)*

Narozdíl od učení s učitelem nemáme u této metody informace o zařazení do tříd, máme jen vstupní data. Cílem tedy je nalézt podobnosti mezi dokumenty a na jejím základě dokumenty klasifikovat. Víme, že nad dokumenty existuje nějaká skrytá struktura, ve statistice nazývaná odhad hustoty, která rozhoduje do jaké třídy, který dokument patří.

Jedna z metod pro určení odhadu hustoty, která nalezne skupiny ve vstupních datech je shlukování.

Shlukování si můžeme vysvětlit na aplikaci komprese obrázků. Jako vstup máme matici RGB hodnot. Program pak stejným nebo hodně podobným pixelům přiřazuje jednu barvu a touto jednou barvou (skupinou, třídou) jsou pak reprezentovány nejčastěji se vyskytující pixely podobných barev (shluky barev).

Výsledkem shlukování jsou jednotlivé třídy, u kterých musí člověk rozhodnout, jak budou pojmenovány. V kategorizaci novinových článků se může jednat například o třídy (shluky) *sport, móda, politika, ...* V úloze kategorizace textů se však metoda učení bez učitele příliš nepoužívá, protože klade větší nároky na uživatele (musí pojmenovat obdržené shluky) a dosahuje horších výsledků.

Kombinovaná metoda *(Semi-Supervised learning)*

Metoda kombinuje předchozí dvě metody učení. K trénování klasifikátoru se používají jak data anotovaná učitelem, tak data bez explicitně přiřazené kategorie. Této metody se využívá, pokud máme malé množství anotovaných dat a velké množství dat neanotovaných.

Klasifikační model je nejdříve natrénován na množině anotovaných dat a na tomto modelu jsou poté klasifikovaná neanotovaná data. Z takto klasifikovaných dokumentů se vyberou ty, které obsahují největší pravděpodobnost zařazení do třídy a přidají se ke trénovací množině a celý model je přetrénován, dokud není množina nepřiřazených dokumentů minimální.

Nevýhodou je časová náročnost trénování a ne vždy se úspěšnost klasifikace zlepšuje.

2.4.2 Naivní Bayesův klasifikátor

Pro úlohu automatické klasifikace dokumentů je nejčastěji v přečtené literatuře používán tento klasifikátor a to kvůli svým vlastnostem. Mezi tyto vlastnosti uvedené v [5] patří například vlastnost inkrementálního učení, je tedy možné klasifikátor dotrénovat větší množinou, relativní jednoduchost, malá velikost a rychlost klasifikace. Předpokladem pro tento klasifikátor je nezávislost příznaků.

Princip

Mějme množinu dokumentů $D = \{d_1, d_2, ..., d_m\}$. Každý dokument d_i je reprezentován příznakovým vektorem, kde každá položka tohoto vektoru obsahuje příznak

pro konkrétní term t_j ze slovníku $V = \{t_1, t_2, ..., t_n\}$ obsahující n různých termů [22].

Před uvedením samotných vztahů pro tento klasifikátor je nutné zmínit, že každý dokument d_i přísluší alespoň jedné třídě c_j z množiny všech tříd $C = \{c_1, c_2, ..., c_c\}$. Zařazení neznámého dokumentu d do třídy pak probíhá výpočtem podmíněných pravděpodobností pro každou třídu a výběrem třídy s maximální aposteriorní pravděpodobností.

Výpočet podmíněné aposteriorní pravděpodobnosti $P(c_j|d)$ (pravděpodobnost zařazení dokumentu d do třídy c_j) je proveden použitím Bayessova pravidla 2.8.

$$P(c_j|d) = \frac{P(c_j)P(d|c_j)}{P(d)} \tag{2.8}$$

Apriorní pravděpodobnost $P(c_j)$ je vypočtena z množiny trénovacích dat jako relativní četnost výskytů dokumentů patřících do třídy c_j (vztah 2.9). $P(d)$ pak určuje pravděpodobnost výskyt dokumentu d.

$$P(c_j) = \frac{Pocet\ termu\ v\ c_j}{Celkovy\ pocet\ termu\ v\ trenovaci\ mnozine} \tag{2.9}$$

Podmíněná pravděpodobnost $P(d|c_j)$ dokumentu d vzhledem k třídě c_j je vypočtena z pravděpodobností výskytu jednotlivých termů přes všechny termy nalezené v dokumentu d podle vztahu 2.10.

$$P(d|c_j) = \frac{|d|!}{\prod_{t \in d} f(t,d)!} \prod_{t \in d} P(t|c_j)^{f(t,d)} \tag{2.10}$$

Hodnota *f(t,d)* v předchozím vzorci určuje počet výskytů termu t v dokumentu d a $|d|$ značí součet frekvencí všech výskytů termů v dokumentu neboli celkovou délku dokumentu. $\frac{|d|!}{\prod_{t \in d} f(t,d)!}$ pak značí všechny možné kombinace pořadí termů.

Pravděpodobnosti jednotlivých termů vzhledem ke všem termům v dané třídě jsou vypočteny podle vztahu 2.11 jako podíl frekvence termu v dané třídě (*f(t,c)*) a součtu frekvencí všech termů v dané třídě.

$$P(t|c_j) = \frac{f(t, c_j)}{\sum_{t' \in V} f(t', c_j)} \tag{2.11}$$

Zařazení dokumentu do třídy, jak jsem již uvedl, je provedeno výběrem třídy s maximální aposteriorní pravděpodobností $P(c_j|d)$ podle vztahu 2.12 uvedeném v [19].

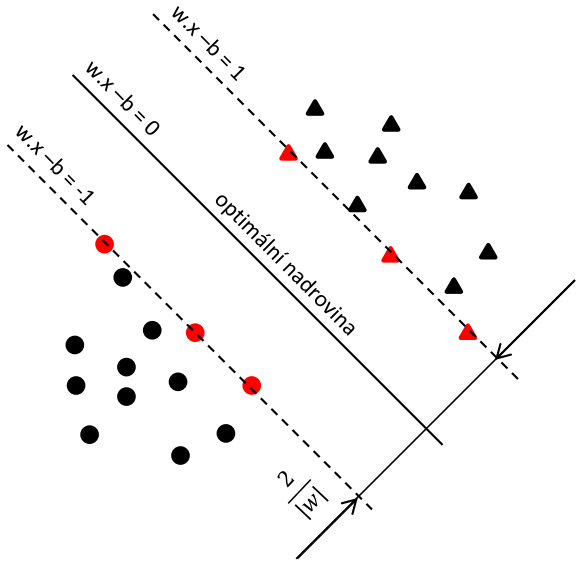
$$c_{max} = \underset{c_j \in C}{arg\,max}\ P(c_j|d) = \underset{c_j \in C}{arg\,max}\ P(c_j) \prod_{k=1}^{n} P(t_k|c_j) \tag{2.12}$$

kde n je počet slov v dokumentu.

2.4.3 Support Vector Machine

Support Vector Machine (dále jen SVM) neboli metoda podpůrných vektorů je druhou metodou strojového učení, kterou budu v této práci využívat, a proto ji zde stručně představím. Informace obsažené v této kapitole byly čerpány převážně z [19] a doplněny z [5], odkud byl také překreslen a doplněn obrázek 2.2.

Omezme se pro jednoduchost a pro lepší vysvětlení na binární klasifikátor. SVM pak funguje na principu hledání nadroviny v prostoru příznaků oddělující od sebe data tak, aby minimální vzdálenost bodů od nadroviny pro obě rozdělené množiny byla maximální. Grafické znázornění metody je na obrázku 2.2.



Obrázek 2.2: Nejlepší možná oddělovací nadrovina a podpůrné vektory

Červené body na obrázku označují podpůrné vektory (*support vectors*), které jsou důležité pro určení optimální nadroviny

$$w \cdot x - b = 0.$$

Klasifikace je pak provedena podle nadrovin pro jednotlivé třídy. Všechny body splňující podmínku

$$w \cdot x - b \geq 1$$

jsou zařazeny do třídy *"trojúhelníčků"* (označme jako y=+1) a všechny body splňující

$$w \cdot x - b \leq -1$$

jsou naopak zařazeny do třídy *"koleček"* (označme jako y=-1). Body, které se vyskytovaly mezi, jsou zařazeny k nejbližší třídě. Za výše uvedených podmínek je hledání

nadroviny řešení optimalizačního problému 2.13.

$$(w, b) = \underset{w,b}{argmax} \frac{1}{2}||w||^2 \tag{2.13}$$

Úloha se ale ve většině případů neřeší přímo, ale využívá se tzv. duálního problému.

Duální případ

Mnohem výpočetně jednodušší je hledat parametry α_i, která jsou řešením rovnice 2.14. K řešení se používá metody *kvadratického programování*.

$$\vec{\alpha} = \underset{\vec{\alpha}}{argmax}(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i, x_j)) \tag{2.14}$$

za podmínek

$$\alpha_i \geq 0, \;\; i = 1, 2...l \quad a \quad \sum_{i=1}^{l} a_i y_i = 0$$

Po vyřešení rovnice přejdeme zpět k přímému řešení a pomocí zjištěných parametrů α vypočítáme hodnotu w podle vzorce 2.15.

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{2.15}$$

Nelineární klasifikace

Obrázek 2.2 ale ilustruje pouze jednoduchý případ, kdy jednotlivé třídy jsou od sebe lineárně separovatelné. Prakticky tomu ale tak není, proto je neoddělitelnou součástí této metody jádrová funkce (tzv. *kernel function* nebo také *kernel transformation*), která převede lineárně neseparovatelnou úlohu na lineárně separovatelnou pomocí projekcí do vyšší dimenze než jsou vstupní data.

Protože pro určení nadroviny a zařazení do třídy pro lineárně separovatelnou úlohu využíváme pouze skalárních součinů, lze využít triku (tzv. *kernel trick*) a pro určení nadroviny ve vícedimenzionálním prostoru použít skalární součiny hodnot jádrové funkce, zapsané jako

$$K(\vec{x_i}, \vec{x_j}) = \vec{x_i}^T \vec{x_j}$$

2.4.4 Maximální entropie

Posledním klasifikátorem je klasifikátor maximální entropie. V oblasti zpracování textu se jedná o rozšířenou metodu použitelnou např. pro zjištění významu slov, strojový překlad nebo označkování textu.

Veškeré informace o tomto klasifikátoru byly čerpány z [8], který velmi podrobně popisuje tento klasifikátor i s názornými obrázky pro pochopení funkce.

Existuje několik definic maximální entropie. Jedna z definic říká, že algoritmus maximální entropie může být využit k nalezení jakéhokoliv pravděpodobnostního rozdělení [9]. Jiná definice říká, že pokud nevíme o statistickém rozdělení vůbec nic, nebo máme jen málo informací, potom nejpravděpodobnější tvar rozdělení je podle informací, které o rozdělení máme a má nejvyšší možnou entropii. To odpovídá principu Occamovy břitvy, která se snaží najít co nejjednoduší popis na základě toho, co známe.

Entropie se spočítá podle vzorce 2.16

$$S = -\sum_{i=1}^{N}(p_i log p_i) \tag{2.16}$$

Princip

Trénovací data se v tomto modelu využívají k omezení rozložení, které musí být splněno při odhadu modelu. Pravděpodobnost p_i získáme jako funkci zařazení daného slova do třídy, která náleží datům v trénovací množině.

Předpokládejme, že máme n-rozměrný vektor příznaků, který odpovídá funkci $f_i(d, c)$ modelující rozložení příznaků pro daný dokument d a třídu c. Naším cílem je nalezení modelu, který bude tomuto rozložení příznaků odpovídat. Nechť D je množina všech trénovacích dat. Námi hledaná podmíněná pravděpodobnost $P(c|d)$ zařazení dokumentu d do třídy c musí splňovat vlastnost 2.17.

$$\frac{1}{|D|}\sum_{d\in D} f_i(d, c(d)) = \sum_{d} P(d)\sum_{c} P(c|d) f_i(d, c) \tag{2.17}$$

P(d) označuje pravděpodobnostní rozdělení dokumentů, které neznáme, proto nás jeho modelování nezajímá. Jako aproximaci této hodnoty použijeme množinu trénovacích dokumentů 2.18.

$$\frac{1}{|D|}\sum_{d\in D} f_i(d, c(d)) = \frac{1}{|D|}\sum_{d\in D}\sum_{c} P(c|d) f_i(d, c) \tag{2.18}$$

Dalším krokem je vypočtení příznakových funkcí z trénovacích dat, které budou použity pro klasifikaci. Potom pro každý příznak vypočítat nad trénovacími daty očekávanou hodnotu, která bude použita jako omezení trénovaného modelu [21].

Parametrická forma

Pokud máme vypočtena všechna omezení, máme zaručeno, že jsme nalezli jednoznačný model, který má maximální entropii. Dle [8] může být ukázáno, že rozdělení má vždy exponenciální charakter (vzorec 2.19)

$$P(c|d) = \frac{1}{Z(d)} exp(\sum_i \lambda_i f_i(d,c)) \tag{2.19}$$

kde $f_i(d,c)$ je model příznaku, λ_i odhadovaný parametr a $Z(d)$ je normalizační faktor pro zajištění správné číselné hodnoty pravděpodobnosti vypočtený podle 2.20.

$$Z(d) = \sum_c exp(\sum_i \lambda_i f_i(d,c)) \tag{2.20}$$

Po vypočtení všech omezení je řešení maximální entropie stejné jako řešení duálního věrohodnostního problému (*Maximum Likehood Problem*) pro modely se stejnou exponenciální funkcí. Obě řešení poskytují konvexní funkci s jedním globálním maximem a žádným lokálním maximem.

K řešení této duální úlohy se používá horolezecký algoritmus (*HillClimbing algorithm*), který z počátečního odhadu nějakého zvoleného exponenciálního rozdělení konverguje k řešení věrohodnostního problému, které je zároveň globálním řešením maximální entropie [21].

Kapitola 3

Analýza

3.1 Nástroje pro klasifikaci textů

Pro řešení této úlohy jsem se na základě přečtených článků, srovnání vlastností a výsledků jednotlivých klasifikátorů rozhodl vyzkoušet celkem tři klasifikační metody.

Základním požadavkem tedy bylo, aby vybraný nástroj pokud možno obsahoval všechny požadované klasifikační algoritmy, byl snadno konfigurovatelný, byly k němu k dispozici zdrojové kódy a abych jejich modifikací neporušil licenční ujednání. Zároveň jsem hledal nástroj, který by bylo možné využít pro komerční využití, nicméně uvádím zde i placené nástroje, zejména kvůli poskytované technické podpoře při výskytu problému s klasifikací.

Nalézt nástroj, který by obsahoval všechny klasifikátory, které jsem chtěl vyzkoušet, se ukázalo jako docela těžký úkol. Většina klasifikačních nástrojů obsahovala pouze dva klasifikační algoritmy z požadovaných tří. Nakonec jsem nalezl nástroj *Minorthird*, který jsem v této práci použil. V několika následujících odstavcích jsou popsány všechny nástroje, o kterých jsem uvažoval. Popis obsahuje přehled licencí, klasifikační algoritmy, které obsahují a jiné důležité vlastnosti, které stojí za zmínku. Zároveň je tato kapitola zakončena tabulkou 3.1 přehledně zobrazující klasifikátory, které obsahují.

3.1.1 SVMlight

SVMlight [16] implementuje metodu *Vapnik's Support Vector Machine*. Tento nástroj jsem uvažoval pro využití v kombinaci s jiným nástrojem, protože většina aplikací tuto metodu neobsahuje kvůli implementační složitosti.

- **Licence:** - Academic Free
- **Cena:** Pro vědecké použití zdarma, pro komeční nutno svolení autora
- **Programovací jazyk:** Java
- **Dokumentace:** Ne

- **Tutoriál:** Ano

3.1.2 Mallet

Aplikace Mallet je dalším balíkem pro klasifikaci textů jiné úlohy strojového učení a zpracování přirozeného jazyka [20]. Nástroj taktéž obsahuje spousty příkladů použití a konfigurace. Nástroj bohužel neobsahuje klasifikátor SVM, proto jsem ho při výběru neuvažoval.

- **Licence:** CPL (Common Public License) licence, OpenSource, možno využít i pro komerční využití
- **Cena:** Zdarma
- **Programovací jazyk:** Java
- **Dokumentace:** API javadoc
- **Tutoriál:** Ano

3.1.3 RTextTools

Velmi podrobně dokumentovaný open source projekt [25] vytvořený v rámci několika univerzit: University of California, University of Washington, Sciences Po Paris, Vrije Universiteit Amsterdam. Obsahuje celkem devět klasifikátorů, bohužel neobsahuje ten nejjednoduší a to Naive Bayes, což byl hlavní důvod, proč jsem ho nepoužil.

- **Licence:** GNU GPL
- **Cena:** Zdarma
- **Programovací jazyk:** Java
- **Dokumentace:** Ano, javadoc i hodně podrobná dokumentace
- **Tutoriál:** Ano

3.1.4 LingPipe

Balík nástrojů LingPipe [1] je určen pro zpracování textů využívající metod počítačové lingvistiky. Velmi drahý nástroj pro komerční využití, složitá licence. Poměr cena/výkon neodpovídá, proto byl nástroj zavržen.

- **Licence:** Licence podle využití
- **Cena:** Zdarma pro akademické použití, 9500$ / 1 rok pro komerční použití
- **Programovací jazyk:** Java
- **Dokumentace:** Ano, javadoc, podrobná dokumentace
- **Tutoriál:** Ano

3.1.5 The Dragon Toolkit

Nástroj [30] je bohužel zdarma k dispozici pouze pro akademické použití a jeho autoři komerční využití vůbec nezmiňují. Aplikace umožňuje využití pro klasifikaci textů, clustering a pro sumarizaci textů. Výhodou nástroje je jeho škálovatelnost. Narozdíl od nástrojů jako Weka se předem počítá s velkými objemy dat, takže data nejsou načítána do paměti všechna, ale postupně, což umožňuje pracovat s mnoha dokumenty v omezené paměti.

- **Licence:** - k dispozici online na *stránkách projektu*[1]
- **Cena:** Zdarma pro akademické použití
- **Programovací jazyk:** Java
- **Dokumentace:** Ano, javadoc i hodně podrobná dokumentace
- **Tutoriál:** Ano

3.1.6 Stanford

Nástroj [11] byl vytvořen na univerzitě Stanford skupinou vědeckých pracovníků. Využit je zejména pro rozpoznávání pojmenovaných entit, ale implementované klasifikátory lze využít i pro klasifikaci textů. Tento nástroj obsahuje zejména *Conditional Random Fields* (CRF) klasifikátor, jež by šlo po správné konfiguraci využít pro klasifikaci algoritmem Maximum Entropy. Taktéž neobsahuje všechny použadované klasifikátory, proto nebyl použit, nicméně ho uvádím protože již s tímto nástrojem mám zkušenosti.

- **Licence:** GNU General Public License, možno využít i pro komerční využití
- **Cena:** Zdarma
- **Programovací jazyk:** Java
- **Dokumentace:** API javadoc
- **Tutoriál:** Ano

3.1.7 Weka 3

Nástroj [14] vytvořený na univerzitě Waikato. Obsahuje algoritmy strojového učení pro využití zejména v aplikaci pro dolování dat z textu. Implementované klasifikátory je možno však využít i pro náš účel, bohužel taktéž neobsahuje všechny požadované. Dokumenty také načítá všechny do paměti, takže není vhodný ani z hlediska paměťové optimalizace.

[1] http://dragon.ischool.drexel.edu/

- **Licence:** GNU General Public License.
- **Cena:** Zdarma pro akademické použití
- **Programovací jazyk:** Java
- **Dokumentace:** Ano, javadoc i hodně podrobná dokumentace
- **Tutoriál:** Ano

3.1.8 Minorthird

Soubor nástrojů vytvořených profesorem na univerzitě Carnegie Mellon [6]. Na stránkách aplikace jsou k dispozici testovací data s ukázkami konfigurace. Nástroj obsahuje všechny tři požadované klasifikátory a mnoho dalších, jejichž výčet je uveden v tabulce 3.1.

- **Licence:** BSD licence, OpenSource, možno využít i pro komerční využití
- **Cena:** Zdarma
- **Programovací jazyk:** Java
- **Dokumentace:** API javadoc
- **Tutoriál:** Ano

Nástroj	Licence	Naive Bayes	Maximum Entropy	SVM	Decision Trees	CRF	HMM	Knn	Neuronové sítě	GLM net
SVMlight	—			×						
Mallet	CPL	×	×		×					
RTextTools	GPL-3		×	×	×				×	×
LingPipe	—	×				×	×	×		
The Dragon Toolkit	—	×		×						
Stanford	GNU GPL		×			×				
Weka 3	GNU GPL	×		×	×					
Minorthird	BSD	×	×	×	×			×		

Tabulka 3.1: Tabulka zobrazující implementované klasifikační algoritmy v klasifikačních balících

3.2 Struktura databáze ČTK

InfoBanka ČTK využívá k uložení svých článků relační a fulltextové databáze. Pro své klienty z nich provádí export do formátu *NewsML* [28], který je založen na XML. Výhodou tohoto formátu je zejména snadná transformace do jiných formátů a přímé využití na webových stránkách. Dále je díky volbě uložení v tomto formátu možno uchovávat větší množství *metadat*[2], která usnadňují uživateli další práci s dokumenty.

3.2.1 Rozdělení InfoBanky

Obsah InfoBanky (více viz [10]) lze rozdělit do několika částí:

- Aktuální zpravodajství
- Archivy
- Fotobanka
- Události
- Podniková data
- Faktografie (znalostní databáze)
- Sportovní relační databáze
- Magazínový výběr Plus
- Monitoring médií

Podrobnější přehled jednotlivých databází je pro přehlednost popsán v příloze D.

3.2.2 Struktura zpracovávaných souborů

Každý novinový článek má pevně dannou strukturu, která je definována exportem z textové databáze.

Kromě samotné textové podoby jsou u každého článku uvedena metadata. Jedná se například o datum, čas, místo a pro mě nejdůležitější hlavní (*hlavni_kategorie*) a vedlejší kategorie (*kategorie*). Každý článek má tedy přiřazen hlavní kategorii a seznam všech kategorií, do kterých patří.

Všechny kategorie mají stejnou prioritu a jedná se o tzv. *ploché schéma*. Mezi kategoriemi tedy neexistuje žádná hierarchie a každý článek může spadat prakticky do libovolné, bez ohledu na existenci v jiných kategorií. Na základě tohoto poznatku nám bude stačit pouze jeden natrénovaný model pro každý klasifikátor.

Podrobnější vysvětlení všech metadat, potažmo XML elementů je uvedeno v tabulce 3.2.

[2]Údaje popisující vlastní dokument.

<dokument />	element, který v sobě vnořuje jednotlivé články
<radek id=""/>	element obsahující identifikátor článku a vnořující elementy s metadaty a samotným textem
<titulek />	titulek článku
<datum />	datum vydání článku
<cas />	čas vydání článku
<lokalita />	místo ke kterému se článek vztahuje
<kw />	klíčová slova
<hlavni_kategorie />	hlavní kategorie článku
<kategorie />	zařazení do vedlejších kategorií
<priorita />	priorita článku
<servis />	servisní informace, např. region článku
<zprava />	samotný textový obsah zprávy

Tabulka 3.2: XML elementy zpracovávaných souborů s novinovými články

Ukázka formátu vstupního souboru

Vstupní soubor pro zpracování parserem je sestaven z XML elementů obsažených v tabulce 3.2 a má následující strukturu:

```
<dokument>
  <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 kat</kategorie>
    <priorita>4</priorita>
    <servis>mus</servis>
    <zprava>
      Tornádo v USA zabilo šest lidí <p>;Washington ... str</p>
    </zprava>
  </radek>
  <radek>
    ...
  </radek>
</dokument>
```

3.2.3 Kategorie článků

Veškeré kategorie článků do kterých budeme články klasifikovat jsou uvedeny v příloze E v tabulce E.1.

3.2.4 Textový korpus

Následující statitiska v tabulce 3.3 ukazuje přehled ručně anotovaných (označených) dokumentů, které budou použity k trénování modelu a k ověření úspěšnosti natrénovaného modelu. Z celého korpusu byly vybrány kategorie, které měly více jak 250 dokumentů. Zároveň jich ale bylo z každé kategorie vybráno maximálně 600, aby trénovací data byla vyvážená. Vyhodnocení bylo provedeno *křížovou validací*[3]

Celkový počet dokumentů	11955
Velikost trénovací množiny	80% korpusu vyhodnoceno *křížovou validací*
Hlavních kategorií	25
Vedlejších kategorií	60
Počet slov	2974040
Počet unikátních slov	193399
Počet unikátních lemmat	152462
Počet podstatných jmen	1243111
Počet přídavných jmen	349932
Počet zájmen	154232
Počet číslovek	216986
Počet sloves	366246
Počet příslovcí	140726
Počet předložek	346690
Počet spojek	144648
Počet citoslovcí	8
Počet částic	10983

Tabulka 3.3: Statistika textového korpusu

3.3 Evaluační metriky

Pro vyhodnocení, zda byl dokument správně klasifikován je potřeba znát třídu nebo třídy, do kterých dokument patří. Následně je použita některá z následujících evaluační metod [19] používaných pro danou úlohu k určení úspěšnosti klasifikace.

V našem případě se pro vysvětlení metrik omezíme na binární klasifikátor, tedy kategorizujeme do dvou tříd. Základem pro většinu metrik je určení konfuzní ma-

[3] Ang. Cross Validation, je statistická metoda vyhodnocování úspěšnosti klasifikace.

tice velikosti 2x2 prvků. Matice a její jednotlivé prvky jsou podrobně vysvětleny tabulkou 3.4.

	správná třída		
		Ano	Ne
predikovaná třída	Ano	tp_c	fp_c
	NE	fn_c	tn_c

Tabulka 3.4: Konfuzní matice binárního klasifikátoru pro třídu c

Jednotlivé buňky tabulky si můžeme vysvětlit následujícím způsobem. Ptáme se, zda *predikovaná třída*[4] patří či nepatří (Ano/Ne) do třídy c a očekáváme odpověď Ano/Ne zda klasifikátor rozhodl správně.

Vysvětlení dílčích buněk:

- tp_c *(true positives)* - kategorie dokumentu se shoduje se správnou kategorií
- fp_c *(false positives)* - kategorie se neshoduje se správnou kategorií, klasifikátor zařadil dokument do špatné kategorie
- fn_c *(false negatives)* - klasifikátor nezařadil dokument do kategorie do které skutečně patří
- tn_c *(true negatives)* - klasifikátor správně rozhodl o nezařazení do kategorie

V ideálním případě budou hodnoty tp_c a tn_c rovny hodnotě 1 a zbývající 0.

3.3.1 Přesnost

Přesnost (*Precision*) určuje, kolik dokumentů označených klasifikátorem bylo kategorizováno správně. Vypočtení je podle vzorce 3.1.

$$P = \frac{tp_c}{tp_c + fp_c} \tag{3.1}$$

3.3.2 Úplnost

Úplnost (*Recall*) označuje, kolik dokumentů z celkového množství bylo klasifikováno správně. Určení úplnosti je podle vzorce 3.2.

$$R = \frac{tp_c}{tp_c + fn_c} \tag{3.2}$$

[4] Výsledek klasifikátoru

3.3.3 F-measure

Metrika F-measure patří obecně k nejpoužívanějším metrikám [19] pro zjištění správnosti klasifikace. Tato metrika je počítána jako harmonický průměr *přesnosti* a *úplnosti*. Vzorec pro výpočet F-measure je dán vztahem 3.3.

$$F = \frac{2PR}{P+R} = \frac{2tp_c}{2tp_c + fp_c + fn_c} \tag{3.3}$$

3.3.4 Cohenenova kappa statistika

Další metrikou pro určení úspěšnosti klasifikace je *Cohenenova kappa* statistika [2]. Pokud máme vypočtenou konfuzní matici, je její výpočet dán jednoduchým vzorcem 3.4

$$\kappa = \frac{P_0 - P_c}{n - P_c} \tag{3.4}$$

ve kterém je P_0 vypočteno podle vzorce 3.5, P_c pak podle vzorce 3.6 a n je součet všech prvků konfuzní matice. V následujících vzorcích n_{i+} a n_{+i} má význam *řádkové*[5], resp. *sloupcové*[6] pravděpodobnosti.

$$P_0 = \sum_{i=1}^{c} n_{ii} \tag{3.5}$$

$$P_c = \sum_{i=1}^{c} \frac{n_{i+} * n_{+i}}{n} \tag{3.6}$$

3.3.5 Chybovost - Error Rate

Další metrikou pro určení úspěšnosti klasifikace je *chybovost* (Error Rate). Zatímco u předchozích metod jsme přistuppovali k hodnocení z pozitivního hlediska, tedy jak byl klasifikátor úspěšný (v procentech), u této metody zjištujeme, jak velké chyby jsme se dopustili.

Výpočet můžeme znovu provést z konfuzní matice podle vzorce 3.7 a to vydělením *chybně klasifikovaných* [7] (E) počtem všech klasifikovaných dokumentů (N).

$$ER = \frac{E}{N} \tag{3.7}$$

Touto metodou můžeme vypočítat dílčí ER pro každou kategorii, kdy počítáme pouze z příslušné řádky (sloupce) konfuzní matice.

Tato metrika má souvislost s metrikou *Accuracy* popsanou v kapitole 3.3.7. Pro klasifikaci do jedné kategorie platí vztah $Acc = 1 - ER$.

[5] Součet pravděpodobností na řádce v konfuzní matici.
[6] Součet pravděpodobností ve sloupci v konfuzní matici.
[7] Součet všech čísel (počtu dokumentů) konfuzní matice mimo diagonálu.

3.3.6 Hammingovo metrika

Předchozí metriky jsou vhodné především pro kategorizaci do jedné kategorie. Pokud ale chceme dokumenty zařazovat do více kategorií, je nutné pro vyhodnocení úspěšnosti použít jiné metriky. Jednou z takovýchto metrik je *Hamming Loss* popsaná v [26]. Tato metrika se vypočítá podle vzorce 3.8 a čím vyšší hodnota, tím je klasifikátor horší.

$$HammLoss = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \oplus Z_i|}{|L|} \tag{3.8}$$

Hodnota $|D|$ vyjadřuje počet dokumentů v testovací množině, Y_i je množina kategorií do kterých článek patří, Z_i množina kategorií do kterých dokument zařadil klasifikátor, $|L|$ je velikost množiny do kterých dokument patří a $\oplus$ je symetrický rozdíl dvou množin neboli XOR operace známá z boolovské logiky.

Pokud například článek patří do čtyř kategorií a klasifikátor rozhodne o správnosti zařazení do dvou, je Hammingova ztráta rovna hodně 0.5, pokud vše klasifikuje správně, je hodnota nulová. Jedná se vlastně o zobecnění metriky *Error Rate* pro více kategorií, protože pro klasifikaci do jedné kategorie je definice stejná.

3.3.7 Accurracy - přesnost

Druhou metrikou, kterou využiji pro zhodnocení úspěšnosti klasifikace do více kategorií je metrika *Accuracy*, překládána též jako přesnost. Protože by ale mohlo dojít k záměně s přesností již definovanou výše, ponechávám anglický název. Metrika je popsána v [26] a definována vzorcem 3.9.

$$Acc = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{3.9}$$

Y_i a Z_i mají stejný význam jako v předchozím vzorci, tedy Y_i je množina kategorií do kterých článek patří a Z_i množina do kterých nám dokument zařadil klasifikátor. Čím vyšší hodnota, tím byla klasifikace úspěšnější.

Kapitola 4

Řešení klasifikační úlohy

Řešení bylo naprogramováno v programovacím jazyce Java jdk 1.7.

Kromě úprav nástroje *Minorthird* tak, aby poskytoval výsledky klasifikace v požadovaném formátu, bylo potřeba vytvořit soubor tříd, které provádí čtení a parsování vstupních *XML* souborů. Dále bylo potřeba vytvořit třídy pro transformaci dat, pro *lemmatizaci* a *POS-tagging*, třídy filtrující výstupní soubory z morfologické analýzy[1] a třídy pro vytvoření vstupních souborů pro klasifikátor.

Poslední částí programového řešení bylo vytvoření GUI aplikace, ve kterém je možno klasifikovat jeden článek zkopírováním do okna aplikace, či více článků načtených z XML souboru. Výsledek je opět zobrazen v okně aplikace a vygenerován příslušný výstupní XML soubor s pravděpodobnostním zařazením článku do jednotlivých kategorií.

4.1 Návrh aplikace

Program se zkládá z několika modulů, které jako celek tvoří systém, umožňující převod XML souborů s novinovými články na textové dokumenty využité klasifikačním systémem MinorThird pro trénování a klasifikaci. Každý modul lze samostatně spouštět a to s parametry popsanými v kapitole C.3. UML diagram nejdůležitějších komponent je zobrazen na obrázku B.1 v příloze B.

Prvním modulem je modul *preprocessing*, který obsahuje metody pro předzpracování XML souborů pomocí SAXu, parsuje načtené dokumenty a generuje vstupní soubory pro morfologickou analýzu ve formátu *CONLL* (formát CONLL popsán v kapitole 4.5). Dále tento modul spouští lemmatizaci a POS-tagging z nástroje *MateTool*.

Dalším modulem je modul *featuresExtractor*, který slouží k načtení výstupních souborů z morfologické analýzy. Obsah těchto souborů je filtrován podle POS tagů uvedených v konfiguračním souboru (viz příloha C.2). Dále modul obsahuje metody pro spočítání statistiky výskytu slov (lemmat) v jednotlivých agenturních zprávách a napříč kategoriemi. Z těchto statistik je pak podle vzorců pro jednotlivé metody

[1] V našem případě již zmiňovaná lemmatizace a POS-tagging.

výběru příznaků vybrán vektor slov, který bude použit pro klasifikaci. Hodnoty příznaků jsou poté zapsány do souboru (formát v kapitole 4.6) pro klasifikátor.

Modul *evaluation* pak vyhodnocuje příslušnými metrikami výstup klasifikace. Formát souborů výsledků klasifikace je popsán v kapitole 4.7.

Posledním modulem je modul *util*, který obsahuje pomocné programy například pro výpis statistik souborů s agenturními zprávami, promíchání souborů a filtraci souborů podle identifikátoru. Dále implementuje program pro přepočtení výsledků klasifikace vyjádřených koeficientem důvěry na procentuální úspěšnost pro Naivní Bayesův klasifikátor a klasifikátor Maximum Entropy.

Veškeré moduly jsou pak propojeny programem *Gui.java*, ve kterém lze vše konfigurovat z jednoho místa. Kromě jiného umožňuje načtení souborů s agenturními zprávami, načtení výsledků klasifikace, jejich procházení v přehledném okně a ukládání upravených výsledků. Dále umožňuje natrénovat vybraný klasifikátor a klasifikovat články na tomto natrénovaném modelu.

4.2 Struktura programu

Program můžeme rozdělit do několika balíků obsahující třídy podle jejich funkce. Všechny třídy využívají konfigurační Java *properties* soubor, který se defaultně jmenuje "*properties.properties*". Tento soubor jde pro každou třídu zaměnit za jiný, pomocí volitelného parametru. Význam položek properties souboru je vysvětlen v příloze C.2, příklady spuštění pak v příloze C.3.

Seznam a význam spustitelných tříd je zobrazen v následující seznamové struktuře. Třídy, které neimplementují *main* metodu zde popisovány nebudou, jejich popis lze najít v javadoc dokumentaci na přiloženém DVD.

- **preprocessing:**

  Preprocessing.java - obsahuje *SAX* parser vstupních XML souborů, filtruje slova (termy) podle *properties* souboru a generuje soubor v *Conll*[2] formátu pro morfologickou analýzu.

  Lemmatization.java - třída spouštějící nástroj *Mate tool*, který provede lemmatizaci.

  POSTagging.java - třída spouštějící nástroj *Mate tool* pro POS tagging.

- **featuresExtractor:**

  FeaturesExtracor.java - třída načítá výstupní soubory tříd v balíku *preprocessing* a podle properties souboru a parametrů příkazové řádky generuje vstup pro klasifikátor.

[2]Conference on Computational Natural Language Learning - http://www.clips.ua.ac.be/conll/

- **evaluation:**

  Evaluation.java - třída načítá výstup klasifikátoru, vyhodnocuje úspěšnost pro *Multi-label* klasifikaci metrikami *Hamming Loss* a *Accuracy*. Zároveň vrací optimální parametr k automatickému přiřazení správných predikovaných kategorií.

- **utils:**

  Statistics.java - vrací statistické informace lemmatizovaného a POS tagovaného souboru.

  Shuffle.java - promíchá články ve výstupním souboru z tříd balíku *preprocessing*. Promíchání souboru není nezbytně nutné provádět. Je použito proto, aby třídy v souboru byly rovnoměrně rozložené.

  FiltrationOrderID.java - ponechá z výstupních souborů tříd balíku *preprocessing* pouze články definované ve zvláštním souborů obsahující *id* ponechávaných dokumentů.

  ConvertXMLResults.java - převede soubory s úspěšnostmi klasifikace vyjádřenými koeficientem důvěry na procentuální úspěšnost.

- **gui:**

  Gui.java - vykreslí GUI aplikace, ve kterém lze provádět všechny předchozí zmíněné akce z centrálního místa. GUI aplikace je popsáno v kapitole 4.3.

4.3 GUI aplikace

Uživatelské prostředí je zobrazeno na obrázku 4.1 s podrobnějším vysvětlením níže. Ovládání programu je intuitivní, proto zde nebudou popsány všechny funkce, které poskytuje.

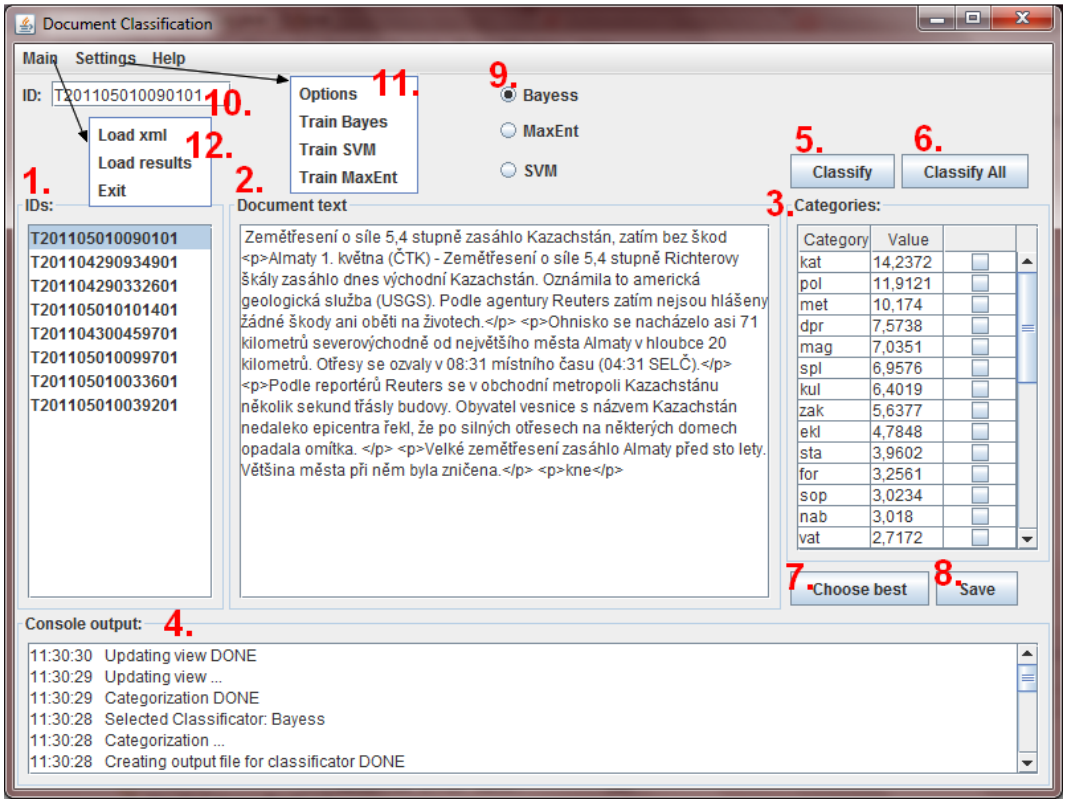
Význam očíslovaných komponent v GUI aplikace je následující:

- 1. Zobrazuje všechna *ID* článků, která byla načtená pomocí *Load XML* z menu aplikace. Toto okno je zobrazeno pouze pokud vstupní XML soubor obsahuje více než jeden článek. Po kliknutí na konkrétní *ID* je zobrazen příslušný článek v komponentě číslo 2.

- 2. Zobrazuje článek po kliknutí na *ID*, umožňuje vložit jakýkoliv vlastní text či zobrazuje článek načtený z XML souboru.

- 3. Zobrazení výsledku klasifikace. Číselná hodnota ve druhém sloupci tabulky udává procentuální úspěšnost pro kategorii zobrazenou v prvním sloupci. Třetí sloupec pak umožňuje uživateli aplikace vlastní výběr kategorií.

- 4. Logování aplikace, zobrazuje prováděné akce s časovým razítkem. Logování je zároveň provedeno i do souboru *log.txt*

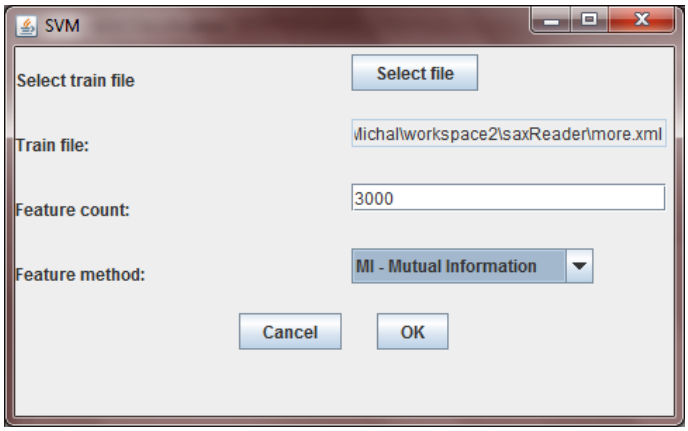


Obrázek 4.1: GUI aplikace

- 5. Klasifikuje právě zobrazený článek v okně č. 2. a mezivýsledky ukládá do souborů definovaných v properties souboru.
- 6. Klasifikuje všechny načtené články.
- 7. Vybere nejpravděpodobnější kategorie na základě Δ[3] parametru uvedeného v konfiguraci.
- 8. Uloží vybrané kategorie do XML souboru ve formátu popsaném v druhé části kapitoly 4.7.
- 9. Výběr klasifikátoru.
- 10. *ID* článku, který je právě zobrazen. Po spuštění obsahuje náhodně vygenerovanou hodnotu a pod touto hodnotou jsou taktéž výsledky uloženy.
- 11. Po výběru *Settings* z menu aplikace je zobrazena nabídka, kde můžeme provést konfiguraci aplikace výběrem položky *Options* a nebo natrénovat požadovaný klasifikátor. Po výběru položky trénování je zobrazeno okno na obrázku 4.2. Trénování započne stiskem tlačítka *OK*, průběžné logy jsou vypisovány do konzolového okna (bod 4).

[3] Číselná hodnota udávající prahovou hodnotu pro určení správných tříd z výstupu klasifikátoru. Pokud je rozdíl po sobě jdoucích tříd menší než Δ, je třída určena jako správná.

- 12. Výběrem *Load xml* z menu aplikace provedeme načtení novinových článků do aplikace. Pokud už máme k dispozici výsledky klasifikace, můžeme tento soubor taktéž nahrát výběrem položky *Load results*. Po kliknutí na identifikátor článku v levém panelu je na pravo zobrazen příslušný výsledek klasifikace, který lze libovolně měnit a změny potvrdit tlačítkem *Save*.



Obrázek 4.2: Okno pro nastavení parametrů trénování klasifikátoru

4.4 Výstup parseru

Protože články obsahují ve svém textu HTML tagy, které nás nezajímají, jsou při parsování automaticky odstraněny pomocí regulárního výrazu. Výstupní soubor parseru má následující strukturu:

```
1       T201105010090101:kat__kat
2       Zemětřesení
3       o
4       síle
5       54
6       stupně
7       zasáhlo
8       Kazachstán
```

Každý řádek souboru se skládá ze dvou sloupců oddělených tabulátorem. První sloupec udává pořadí slova v dokumentu, druhý sloupec slovo. Vyjímku tvoří první řádek každého článku, který místo slova obsahuje identifikátor článku, za dvojtečkou hlavní kategorii a za dvěma podtržítky všechny kategorie, do kterých článek patří.

Jednotlivé články jsou v dokumentu odděleny prázdnou řádkou a pro každý dokument začíná číslování v prvním sloupci od hodnoty jedna.

Soubor je navržen tak, aby byl kompatibilní s formátem *CONLL 2009* podrobněji popsaném v [23]. S tímto souborem pracují algoritmy morfologické analýzy implementované v nástroji *MateTool*.

4.5 Morfologická analýza

Trénování nástroje MateTool

Trénovací soubor pro morfologickou analýzu je ve formátu *CONLL 2009* a má následující strukturu:

1	Zatím	zatím	zatím	D	D
2	však	však	však	J	J
3	k	k−1	k−1	R	R
4	němu	on−1	on−1	P	P
5	nedošlo	dojít	dojít	V	V
6	a	a−1	a−1	J	J

Význam prvního a druhého sloupečku je stejný jako v předchozí kapitole, tedy první sloupeček udává pořadí slova v dokumentu, druhý slovo. Třetí a čtvrtý sloupeček označuje *lemma* a poslední dva sloupečky slovní druh, tedy *POS tag*. Význam POS tagů je uveden v kapitole 2.2.1.

Výstup morfologické analýzy

Výstup morfologické analýzy je taktéž jako vstup ve formátu *CONLL 2009*. Formát je podobný jako trénovací soubor v kapitole 4.5, pouze jsou v něm vynechány nějaké sloupečky podle toho, zda jsme prováděli lemmatizaci nebo POS-tagging.

4.6 Vstup klasifikátoru

Formát vstupního souboru pro klasifikátor je ve tvaru:

```
k T201105150463801 med reklamní=0.4495255053164 reklama=0.407131762840
k T201101310244601 zdr nemocnice=0.5237997805152 akutní=0.401031908799
k T201101040515101 mak ekonomika=0.5120925109851 čínský=0.276210879360
k T201101100857202 pol ksčm=0.40702740711870 konflikt=0.26838959499898
```

První příznak může nabývat dvou hodnot a to *b* pro *binární klasifikaci*[4] a *k* pro klasifikaci do více tříd. V našem případě bude hodnota vždy `k`. Druhé slovo obsahuje identifikátor dokumentu. Třetí slovo označuje kategorii do které článek skutečně patří. Pokud článek patří do více kategorií, bude se celá řádka opakovat, pouze s touto změněnou hodnotou. Za kategoriemi následují samotné příznaky ve tvaru `slovo=váha` oddělené mezerou. Pokud váha slova není uvedena, je automaticky rovna hodnotě jedna.

[4] Klasifikace do dvou tříd.

4.7 Výstup klasifikátoru

Pro tuto úlohu bylo nutné přizpůsobit nástroj *Minorthird* tak, aby vracel pravděpodobnosti zařazení do všech tříd a ne jen jednu třídu. Takto upravený nástroj je k dispozici na přiloženém DVD, stejně jako veškeré výstupy prováděných experimentů.

Nástroj byl upraven tak, aby poskytoval dva možné výstupy.

První výstup

Prvním výstupem je soubor, který má následující strukturu:

```
        Class1   Class2   Class3
Class1:  2        0        0
Class2:  1        1        1
Class3:  0        2        0
...
-----
error Rate:              0.121223
error Rate Class1:       0
...
Kappa:                   0.879908


-----
 [Class1, Class2, Class3]
 ID1  Class1  [Class1=0.6450,  Class2=0.2828,  Class3=0.0153]  Class1
 ID2  Class2  [Class2=0.8234,  Class1=0.7775,  Class3=0.0736]  Class1
 ...
```

V první části dokumentu je zobrazena konfuzní matice, která zobrazuje počet správně/špatně klasifikovaných článků. Následuje vypočtení úspěšnosti klasifikace metrikou *error rate* pro celou testovací množinu, pro jednotlivé kategorie a úspěšnost klasifikace metrikou *Kappa*.

Poslední částí dokumentu jsou podrobnější výsledky a zařazení jednotlivých článků do kategorií. První řádka obsahuje seznam všech kategorií vyskytujících se v testovací množině. Každá další řádka odpovídá jednomu klasifikovanému dokumentu. Řádka začíná identifikátorem dokumentu, následuje predikovaná třída, v hranatých závorkách seřazené třídy podle nejpravděpodobnější a končí třídou, do které dokument skutečně patří. Pravděpodobnost predikovaných tříd je pouze u SVM klasifikátoru, pro ostatní klasifikátory je pravděpodobnost vyjádřena koeficientem důvěry.

Druhý výstup

Druhým výstupem je *XML* soubor s následující strukturou,

```
<document>
  <doc id="953565" realCategory="-">
    <category id="Class1">0.2818557290317149</category>
    <category id="Class2">0.1794240811363971</category>
    <category id="Class3">0.1794240811363971</category>
    ...
  </doc>
</document>
```

ve kterém je každý dokument jednoznačně identifikován podle *id* a obsahuje XML elementy s predikovanými kategoriemi a jejich pravděpodobnostmi seřazené podle nejpravděpodobnější. Stejně jako u předchozího formátu prvního výstupu je pro klasifikátor SVM použita pravděpodobnost a pro zbylé klasifikátory koeficient důvěry. Soubory s tímto koeficientem důvěry lze převést na soubory s vyjádřením pravděpodobnosti pomocí programu `ConvertXMLResults.java` balíku `utils`.

Kapitola 5

Dosažené výsledky

Pokud není uvedeno jinak, všechny klasifikátory byly trénovány na množině obsahující 425 +/- 175 článků v každé kategorii. Toto omezení jsem aplikoval z důvodu, aby byly třídy vyvážené a výsledky nebyly zkreslené. Experimentálně bylo zjištěno, že větší počet článků již nemá na úspěšnost klasifikace vliv. U vícetřídní klasifikace se vyvažovalo pouze podle první kategorie, nicméně i přes toto opatření měly časté kategorie jako například politika více článků.

Na základě přečtených prací jsem se rozhodl vyzkoušet celkem tři klasifikátory, Naivní Bayesův klasifikátor, SVM a Maximum Entropy. Klasifikátory byly vybírány podle nejpoužívanějších a podle toho, jaké úspěšnosti dosahovaly. Pokud bych měl srovnat úspěšnosti, nejlepších výsledků by měl dosahovat klasifikátor SVM a nejhorších výsledků Naivní Bayesův klasifikátor. Samozřejmě nelze na základě literatury přesně srovnat výsledky, protože každý článek používal jiné nastavení klasifikátorů, různé metody výběru příznaků a byly používány pro různé počty tříd a jiné druhy/množství klasifikovaných dokumentů.

Pro výběr vhodných příznaků byly použity běžně používané metody: Dokumentová frekvence, Mutual information, Information Gain, Chi-kvadrát test a metoda GSS popsané v kapitole 2.3.

Pro vyhodnocení výsledků byly v případě klasifikace do jedné kategorie použity metriky *Error rate* a *Kappa* statistika a pro případ klasifikace do více kategorií pak *Hammingova metrika* a *Accuracy* (vše popsané v kapitole 3.3). Tyto metriky vyhodnocení úspěšnosti klasifikace jsou běžně používány pro tento typ úlohy a *Error rate* a *Kappa* statistika byly již implementovány v klasifikačním systému, který byl použit.

Veškeré výsledky experimentů pocházejí z *křížové validace* a to v poměru 1:5. Tedy soubor s příznakovými vektory byl rozdělen na 5 stejných částí, kdy pro trénovaní byly použity 4/5 a pro testování zbývající část. Celkem tedy byla klasifikace pro každý experiment spuštěna 5x pro všechny trénovací kombinace a otestována zbývající částí. Výsledky křížové validace jsou zobrazeny v tabulkách obsažených v podkapitolách níže. Křížová validace je již implementována v klasifikačním systému.

5.1 Určení velikosti příznakového vektoru

V této kapitole jsou zobrazeny výsledky vlivu velikosti vektoru příznaků na úspěšnost klasifikace pro různé metody výběru příznaků. Předem nutno zopakovat, že vektor obsahuje množinu slov společnou pro všechny kategorie, tedy testovací soubor bude obsahovat pouze slova obsažená v tomto vektoru, stejně jako trénovací soubor. Pro testovací soubor je vektor slov načítán ze souboru, který byl vytvořen při vytváření souboru pro trénování klasifikátoru.

Na základě výsledků těchto experimentů byly zvoleny velikosti příznakových vektorů pro každý klasifikátor, které budou použity v následujících experimentech, včetně rozpoznávání do více kategorií.
Na základě doporučené literatury byla stanovena počáteční konfigurace klasifikátoru, která by měla poskytovat nejlepší výsledky. V tomto testu byla použita lemmatizace (podle [19]) a filtrování podle POS tagů. Trénovací / testovací vektory obsahovaly pouze slova, jejichž POS tagy odpovídaly podstatným jménům, přídavným jménům, příslovcům a slovesům (odpovídající POS zkratkám *N, A, D, V*). Ostatní slovní druhy byly vynechány, protože by měly být pro klasifikaci irelevantní, jak se lze dočíst například v [17] nebo ve [3], kde dokonce používají pouze podstatná jména.

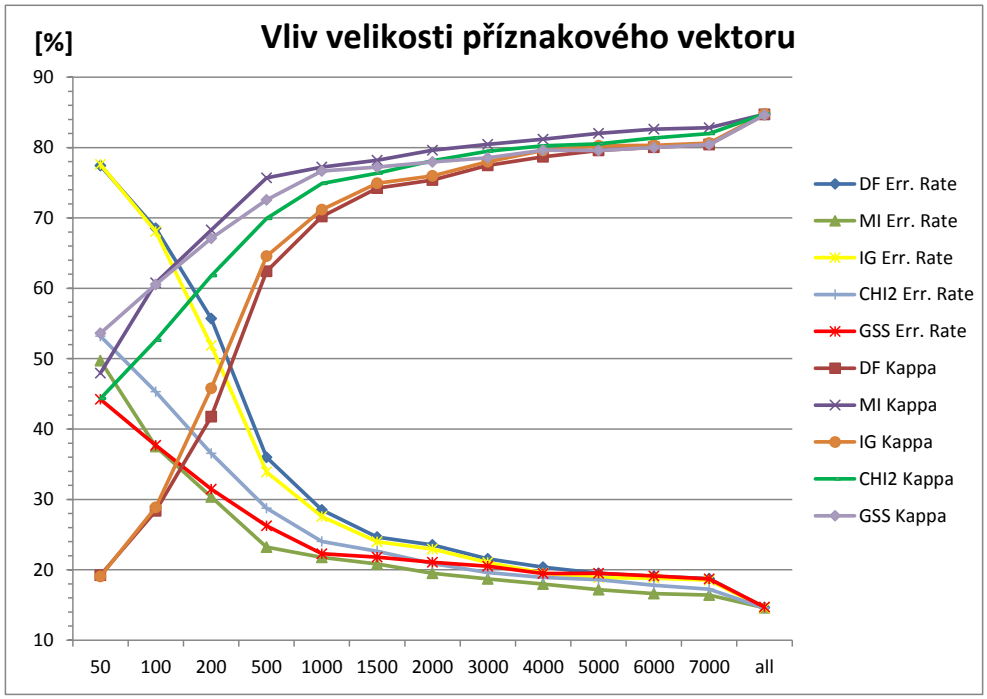
Výsledky experimentu pro Naivní Bayesův klasifikátor jsou zobrazeny v % v tabulce 5.1, kde v prvním sloupci je zobrazena délka příznakového vektoru. Položka *all* v tomto sloupci značí, že délka vektoru nebyla nijak omezena. Pro toto nastavení by měly všechny metody výběru příznaků poskytovat stejné výsledky, jelikož všechny poskytují vektor obsahující všechna slova.

Graf zobrazující závislost velikosti příznakového vektoru na úspěšnost klasifikace pro jednotlivé metody výběru příznaků vyhodnocené metrikou *Error rate* je zobrazen klesajícími křivkami na obrázku 5.1. Metrikou *Kappa* pak stejným grafem, rostoucími křivkami. Pro zbývající dva klasifikátory jsou grafy a tabulky obdobné. Pro klasifikátor SVM jsou dány tabulkou 5.2 a grafem na obrázku 5.2, pro klasifikátor Maximum Entropy pak tabulkou 5.3 a grafem 5.3.

5.1.1 Naivní Bayesův klasifikátor

velikost vektoru	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
50	77,44	19,22	49,71	47,95	77,6	19,05	53,21	44,33	44,22	53,63
100	68,54	28,37	37,51	60,79	68,08	28,86	45,3	52,61	37,69	60,54
200	55,71	41,76	30,35	68,28	51,86	45,78	36,55	61,79	31,46	67,1
500	35,99	62,40	23,24	75,71	33,91	64,56	28,77	69,92	26,25	72,55
1000	28,50	70,21	21,77	77,24	27,58	71,18	24,02	74,89	22,3	76,68
1500	24,65	74,23	20,84	78,21	23,99	74,92	2,63	76,34	21,79	77,22
2000	23,55	75,38	19,5	79,61	22,96	75,99	20,91	78,14	21,07	77,97
3000	21,55	77,47	18,7	80,45	21,05	77,99	19,62	79,49	20,52	78,54
4000	20,37	78,70	17,98	81,2	19,53	79,58	18,9	80,24	19,47	79,64
5000	19,52	79,59	17,18	82,03	18,88	80,25	18,61	80,54	19,52	79,59
6000	19,07	80,06	16,62	82,62	18,82	80,31	17,8	81,38	19,11	80,02
7000	18,73	80,41	16,42	82,83	18,52	80,63	17,23	81,98	18,72	80,43
all	14,63	84,70	14,6	84,73	14,56	84,77	14,57	84,76	14,69	84,63

Tabulka 5.1: Vliv velikosti příznakového vektoru na úspěšnost klasifikace pro Naivní Bayesův klasifikátor

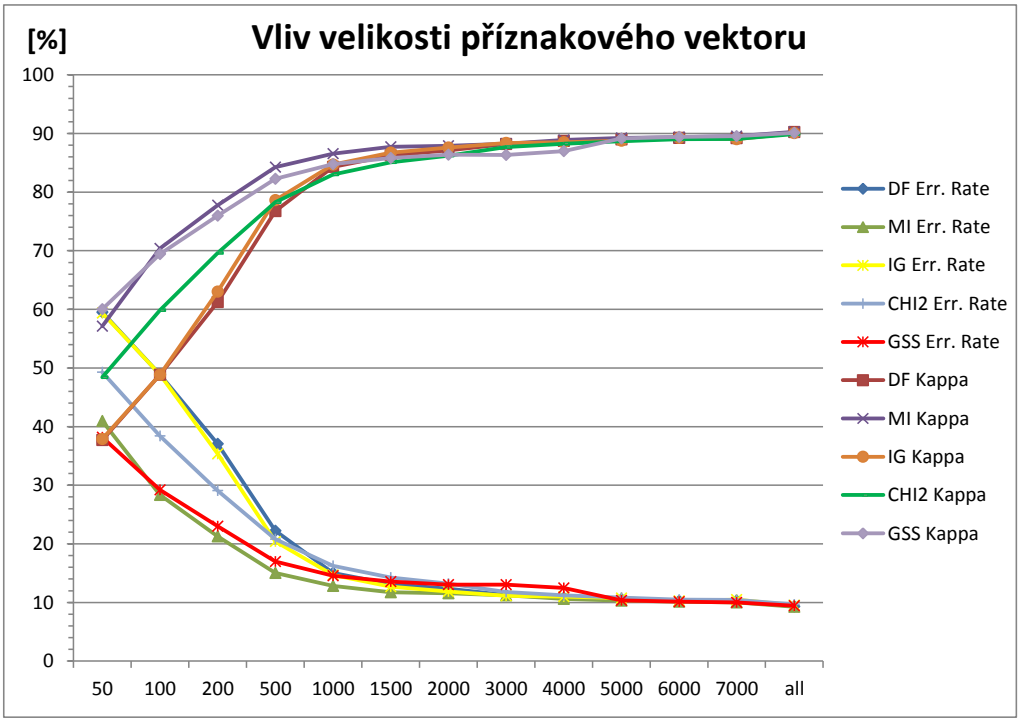


Obrázek 5.1: Graf závislosti velikosti příznakového vektoru na úspěšnost klasifikace pro Naivní Bayesův klasifikátor

5.1.2 SVM

velikost vektoru	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
50	59,52	37,70	40,97	57,14	59,36	37,88	49,3	48,42	38,2	60,01
100	48,89	48,85	28,33	70,39	48,9	48,85	38,4	59,85	29,22	69,44
200	37,06	61,25	21,27	77,76	35,38	63	29,08	69,6	23	75,96
500	22,25	76,75	15,05	84,26	20,43	78,64	20,79	78,27	16,98	82,25
1000	15,03	84,28	12,84	86,57	14,67	84,67	16,27	82,99	14,57	84,77
1500	13,14	86,26	11,76	87,71	12,71	86,71	14,26	85,1	13,57	85,81
2000	12,34	87,10	11,56	87,92	11,88	87,58	13,21	86,19	13,05	86,36
3000	11,28	88,21	11,23	88,26	11,12	88,37	11,8	87,67	13,06	86,35
4000	10,79	88,71	10,61	88,91	11,01	88,49	11,24	88,25	12,47	86,97
5000	10,55	88,97	10,3	89,23	10,76	88,75	10,82	88,69	10,36	89,17
6000	10,27	89,26	10,15	89,39	10,28	89,25	10,49	89,03	10,13	89,4
7000	10,33	89,20	9,99	89,55	10,48	89,04	10,45	89,07	10,02	89,53
all	9,31	90,26	9,29	90,29	9,51	90,06	9,66	89,9	9,46	90,1

Tabulka 5.2: Vliv velikosti příznakového vektoru na úspěšnost klasifikace pro klasifikátor SVM

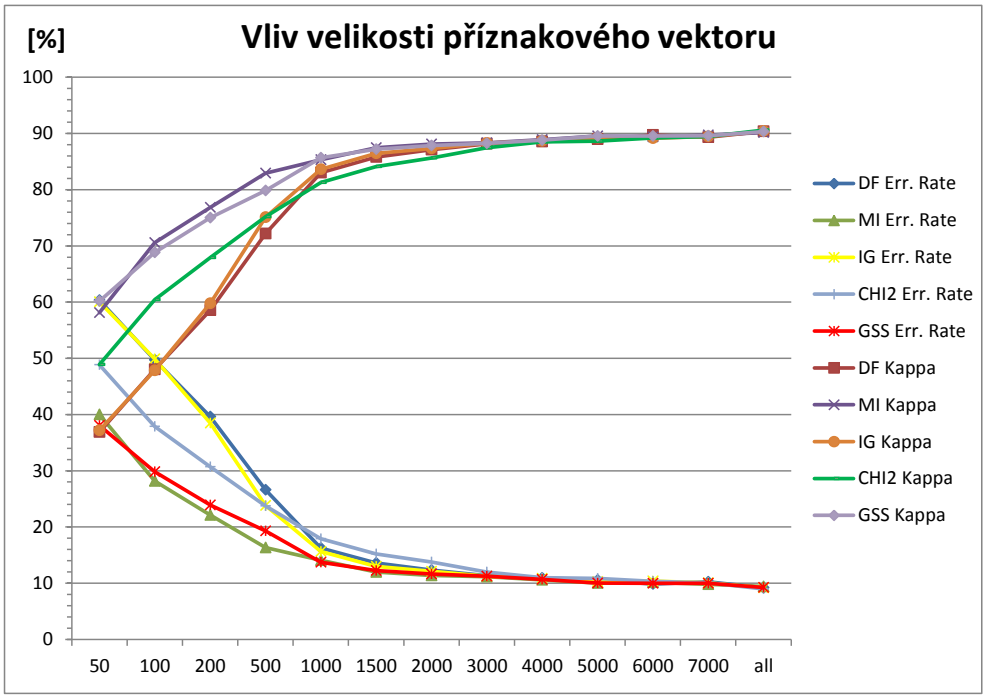


Obrázek 5.2: Graf závislosti velikosti příznakového vektoru na úspěšnost klasifikace pro klasifikátor SVM

5.1.3 Maximum Entropy

velikost vektoru	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
50	60,36	36,86	40,03	58,14	60,04	37,19	48,84	48,91	38,06	60,16
100	49,73	48,00	28,16	70,56	49,89	47,83	37,87	60,41	29,81	68,84
200	39,64	58,57	22,13	76,86	38,47	59,79	30,69	67,92	23,89	75,03
500	26,60	72,19	16,32	82,94	23,82	75,1	23,74	75,18	19,28	79,84
1000	16,23	83,03	14,05	85,31	15,65	83,63	17,92	81,27	13,71	85,66
1500	13,58	85,80	11,99	87,46	12,97	86,44	15,22	84,09	12,26	87,18
2000	12,32	87,12	11,35	88,13	12,1	87,35	13,75	85,62	11,64	87,83
3000	11,31	88,17	11,15	88,35	11,22	88,27	11,99	87,46	11,25	88,23
4000	10,95	88,55	10,6	88,91	10,83	88,68	10,99	88,5	10,69	88,83
5000	10,53	88,99	10,01	89,53	10,28	89,25	10,85	88,65	10,01	89,53
6000	9,84	89,71	10,16	89,38	10,36	89,17	10,32	89,21	9,98	89,56
7000	10,23	89,31	9,82	89,73	10,07	89,46	10,07	89,47	9,96	89,59
all	9,2	90,37	9,41	90,15	9,15	90,42	8,98	90,61	9,24	90,34

Tabulka 5.3: Vliv velikosti příznakového vektoru na úspěšnost klasifikace pro klasifikátor Maximum Entropy



Obrázek 5.3: Graf závislosti velikosti příznakového vektoru na úspěšnost klasifikace pro klasifikátor Maximum Entropy

Zhodnocení

Nejhorší výsledky, dle očekávání poskytoval Bayesův klasifikátor. Z grafů je vidět, že jeho Error rate křivky jsou posunuty zhruba o 5% k nižší úspěšnosti, nicméně čas trénování modelu byl oproti ostatním klasifikátorům výrazně kratší (řádově minuty). Naopak klasifikátory SVM a Maximum Entropy dosahovaly podobných výsledků, lišily se řádově pouze o desetiny procent. Trénování modelů pak trvalo řádově hodiny, přičemž klasifikátor Maximum Entropy pak přibližně trojnásobně delší dobu oproti SVM. Dále je z grafů vidět, že čím delší je délka vektoru, tím podobnější jsou výsledky klasifikace pro jednotlivé metody výběru příznaků. Tato vlastnost je dána především velkým počtem tříd do kterých klasifikujeme, protože metody výběru příznaků mohou preferovat slova jen z nějaké malé vybrané množiny tříd.

Na základě výsledků byla jako kompromis mezi časovou náročností trénování modelu a obdrženými výsledky stanovena následující délka vektoru. Pro Bayesův klasifikátor byl zvolen vektor délky 4000, pro zbývající klasifikátory se ukázala jako vhodná délka 3000. Následující experimenty, pokud nebude řečeno jinak, jsou prováděny s tímto nastavením, i když se pravděpodobnosti ještě o něco málo zlepšují.

5.2 Výběr vhodných POS tagů

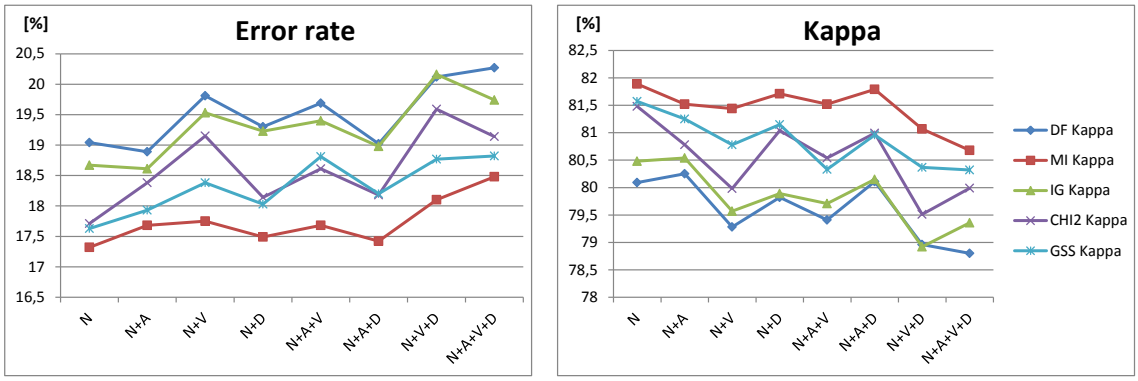
Dalším prováděným experimentem bylo zjišťování, jakým způsobem POS tagy ovlivňují úspěšnost klasifikace. V tabulkách jsou uvedeny výsledky klasifikace v % pro všechny kombinace POS tagů N, A, V, D, přičemž podstatná jména (N), jako slovní druh ovlivňující klasifikaci nejvíce (viz [3]), jsou použita vždy. Úspěšnosti jednotlivých klasifikátorů jsou vyneseny do grafů v závislosti na použitých kombinacích POS tagů. Křivky úspěšnosti jsou stejné pro oba grafy použitých metrik. Délky vektorů byly použity dle výsledků z předchozí kapitoly.

5.2.1 Naivní Bayesův klasifikátor

Výsledky experimentu jsou zobrazeny v tabulce 5.4 a vyneseny na grafu 5.4.

POS tagy	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
N	19,04	80,09	17,32	81,89	18,67	80,48	17,71	81,48	17,63	81,57
N,A	18,89	80,25	17,68	81,52	18,61	80,54	18,38	80,78	17,93	81,25
N,V	19,81	79,28	17,75	81,44	19,53	79,57	19,15	79,98	18,38	80,78
N,D	19,3	79,82	17,49	81,71	19,23	79,89	18,14	81,04	18,03	81,15
N,A,V	19,69	79,41	17,68	81,52	19,4	79,71	18,61	80,54	18,81	80,33
N,A,D	19,02	80,11	**17,42**	**81,79**	18,98	80,15	18,18	80,99	18,2	80,96
N,V,D	20,12	78,96	18,1	81,07	20,16	78,92	19,59	79,51	18,77	80,37
N,A,V,D	20,27	78,8	18,48	80,68	19,74	79,36	19,14	79,99	18,82	80,32

Tabulka 5.4: Vliv kombinace POS tagů na výsledky klasifikace pro Naivní Bayesův klasifikátor



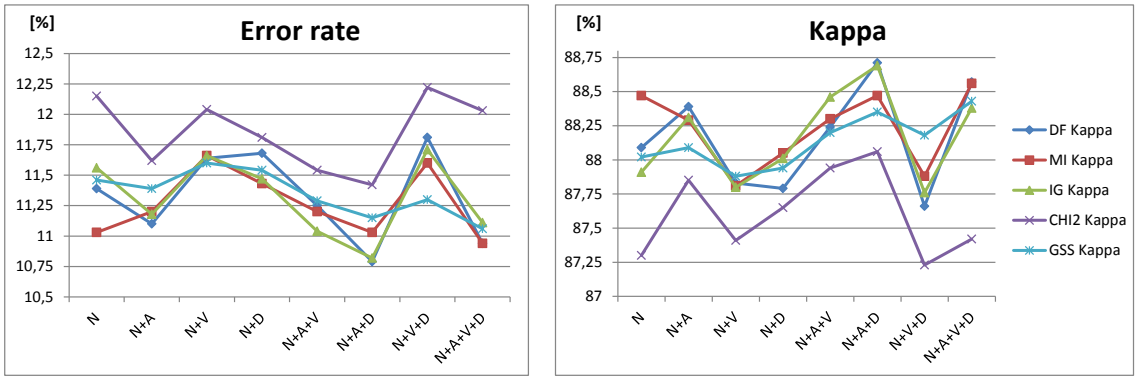
Obrázek 5.4: Graf závislosti kombinace POS tagů na výsledky klasifikace pro Naivní Bayesův klasifikátor

5.2.2 SVM

Výsledky experimentů jsou zobrazeny v tabulce 5.5 a vykresleny v grafu 5.5.

POS tagy	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
N	11,39	88,09	11,03	88,47	11,56	87,91	12,15	87,3	11,46	88,02
N,A	11,1	88,39	11,2	88,29	11,18	88,31	11,62	87,85	11,39	88,09
N,V	11,64	87,83	11,66	87,81	11,66	87,8	12,04	87,41	11,6	87,88
N,D	11,68	87,79	11,43	88,05	11,47	88,01	11,81	87,65	11,54	87,94
N,A,V	11,25	88,24	11,2	88,3	11,04	88,46	11,54	87,94	11,29	88,2
N,A,D	10,79	88,71	**11,03**	**88,47**	10,82	88,69	11,42	88,06	11,15	88,35
N,V,D	11,81	87,66	11,6	87,88	11,71	87,76	12,22	7,23	11,3	88,18
N,A,V,D	10,94	88,57	10,94	88,56	11,11	88,38	12,03	87,42	11,06	88,43

Tabulka 5.5: Vliv kombinace POS tagů na výsledky klasifikace pro klasifikátor SVM



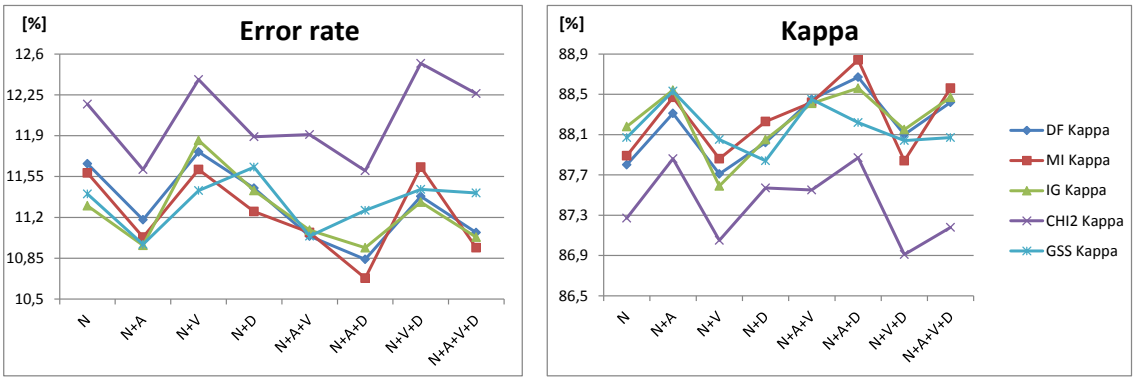
Obrázek 5.5: Graf závislosti kombinace POS tagů na výsledky klasifikace pro klasifikátor SVM

5.2.3 Maximum Entropy

Výsledky vlivu POS tagů jsou zobrazeny v tabulce 5.6 a vykreslena v grafu 5.6.

POS tagy	Metody výběru příznaků [výsledky v %]									
	DF		MI		IG		χ^2 test		GSS	
	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
N	11,66	87,8	11,58	87,89	11,3	88,18	12,17	87,27	11,4	88,07
N,A	11,18	88,31	11,03	88,47	10,96	88,54	11,61	87,86	10,97	88,53
N,V	11,76	87,71	11,61	87,86	11,86	87,59	12,38	87,05	11,43	88,05
N,D	11,45	88,02	11,25	88,23	11,43	88,05	11,89	87,57	11,63	87,84
N,A,V	11,04	88,45	11,07	88,42	11,09	88,41	11,91	87,55	11,04	88,45
N,A,D	10,84	88,67	**10,68**	**88,84**	10,94	88,56	11,6	87,87	11,26	88,22
N,V,D	11,38	88,1	11,63	87,84	11,33	88,15	12,52	86,91	11,44	88,04
N,A,V,D	11,07	88,42	10,94	88,56	11,03	88,47	12,26	87,18	11,41	88,07

Tabulka 5.6: Vliv kombinace POS tagů na výsledky klasifikace pro klasifikátor Maximum Entropy



Obrázek 5.6: Graf závislosti kombinace POS tagů na výsledky klasifikace pro klasifikátor Maximum Entropy

Zhodnocení

Z grafů pro jednotlivé klasifikátory je vidět, že o něco málo lepší je volit kombinaci bez sloves oproti prvnímu experimentu v kapitole 5.1, tedy kombinaci s ponecháním podstatných jmen, přídavných jmen a příslovcí (N, A, D). Veškeré následující experimenty budou prováděny s kombinací těchto POS tagů. Dále se ukázalo, že nejlepší je volit pro výběr příznaků metodu *Mutual Information*, která dosahovala v převážné většině experimentů nepatrně lepších výsledků (přibližně 0,5 % oproti další nejlepší). Nejlepších výsledků bylo dosaženo klasifikátorem Maximum Entropy, který měl přibližně o půl procenta lepší úspěšnost než klasifikátor SVM. Naopak nejhorší výsledek měl Naivní Bayesův klasifikátor, kde byl rozdíl cca 7 %.

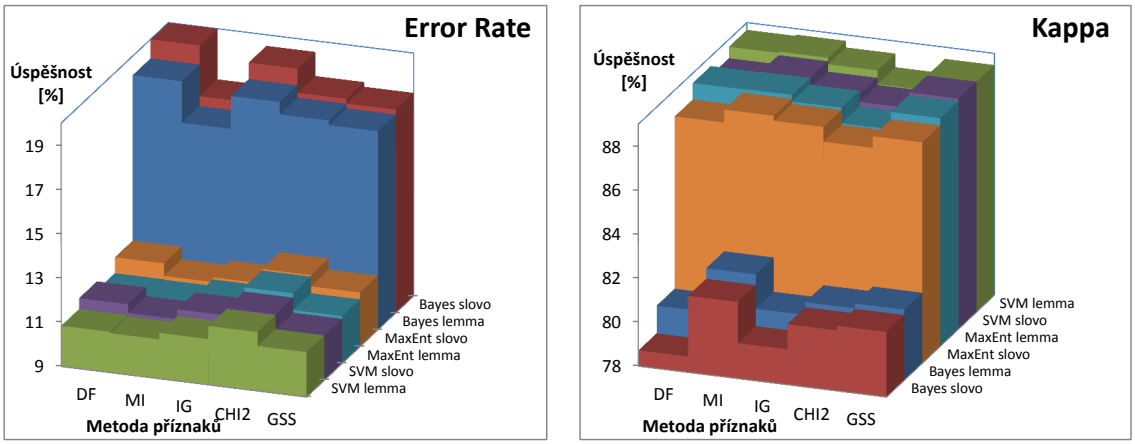
5.3 Vliv lemmat na úspěšnost klasifikace

Tato kapitola zobrazuje výsledky experimentu, při kterém byl zkoumán vliv lemmat na úspěšnost klasifikace. Výsledky experimentu jsou zobrazeny v tabulce 5.7 a vyneseny do grafu 5.7. Druhý sloupec tabulky značí, zda byla v testu použita lemmata (značka *'l'*) nebo slova (značka *'w'*).

Tento experiment zkoumá předpoklad, že použití lemmat poskytuje lepší výsledky než při použití slov. V 1. experimentu (viz kapitola 5.1) byla na základě tohoto předpokladu použita lemmata rovnou.

		Metoda výběru příznaků [výsledky v %]									
		DF		MI		IG		χ^2 test		GSS	
		Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa	Err.	Kappa
Bayes	l	19,14	79,98	17,26	81,96	18,75	80,4	18,19	80,98	17,98	81,2
	w	20,35	78,72	17,71	81,48	19,5	79,61	18,41	80,75	18,2	80,96
SVM	l	10,84	88,67	10,78	88,73	11,07	88,43	11,66	87,81	11,03	88,47
	w	11,33	88,15	10,93	88,57	11,38	88,1	11,7	87,77	11,02	88,48
MaxE	l	10,94	88,56	10,92	88,58	11,21	88,28	11,89	87,57	11,14	88,36
	w	11,66	87,81	11,09	88,4	11,31	88,17	11,96	87,5	11,45	88,03

Tabulka 5.7: Vliv lemmat na úspěšnost klasifikace pro jednotlivé klasifikátory



Obrázek 5.7: Graf zobrazující vliv lemmat na úspěšnost klasifikace

Zhodnocení

Z tabulky 5.7 či grafu 5.7 je vidět, že použití lemmat má drobný vliv na úspěšnost klasifikace. Úspěšnost se zlepšila přibližně o 1%. Pokud se podíváme do kapitoly

3.2.4 na statistiku korpusu, vidíme, že lemmatizace nám počet unikátních slov zredukovala přibližně o 20%, což při výběru vektoru slov o velikosti 3000 z množiny obsahující zhruba 150 000 slov je zanedbatelné. Vzhledem k alespoň drobnému zlepšení úspěšnosti rozpoznávání bude lemmatizace použita i v následujících experimentech a bude záležet na uživateli aplikace, zda bude lemmatizaci využívat či nikoliv.

5.4 Klasifikace do hlavní kategorie

V tabulce 5.8 jsou uvedeny výsledky klasifikace v procentech při použití různých klasifikátorů při optimálním výběru délky příznakového vektoru z kapitoly 5.1 a při neomezené délce příznakového vektoru. Dále byla použita lemmata a byly ponechány pouze POS tagy (N, A, D), které jak se ukázalo v kapitole 5.2, dosahovaly nejlepších výsledků. Dále jsem se omezil jen na výběr příznaků pomocí *Mutual Information (MI)* parametrizace, protože v předchozím testování předčila ostatní metody.

Délka vektoru:	4000	3000	3000	neomezena	neomezena	neomezena
Klasifikátor:	Bayess	SVM	MaxEnt	Bayess	SVM	MaxEnt
Error. rate	17,26	10,78	10,92	13,94	8,79	9,1
Kappa	81,96	88,73	88,58	85,42	90,81	90,48

Tabulka 5.8: Úspěšnost klasifikace pro hlavní kategorii při použití parametrizace MI vyhodnocená metrikou Error rate a Kappa

Zhodnocení

Z výsledků je vidět, že nejlepší výsledek poskytoval klasifikátor SVM na neomezené délce vektoru. Nicméně pokud velikost omezíme na 3000 slov, úspěšnost se příliš nezhorší a poměr *cena / výkon* výrazně vzroste. Je nutné si uvědomit, že *trénování modelu*[1] na neomezené délce vektoru je časově mnohem náročnější, trvá zhruba trojnásobný čas oproti omezené délce. Časová náročnost trénování modelů pro klasifikaci do více kategorií je v tabulce 5.12, pro klasifikaci do jedné kategorie je čas zhruba 3x menší.

5.5 Klasifikace do více kategorií

Pro vyhodnocení úspěšnosti klasifikace do více kategorií jsem použil metriky *Hamming loss* a metriku *Accuraccy* popsané v kapitole 3.3. Velikost příznakového vektoru byla použita stejná jako při klasifikaci do jedné kategorie v předchozí kapitole 5.4. Testy jsem prováděl pouze pro metodu výběru příznaků *MI*, protože ve všech předchozích testech dosahovala nejlepších výsledků. Výsledky jsou zobrazeny v tabulce 5.9.

[1] Trénování modelu prováděno na stroji s konfigurací v tabulce 5.11

Délka vektoru:	4000	3000	3000	neomezena	neomezena	neomezena
Klasifikátor:	Bayess	SVM	MaxEnt	Bayess	SVM	MaxEnt
Hamming	19,04	**9,44**	9,96	16,59	9,84	8,09
Accuracy	75,73	**87,34**	86,84	79,19	86,86	89,19

Tabulka 5.9: Úspěšnost klasifikace pro všechny kategorie při použití parametrizace MI vyhodnocená metrikou Hamming Loss a Accuracy

Dále byly obdržené výsledky z klasifikátoru vyhodnoceny se zahrnutím jedné kategorie navíc, než do kolika článek skutečně patří. Výsledky jsou zobrazeny v tabulce 5.10.

Délka vektoru:	4000	3000	3000	neomezena	neomezena	neomezena
Klasifikátor:	Bayess	SVM	MaxEnt	Bayess	SVM	MaxEnt
Hamming	13,3	**5,12**	5,54	11,47	5,57	4,4
Accuracy	79,49	**90,25**	89,77	82,49	89,71	91,66

Tabulka 5.10: Úspěšnost klasifikace pro všechny kategorie při použití parametrizace MI se zahrnutím jedné kategorie navíc vyhodnocená metrikou Hamming Loss a Accuracy

Zhodnocení

Při klasifikování do více kategorií znovu nejlépe dopadl klasifikátor SVM. Error rate byl v tomto případě *9,44 %* a již zde není velký rozdíl mezi omezeným a neomezeným vektorem (přibližně 0,5 % pro SVM a 2 % pro zbývající klasifikátory). Tato vlastnost je pravděpodobně způsobena větší trénovací množinou, protože každý článek se v trénovací množině vyskytuje tolikrát, do kolika tříd patří.

Pokud do vyhodnocení úspěšnosti zahrneme ještě jednu predikovanou kategorii navíc (viz tabulka 5.10, zjistíme, že úspěšnost se zlepší zhruba o 5 %. Jinými slovy, přibližně 50 % chybně rozpoznaných kategorií se nachází o jednu kategorii dále než by mělo, což může být způsobeno neúplně anotovanými daty. Dále se ukázalo, že neomezená délka vektoru zbytečně prodlužuje dobu klasifikace bez významného zlepšení úspěšnosti jak ukazuje následující kapitola.

Pokud úspěšnost klasifikace vyhodnotíme tak, že zahrneme ještě jednu predikovanou kategorii navíc, než do kolika článek patří, úspěšnost se zvýší cca. o 5 %.

5.5.1 Časová náročnost trénování modelu

Dalším kritériem pro výběr klasifikátoru je časová náročnost trénování modelu. Experimenty byly prováděny na počítači jehož konfigurace je uvedena v tabulce 5.11. Časová náročnost trénování a vyhodnocení modelu pro výsledky experimentů uvedených v tabulce 5.10 je uvedena v tabulce 5.12.

Operační systém	Linux Ubuntu 2.6.31-23
CPU	Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz
RAM	8 GB

Tabulka 5.11: Konfigurace počítače na kterém byly prováděny experimenty

Délka vektoru:	4000	3000	3000	neomezena	neomezena	neomezena
Klasifikátor:	Bayess	SVM	MaxEnt	Bayess	SVM	MaxEnt
Čas	3,6 min	4,5 h	11,89 h	9,4 min	10,5 h	31,8 h

Tabulka 5.12: Časová náročnost trénování modelů při dané konfiguraci experimentů

5.6 Shrnutí výsledků

Z prováděných experimentů v kapitole 5.1 byla zjištěna optimální délka vektoru slov pro trénovací a testovací soubory. Tato délka odpovídá hodnotě 4000 pro Bayesův klasifikátor a pro zbývající klasifikátory byla stanovena na hodnotu 3000.

V kapitole 5.2 jsem se pak zaměřil na výběr vhodných POS tagů. Na základě přečtené literatury jsem vybíral slova odpovídající všem kombinacím podstatných jmen, přídavných jmen, příslovcí a sloves, přičemž podstatná jména byla přítomna vždy, protože na klasifikaci mají největší vliv. Došel jsem k závěru, že pro klasifikaci je lepší použít kombinaci bez sloves.

V experimentu v podkapitole 5.3 jsem pak testoval vliv lemmat na klasifikaci. Při použití lemmat se úspěšnost již příliš nezlepšila, proto ji lze případně pro urychlení klasifikace vynechat. Na základě tohoto a předchozího experimentu byla také vybrána metoda *MI* pro výběr příznaků, která v testech dosahovala nejlepších výsledků.

Při klasifikaci do jedné, hlavní, kategorie se Error rate pohyboval kolem 17 % pro Naivní Bayesův klasifikátor a kolem 11 % pro ostatní klasifikátory na omezené délce vektoru. Pro neomezenou délku vektoru Error rate klesl zruba o 3 % pro Naivní Bayesův a o 2 % pro ostatní klasifikátory. Tento výsledek ale není pro tuto práci natolik důležitý, protože hlavní cíl je klasifikace do více kategorií.

Úspěšnost klasifikace do více kategorií vyhodnocená metrikou Hamming loss dosahovala hodnoty 19 % pro Naivní Bayesův klasifikátor a hodnoty mezi 9,5 % - 10 % pro ostatní klasifikátory na omezené délce vektoru. Pro neomezenou délku pak dokonce u klasifikátoru Maximum Entropy klesl k 8 %. Dále bylo zjištěno, že klasifikace pomocí Naivního Bayesova klasifikátoru trvala řádově minuty, zatímco SVM v řádech hodin a Maximum Entropy pak dokonce 3x déle než SVM. Samozřejmě doba trénování a testování je závislá na konfiguraci počítače. Pro tuto práci byla použita konfigurace zobrazená v tabulce 5.11.

Získaná data z klasifikace do více kategorií jsem ještě vyhodnotil tak, že jsem zahrnul jednu predikovanou kategorii navíc, abych zjistil, jak se změní úspěšnost klasifikace. Bylo zjištěno, že úspěšnost se zlepšila zhruba o 5 %, což pro klasifikátor SVM a Maximum Entropy činí 50 % zlepšení. Jinými slovy to znamená, že 50 % chybně klasifikovaných článků se nachází o jednu kategorii dále než by mělo.

Ve skutečnosti je ale úspěšnost klasifikace mnohem větší. Některé klasifikované články totiž obsahovaly pouze jednu či dvě krátké věty a klasifikátor pak nedokázal správně určit třídu klasifikace. Tyto články nebyly z trénovacích a testovacích množin filtrovány, protože cílem práce není dosáhnout co nejlepší úspěšnosti z hlediska čísel, ale co nejlepší úspěšnosti pro praktické použití. Chyba také mohla být způsobena neúplnou anotací množiny tříd, do kterých články ve skutečnosti patří. Chybějící třídy pak klasifikátor určil jako chybně klasifikované.

Po přečtení této kapitoly by mělo být čtenáři jasné, jakou kombinaci metod a nastavení klasfikačního systému použít jako kompromis mezi úspěšností a časovou náročností.

Osobně doporučuji použít klasifikátor SVM, metodu MI pro výběr příznaků na omezeném vektoru délky 3000 a POS-tagging pro filtrování příznaků s vynecháním lemmatizace, která je časově náročná a zlepšení je pouze v řádech desetin procenta.

Kapitola 6

Závěr

Toto téma vyplynulo z potřeb České tiskové kanceláře, se kterou jsem řešení také konzultoval. Cílem práce bylo prozkoumat metody klasifikace článků s podobným obsahem a navrhnout programové řešení, které by umožnilo klasifikovat dokumenty do více kategorií, tzv. *Multi-label* klasifikace.

Při výběru klasifikátorů a metod pro výběr příznaků byl kladen důraz na jejich úspěšnost, rychlost a dostupnost. Na základě přečtené literatury bylo zvoleno pět metod výběru příznaků (DF, MI, IG, CHI2, GSS) a tři klasifikátory (Naivní Bayesův, SVM a Maximum Entropy). Dále byl zvolen klasifikační nástroj *Minorthird* pro implementaci vybraných klasifikátorů. Nástroj byl vhodně přizpůsoben pro potřeby této práce, aniž by byla porušena licence a byly splněny podmínky pro využití ČTK.

Pro zlepšení úspěšnosti klasifikace byly použity metody morfologické analýzy textu, konkrétně se jednalo o *POS-tagging* a *lemmatizaci*. Z výsledků experimentů v kapitole 5 se ukázalo, že lemmatizace není pro klasifikaci příliš důležitá. Zlepšovala úspěšnost průměrně o 1 %. Naopak POS-tagging se ukázal jako velmi užitečný pro filtraci slov. Kombinace POS tagů, která poskytovala nejlepší řešení byla podstatná jména, přídavná jména a příslovce.

Dále jsem ukázal, že dostatečná délka vektoru příznaků je pro Naivní Bayesův klasifikátor 4000 a pro ostatní klasifikátory 3000. Při větší délce vektoru se prodlužuje čas klasifikace bez znatelného zlepšení úspěšnosti. Při prodlužující se délce vektoru se také stírají rozdíly mezi metodami výběru příznaků.

Nejlepší úspěšnosti bylo dosaženo s klasifikátorem SVM, kde metrika *Hamming loss* dosahovala vynikající hodnoty 9,44 %. Tento klasifikátor je proto doporučen ČTK pro klasifikaci článků do více kategorií spolu s nastavením uvedeným v kapitole 5.6. Nakonec byla implementována aplikace s grafickým rozhraním pro snadnou konfiguraci úlohy.

ČTK předpokládá nasazení a rozšíření klasifikačního systému o metody analýzy sentimentu. Při dalším vývoji nebude mít velký smysl zkoumat další metody výběru příznaků. Spíše by se pozornost měla věnovat výběru jiných příznaků, které by od sebe kategorie dokázaly lépe separovat. Za zkoušku by stálo prozkoumat možnost použití syntaktické struktury věty, pojmenovaných entit, atd.

Přehled termínů a zkratek

Anotace Ruční přiřazení kategorií k dokumentům

CRF Conditional random fields (podmíněná náhodná pole)

Cross-validation (křížová validace) Metodika rozdělování dat na disjunktní množiny pro trénování a testování klasifikátoru

ČTK Česká Tisková Kancelář

DF Document Frequency (dokumentová frekvence)

GSS Gallavotti, Sebastiani, Simi (metoda výběru příznaků pojmenovaná podle autorů)

HTML HyperText Markup Language (značkovací jazyk pro hypertext)

CHI2 Chi-kvadrát metoda pro výběr příznaků

IG Information Gain (informační zisk)

Lemmatizace Lemmatization (proces určování základního tvaru slov)

Lemma Základní podoba lexému, také slovníkový tvar

Metadata Strukturovaná data o datech

MI Mutual Information (metoda vzájemné informace)

Multi-class classification Klasifikace právě do jedné množiny z množiny definovaných tříd

Multi-label classification Klasifikace do více množin z množiny definovaných tříd

NLP Natural Language Procesing (zpracování přirozeného jazyka)

POS Part of Speech (slovní druhy)

SAX Simple API for XML (jednoduché rozhraní pro proudové zpracování XML)

Stematizace Stematization (proces určování základu slova)

STOP list Seznam slov, která nebudou při klasifikaci uvažována

SVM Support Vector Machine (metoda podpůrných vektorů)

Term Ekvivalentní výraz pro *Token*

Token Označuje slovo dokumentu

Tokenizace Převod novinového článku na seznam tokenů

XML Extensible Markup Language (rozšiřitelný značkovací jazyk)

Literatura

[1] ALIAS-I. LingPipe 4.1.0. 2008. Dostupné z: http://alias-i.com/lingpipe.

[2] ARIE – BEN-DAVID. Comparison of classification accuracy using Cohen's Weighted Kappa. *Expert Systems with Applications*. 2008, 34, 2, s. 825 – 832. ISSN 0957-4174. doi: 10.1016/j.eswa.2006.10.022. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0957417406003435.

[3] BEIL, F. – ESTER, M. – XU, X. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, s. 436–442, New York, NY, USA, 2002. ACM. doi: 10.1145/775047.775110. Dostupné z: http://doi.acm.org/10.1145/775047.775110. ISBN 1-58113-567-X.

[4] BJöRKELUND, A. – HAFDELL, L. – NUGUES, P. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, s. 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. Dostupné z: http://dl.acm.org/citation.cfm?id=1596409.1596416. ISBN 978-1-932432-29-9.

[5] BRATKO, A. – FILIPIČ, B. Exploiting structural information for semi-structured document categorization. In *Information Processing and Management*, s. 679–694, 2004.

[6] COHEN, W. W. MinorThird: Methods for Identifying Names and Ontological Relations in Text using Heuristics for Inducing Regularities from Data. 2004. Dostupné z: http://minorthird.sourceforge.net.

[7] COVER, T. – THOMAS, J. *Elements of information theory*. New York : Wiley, 1991.

[8] DELLA PIETRA, S. – DELLA PIETRA, V. – LAFFERTY, J. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997, 19, 4, s. 380–393. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=588021.

[9] DUŠEK, O. Deep Automatic Analysis of English. Diplomová práce, Charles University in Prague, Faculty of Mathematics and Physics, 2010.

[10] EXPRIT. Copyright Exprit s.r.o. Technická dokumentace InfoBanky ČTK. 1997.

[11] FINKEL, J. Stanford Named Entity Recognizer. 2005. Dostupné z: `http://nlp.stanford.edu/software/CRF-NER.shtml`.

[12] FORMAN, G. – GUYON, I. – ELISSEEFF, A. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*. 2003, 3, s. 1289–1305.

[13] GALAVOTTI, L. – SEBASTIANI, F. – SIMI, M. Experiments on the Use of Feature Selection and Negative Evidence in Automated Text Categorization. In *Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '00, s. 59–68, London, UK, UK, 2000. Springer-Verlag. Dostupné z: `http://dl.acm.org/citation.cfm?id=646633.699638`. ISBN 3-540-41023-6.

[14] HALL, M. et al. The WEKA data mining software: an update. *SIGKDD Explorations*. November 2009, 11, 1, s. 10–18. ISSN 1931-0145. Dostupné z: `http://dx.doi.org/10.1145/1656274.1656278`.

[15] JOACHIMS, J. T. Y. Y. Text categorization. *Scholarpedia*. 2008, 3, 5, s. 4242.

[16] JOACHIMS, T. SVMlight: Support Vector Machine. http://svmlight.joachims.org/, 2008.

[17] LIM, C. S. – LEE, K. J. – KIM, G. C. Multiple sets of features for automatic genre classification of web documents. *Information Processing and Management*. 2005, 41, 5, s. 1263 – 1276. ISSN 0306-4573. doi: 10.1016/j.ipm.2004.06.004. Dostupné z: `http://www.sciencedirect.com/science/article/pii/S0306457304000676`.

[18] LUO, X. – ZINCIR-HEYWOOD, A. N. Incorporating Temporal Information for Document Classification. In *ICDE Workshops*, s. 780–789, 2007.

[19] MANNING, C. D. – RAGHAVAN, P. – SCHüTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008. Dostupné z: `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0521865719`. ISBN 0521865719.

[20] MCCALLUM, A. K. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu, 2002.

[21] NIGAM, K. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, s. 61–67, 1999.

[22] NOVOVIČOVÁ, J. – MALÍK, A. – PUDIL, P. *Feature Selection using Improved Mutual Information for Text Classification*, 3138, s. 1010–1017. Springer, 2004. Dostupné z: `http://dx.doi.org/10.1007/978-3-540-27868-9_111`.

[23] FORMAL, I. – LINGUISTICS, A. *CoNLL-2009 Shared Task Description* [online]. 2008. [cit. 19. 3. 2012]. Dostupné z: `http://ufal.mff.cuni.cz/conll2009-st/task-description.html#Dataformat`.

[24] SEBASTIANI, F. Machine learning in automated text categorization. *ACM Comput. Surv.* March 2002, 34, 1, s. 1–47. ISSN 0360-0300. doi: 10.1145/505282.505283. Dostupné z: `http://doi.acm.org/10.1145/505282.505283`.

[25] TIMOTHY P. JURKA, A. E. B. E. G. L. C. – ATTEVELDT, W. RTextTools: Automatic Text Classification via Supervised Learning. http://www.rtexttools.com, 2011.

[26] TSOUMAKAS, G. – KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*. 2007, 3, 3, s. 1–13. Dostupné z: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.9401&rep=rep1&type=pdf`.

[27] UGUZ, H. A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowl.-Based Syst.* 2011, 24, 7, s. 1024–1032.

[28] WWW.IPTC.ORG. IPTC Web - NewsML, 2012. Dostupné z: `http://www.iptc.org/cms/site/single.html?channel=CH0087&document=CMS1206527546450`.

[29] YANG, Y. – PEDERSEN, J. O. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, s. 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. Dostupné z: `http://dl.acm.org/citation.cfm?id=645526.657137`. ISBN 1-55860-486-3.

[30] ZHOU, X. – ZHANG, X. – HU, X. Dragon Toolkit: Incorporating Auto-learned Semantic Knowledge into Large-Scale Text Retrieval and Mining. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2007. Dostupné z: `http://dragon.ischool.drexel.edu/`.

Přílohy

Příloha A

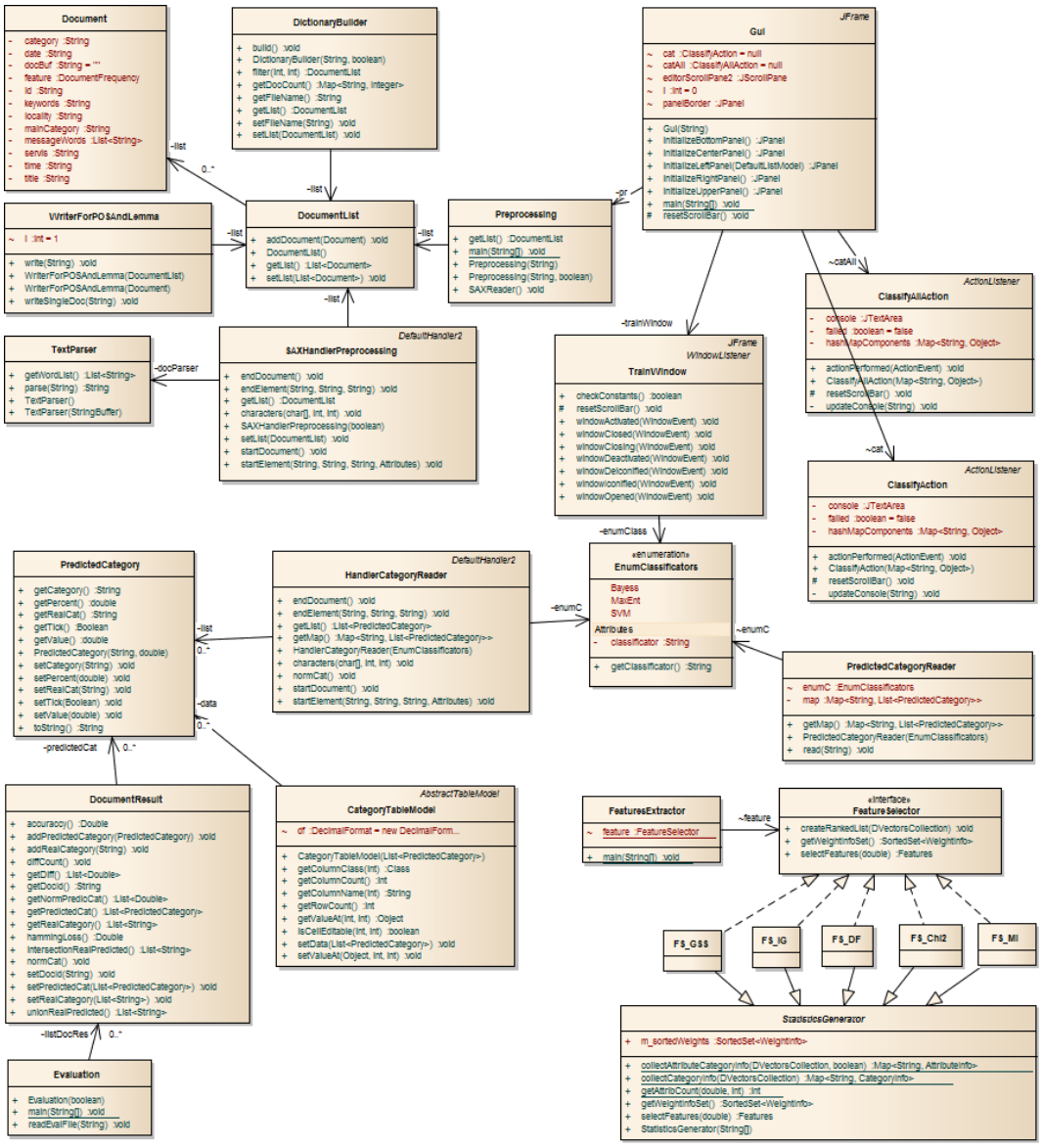
Struktura DVD

Na přiloženém DVD se nachází následující stromová struktura adresářů:

- adresář `doc`
  - adresář `src` – zdrojové LaTeXové soubory této práce
  - adresář `pdf` – tato práce v PDF formátu
- adresář `classification` – adresář se samotným programem
  - adresář `bin` – adresář s přeloženou aplikací
  - adresář `config` – obsahuje ukázkový konfigurační soubor aplikace
  - adresář `data` – adresář s daty (pouze ve verzi pro ČTK)
  - adresář `jar` – obsahuje *jar* soubory aplikace a upraveného nástroje *Minorthird*
  - adresář `javadoc` – adresář s javadoc dokumentací
  - adresář `lib` – knihovny využívané aplikací
  - adresář `models` – natrénované modely
  - adresář `src` – zdrojové soubory aplikace
  - `compile.bat` – kompilace zdrojových souborů pro Windows
  - `compile.sh` – kompilace zdrojových souborů pro Linux
  - `logging.properties` – soubor pro nastavení logování
  - `run.bat` – spuštění GUI aplikace pro Windows
  - `run.sh` – spuštění GUI aplikace pro Linux
- adresář `scripts` – adresář se skripty pro provedené experimenty
- adresář `results` – adresář s výsledky provedených experimentů včetně souboru s grafy

Příloha B

UML diagram aplikace



Obrázek B.1: UML diagram aplikace

Příloha C

Uživatelská příručka

Kapitola popisuje způsob přeložení programu, jeho konfiguraci a příklady spuštění.

C.1 Překlad programu

Program lze přeložit skriptem, který se nachází na přiloženém DVD v adresáři `classification`. K dispozici jsou dva soubory, `compile.bat` pro operační systém Windows a `compile.sh` pro operační systém Linux.

C.2 Konfigurace programu

Konfigurace programu je velmi jednoduchá a provádí se pomocí Java properties souboru. Jeho jednotlivé položky jsou vysvětleny níže.

- `POS_FILTER` - Seznam POS tagů, které budou ponechány. Oddělovač je mezera.
- `IGNORED_CATEGORIES` - Seznam kategorií které budou ignorovány.
- `USE_MULTI_CAT` - Hodnota *true* pro klasifikaci do více kategorií.
- `USE_WORDS` - Hodnota *true* pokud budou použita slova namísto lemmat.
- `USE_POS_TAG` - Hodnota *true* pokud bude použit POS tagging.
- `STOP_WORDS` - Slova, která budou vyfiltrována. Oddělovač je mezera.
- `EOL` - Znak ukončení řádku.
- `MIN_DOC_CAT` - Minimální počet článků na kategorii.
- `MAX_DOC_CAT` - Maximální počet článků na kategorii.
- `FEATURE_SERIALIZATION` - Soubor se serializovanými statistikami. Soubor je vytvářen při trénování modelu.

- `POS_TAG_MODEL` - Cesta k natrénovanému modelu pro POS-tagging.
- `LEMMA_MODEL` - Cesta k natrénovanému modelu pro lemmatizaci.
- `BAYESS_TRAINED_MODEL` - Cesta k natrénovanému (pro uložení) Baysovskému modelu.
- `SVM_TRAINED_MODEL` - Cesta k natrénovanému (pro uložení) SVM modelu.
- `MAXENT_TRAINED_MODEL` - Cesta k natrénovanému (pro uložení) Max. Ent. modelu.
- `TEMP_PARSER_FILE` - Cesta pro uložení/načtení parsovaného souboru.
- `TEMP_FILE_OUTPUT_LEMMA` - Cesta pro uložení/načtení výstupu lemmatizátoru.
- `TEMP_FILE_OUTPUT_POSS` - Cesta pro uložení/načtení výstupu POS taggeru.
- `FILE_FOR_CLASSIFICATOR` - Cesta pro uložení/načtení souboru pro klasifikátor.
- `PREDICTED_CATEGORY` - Cesta pro uložení výsledku klasifikace.
- `TRESHOLD` - Prahová hodnota pro určení správných kategorií.

C.2.1 Ukázková konfigurace

```
POS_FILTER=N A V D
USE_MULTI_CAT=true
TEMP_FILE_OUTPUT_LEMMA=temp/temp_output_lemma.txt
STOP_WORDS=. – , \: ? \! " \u201C \u201E ) ( { } \=
EOL=\r\n
MAX_DOC_CAT=600
FEATURE_SERIALIZATION=words_feature.ser
SVM_TRAINED_MODEL=models/SvmModel.ser
USE_WORDS=false
FILE_FOR_CLASSIFICATOR=class.txt
TEMP_PARSER_FILE=temp/temp_lemma.txt
BAYESS_TRAINED_MODEL=models/BayessModel.ser
PREDICTED_CATEGORY=class.xml
MAXENT_TRAINED_MODEL=models/MaxEntModel.ser
TEMP_FILE_OUTPUT_POSS=temp/temp_output_pos.txt
IGNORED_CATEGORIES=xhs den pla sue mnt eng bns
MIN_DOC_CAT=250
USE_POS_TAG=true
POS_TAG_MODEL=models/tag-cz.model
LEMMA_MODEL=models/lemma-cz.model
TRESHOLD=20
```

C.3 Příklady spuštění

Příklady spuštění budou ukázány pro *jar* soubor s aplikací, který v sobě obsahuje již zabalený klasifikátor a nástroj pro morfologickou analýzu. Tento soubor se nachází na přiloženém DVD na cestě `classification/jar/Classification.jar`.

Aplikace umožňuje veškeré experimenty konfigurovat a spouštět z grafického prostředí. Předpokládejme tedy, že se nacházíme v adresáři `classification`. Spuštění grafického prostředí provedeme příkazem

```
java -jar jar/Classification.jar config/prop.properties
```

případně můžeme použít spouštěcí skript `run.bat` pro OS Windows či `run.sh` pro OS Linux.

Pokud bychom nechtěli pracovat z grafického rozhraní, můžeme celý program od parsování, lemmatizaci, POS-tagging, extrakci příznaků až po samotnou klasifikaci konfigurovat ve skriptech obsahujících spouštění jednotlivých podprogramů. Toto použití je také doporučeno z důvodu rychlosti a konfigurace více experimentů najednou.

Pokud nebudeme chtít explicitně u každého programu při spuštění uvádět konfigurační soubor, můžeme vytvořit soubor `properties.properties` přímo u jar souboru celé aplikace. Tento soubor bude použit jako defaultní v případě, že nebude specifikován jiný.

- Parsování XML souboru a vygenerování vstupu pro lemmatizátor, POS-tagger nebo pro program extrahující příznaky můžeme provést příkazem

  ```
  java -cp jar/Classification.jar preprocessing.Preprocessing
  <input_file> <output_file> -conf <config_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar preprocessing.Preprocessing
  data/input.xml data/output.txt -conf config/prop.properties
  ```

- Spuštění lemmatizace lze provést příkazem

  ```
  java -cp jar/Classification.jar preprocessing.Lemmatization
  -model <model> -test <input_file> -out <output_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar preprocessing.Lemmatization
  -model ./models/lemma-cz.model -test data/output.txt -out
  data/lemma.txt
  ```

- POS-tagger spustíme příkazem

  ```
  java -cp jar/Classification.jar preprocessing.POSTagging -model
  <model> -test <input_file> -out <output_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar preprocessing.POSTagging -model
  ./models/tag-cz.model -test data/lemma.txt -out data/POS.txt
  ```

- Vygenerování souboru pro klasifikátor, ať už pro trénovací nebo testovací množinu, provedeme příkazem

  ```
  java -cp jar/Classification.jar
  featuresExtractor.FeaturesExtractor <input_file> <output_file>
  <feature_method> <feat_count> <file_type> -conf <config_file>
  ```

  - `<input_file>` - výstup z parseru, lemmatizátoru nebo POS-taggeru
  - `<output_file>` - soubor pro vstup do klasifikátoru
  - `<feature_method>` - metoda výběru příznaků, možné hodnoty: `DF, MI, CHI2, IG, GSS`
  - `<feat_count>` - délka příznakového vektoru
  - `<file_type>` - typ souboru, možné hodnoty: `train, test`. Při volbě trénovacího souboru se ukládají serializované statistiky pro tvorbu testovacího souboru, při druhé se naopak tyto statistiky načítají, takže musejí být vytvořeny.
  - `<config_file` - konfigurační soubor

  např.

  ```
  java -cp jar/Classification.jar
  featuresExtractor.FeaturesExtractor data/POS.txt data/train.txt
  MI 4000 train -conf config/prop.properties
  ```

- Trénování klasifikátoru

  ```
  java -Xmx4G -cp jar/Minorthird_updated.jar
  edu.cmu.minorthird.classify.Train -data <train_file> -learner
  <learner> -saveAs <model_path>
  ```

  - `<train_file>` - trénovací soubor s příznaky
  - `<learner>` - typ klasifikátoru který bude použit. Možné hodnoty:
    - `"new OneVsAllLearner(\"new NaiveBayes()\")"`

    * "new MaxEntLearner()"
    * "new SVMLearner()"
  - `<model_path>` - cesta, kam bude uložen natrénovaný model

  např.

  ```
  java -Xmx4G -cp jar/Minorthird_updated.jar
  edu.cmu.minorthird.classify.Train -data data/train.txt -learner
  "new OneVsAllLearner(\"new NaiveBayes()\")"-saveAs
  models/bayes.ser
  ```

- Testování klasifikátoru

  ```
  java -Xmx4G -cp jar/Minorthird_updated.jar
  edu.cmu.minorthird.classify.Test -test <test_file> -loadFrom
  <model_path> -saveAs <result_path> <result_type>
  ```

  - `<test_file>` - testovací soubor s příznaky
  - `<model_path>` - cesta odkud bude načten model
  - `<result_path>` - soubor do kterého budou uloženy výsledky
  - `<result_type>` - nepovinný parametr, určuje jakého typu budou výsledky. V kapitole 4.7 jsou popsány oba formáty souborů. Možné hodnoty: `-xml`.

  např.

  ```
  java -Xmx4G -cp jar/Minorthird_updated.jar
  edu.cmu.minorthird.classify.Test -test data/test.txt -loadFrom
  models/bayes.ser -saveAs results.eval -xml
  ```

  Pokud budeme klasifikovat do jedné kategorie, úspěšnost klasifikace metrikami *Error rate* a *Kappa* je vypsána do konzole. Pokud nepoužijeme parametr `-xml`, jsou výsledky zapsány i do souboru o struktuře uvedené v první podkapitole kapitoly 4.7.

  Pokud klasifikujeme do více kategorií, metrik *Error rate* a *Kappa* si nevšímáme a úspěšnost klasifikace je potřeba vyhodnotit jiným způsobem.

- Vyhodnocení úspěšnosti klasifikace do více kategorií je provedeno metrikami *Hamming loss* a *Accuracy*. Samozřejmě vyhodnocení funguje jen v případě znalostí kategorií do kterých dokument skutečně patří.

  ```
  java -cp jar/Classification.jar evaluation.Evaluation
  <result_path>
  ```

  např.

  ```
  java -cp jar/Classification.jar evaluation.Evaluation
  results.eval
  ```

- Promíchání výstupních souborů tříd balíku *preprocessing*, výsledek je zapsán do stejného souboru.

  ```
  java -cp jar/Classification.jar utils.Shuffle <file> -conf
  <config_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar utils.Shuffle data/POS.txt -conf
  config/prop.properties
  ```

- Převedení souboru s výsledky klasifikace vyjádřenými koeficientem důvěry na procentuální úspěšnost.

  ```
  java -cp jar/Classification.jar utils.ConvertXMLResults
  <input_file> <output_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar utils.ConvertXMLResults
  results.eval results_percent.eval
  ```

- Filtrace výstupu souborů tříd balíku *preprocessing*. Ponechány jsou pouze články obsahující *ID* v souboru předávaným jako druhý parametr. Každý řádek souboru obsahuje jeden *ID* dokumentu.

  ```
  java -cp jar/Classification.jar utils.FiltrationOrderID
  <input_file> <file_with_IDs> <output_file>
  ```

  např.

  ```
  java -cp jar/Classification.jar utils.FiltrationOrderID
  data/POS.txt data/ids.txt data/filtered.txt
  ```

Příloha D

Tabulky textové databáze ČTK

Aby si čtenář udělal představu o databázi ČTK, je zde popsána její základní struktura tak, jak se nachází v [10].

D.1 Přehled databází

D.1.1 Databáze periodik

Název	**Tabulka**
TV – Rozhlas	ACONNECT
Blesk	BLESK
Brněnský den	BRNODEM
Haló noviny	HALO
Hospodářské noviny	HOSPN
Lidové noviny	LN
Mladá fronta DNES	MF
Moravský den	MODEN
Moravskoslezský den	MOSI
Práce	PRACE
Právo	PRAVO
Profit	PROFIT
Reflex	REFLEX
Respekt	ESPEKT
Rovnost	ROVNOST
Svobodné slovo	SLOVO
Svoboda	SVOBODA
Telegraf	TELEGRAF
Všechny noviny	MCNOVINY
Zemské noviny	ZN

Tabulka D.1: Struktura databáze periodik

D.1.2 Dokumentační databáze

Název	**Tabulka**	**Popis**
Biografie	BIO	Životopisy slavných osobností
Česká republika	CR	Informace o České republice
Očekávané události	OCEK	Databáze očekávaných událostí
Sportovní rekordy	SPORTEK	Datbáze sportovních rekordů
Sport	SPORTY	Databáze sportovních informací
Svět	SVET	Databáze všeobecných informací
Výročí	VYROCI	Databáze významných výročí
Země	ZEME	Informace o zemích

Tabulka D.2: Struktura dokumentační databáze

D.1.3 Databáze agenturních zpráv

Název	**Tabulka**
Aktuální zpravodajství	AKTU
Zpravodajství z roku 88	FOND88
Zpravodajství z roku 89	FOND89
Zpravodajství z roku 90	FOND90
Zpravodajství z roku 91	FOND91
Zpravodajství z roku 92	FOND92
Zpravodajství z roku 93	FOND93
Zpravodajství z roku 94	FOND94
Zpravodajství z roku 95	FOND95
Zpravodajství z roku 96	FOND96
Zpravodajství z roku 97	FOND97
Zpravodajství od začátku roku včetně Aktualit	ROK
Všechno zpravodajství	MFONDY

Tabulka D.3: Struktura databáze agenturních zpráv

D.2 Schémata tabulek

Schémata tabulek databází uvedených v předchozí kapitole E.1 jsou dány tabulkami uvedenými v následujících podkapitolách.

D.2.1 Tabulka typu FOND, AKTU

Název sloupce	**Popis**	Typ
ID	Identifikátor	VARCHAR(100)
D	Datum vzniku	DATE
TI	Čas vzniku	INTEGER
O	Od agentury	VARCHAR(100)
A	Autor zprávy	VARCHAR(256)
E	Redaktor zprávy	VARCHAR(256)
L	Jazyk	VARCHAR(100)
P	Priorita zprávy	INTEGER
SV	Zdrojový kód	VARCHAR(4)
CA	Kód kategorie	CHAR(4)
BU	Odkaz na bulletin	VARCHAR(260)
K	Klíčová slova	VARCHAR(512)
ADR	Adresy výstupu	VARCHAR(260)
COM	Komentář	VARCHAR(260)
WC	Počet slov	INTEGER
REM	Poznámka	VARCHAR(256)
FT_TEXT	Text zprávy	APVARCHAR
DAY	Počet dní	INTEGER
ST	Typ opravy	CHAR
LDI	Log. identifikátor DB	CHAR

Tabulka D.4: Schéma tabulky typu FOND, AKTU

D.2.2 Tabulka typu CR

Název sloupce	**Popis**	Typ
K	Tématická rubrika	VARCHAR(100)
KC	Kód tematické rubriky	VARCHAR(100)
TT	Titulek	VARCHAR(256)
F	Text záznamu	APVARCHAR

Tabulka D.5: Schéma tabulky typu CR

D.2.3 Tabulka typu Periodikum

Název sloupce	**Popis**	Typ
ZDROJ	Periodikum	VARCHAR(100)
D	Datum vydání	DATE
STRANA	Strana v periodiku	INTEGER
K	Rubrika	VARCHAR(100)
FT_TEXT	Text	APVARCHAR
TT	Nadpis	VARCHAR(256)

Tabulka D.6: Schéma tabulky typu Periodikum (časopis a noviny)

D.2.4 Tabulka typu BIO

Název sloupce	**Popis**	Typ
JM	Jméno	VARCHAR(256)
K	Klíčová slova	VARCHAR(256)
TT	Titulek	VARCHAR(256)
LO	Lokalita	VARCHAR(256)
D	Datum	DATE(256)
VY	Datum výročí	DATE(256)
ZP	Jméno zpracovatel	VARCHAR(100)
DATUMNAR	Datum narození	DATE
DATUMUMR	Datum úmrtí	DATE
F	Text záznamu	APVARCHAR

Tabulka D.7: Schéma tabulky typu BIO

D.2.5 Tabulka typu SVET

Název sloupce	**Popis**	Typ
LO	Lokalita	VARCHAR(100)
K	Tematická rubrika	VARCHAR(100)
KC	Kód tematické rubriky	VARCHAR(256)
TT	Titulek	VARCHAR(256)
F	Text záznamu	APVARCHAR

Tabulka D.8: Schéma tabulky typu SVET

D.2.6 Tabulka typu SPORTREK

Název sloupce	**Popis**	Typ
SP	Název sportu	VARCHAR(256)
DISC	Sportovní disciplína	VARCHAR(256)
VYKON	Výkon	VARCHAR(256)
JM	Příjmení, jméno	VARCHAR(256)
SEX	Pohlaví (M nebo W)	VARCHAR(256)
LC	Kód země podle IAAF	VARCHAR(256)
LCC	Zkratka světadílu	VARCHAR(256)
LO	Místo dosažení	VARCHAR(256)
ROK	Čtyřčíslí roku	VARCHAR(256)
D	Datum dosažení	DATE
REKORD	Kód rekordu	VARCHAR(256)
P1	Pořadí v soutěží	VARCHAR(256)
P2	Rozběh, semifinále	VARCHAR(256)
ZAVOD	Závod /zkratka/	VARCHAR(256)
F	Text zprávy	APVARCHAR
KC	Kód výkonu	VARCHAR(256)

Tabulka D.9: Schéma tabulky typu SPORTREK

D.2.7 Tabulka typu SPORTY

Název sloupce	**Popis**	Typ
SP	Název sportu	VARCHAR(256)
DS	Sportovní disciplína	VARCHAR(256)
SO	Název soutěže	VARCHAR(256)
ROK	Rok soutěže	VARCHAR(256)
P	Pohlaví (M,W,S)	VARCHAR(256)
LO	Lokalita soutěže	VARCHAR(256)
TT	Název	VARCHAR(256)
F	Text zprávy	APVARCHAR

Tabulka D.10: Schéma tabulky typu SPORTY

D.2.8 Tabulka typu VYROCI

Název sloupce	**Popis**	Typ
K	Rubrika (D, Z)	VARCHAR(100)
DEN	Den v měsíci (1 - 31)	VARCHAR(100)
MES	Měsíc (1 - 12)	VARCHAR(100)
ROK	Koncové, číslo roku	VARCHAR(100)
TT	Název	VARCHAR(256)
F	Text záznamu	APVARCHAR

Tabulka D.11: Schéma tabulky typu VYROCI

D.2.9 Tabulka typu ZEME

Název sloupce	**Popis**	Typ
LO	Název země	VARCHAR(100)
K	Tematická rubrika	VARCHAR(100)
LC	Kódové označení země	VARCHAR(256)
KC	Kód tematické rubriky	VARCHAR(256)
F	Text záznamu	APVARCHAR
TT	Titulek	VARCHAR(256)

Tabulka D.12: Schéma tabulky typu ZEME

Příloha E

Přehled kategorií článků

1	aut	Automobilový průmysl	31	nab	Náboženství
2	bos	Business News	32	obo	Obchod
3	bsk	Sklářský průmysl	33	odb	Práce a odbory
4	bua	Burzy akciové	34	pit	Telekomunikace a IT
5	buk	Burzy komoditní	35	pla	Plány zpravodajství ČTK
6	bup	Burzy peněžní	36	pod	Politika ČR
7	bur	Burzy peněžní	37	pol	Politika
8	cen		38	prg	Pragensie
9	che	Chemický	39	prm	Lehký průmysl
		a farmaceutický průmysl	40	ptr	Potravinářství
10	den	Zpravodajské deníky	41	reg	Region
11	dpr	Doprava	42	sko	Školství
12	dre	Dřevozpracující průmysl	43	slo	Slovenika
13	efm	Firmy	44	slz	Služby
14	ekl	Životní prostředí	45	sop	Sociální problematika
15	eko	Ekologie	46	spc	
16	ene	Energie	47	spl	Životní styl
17	eur	Evropská unie - zprávy	48	spo	Sportovní zpravodajství
18	fin	Finanční služby	49	sta	Stavebnictví a reality
19	for	Parlamenty a vlády	50	str	Strojírenství
20	fot	Fotbal - zprávy	51	sur	Suroviny
21	hok	Hokej - zprávy	52	tlk	Telekomunikace
22	hut	Hutnictví	53	tok	Textil
23	kat	Neštěstí a katastrofy	54	tur	Cestovní ruch
24	kul	Kultura	55	vat	Věda a technika
25	mag	Magazínový výběr	56	zah	Zahraniční
26	mak	Makroekonomika	57	zak	Kriminalita a právo
27	med	Média a reklama	58	zbr	Zbraně
28	met	Počasí	59	zdr	Zdravotnictví
29	mix	Mix	60	zem	Zemědělství
30	mot	Monitor			

Tabulka E.1: Všechny dostupné kategorie článků