VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

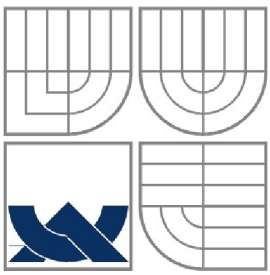
# KLASIFIKACE TEXTU POMOCÍ METODY SVM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

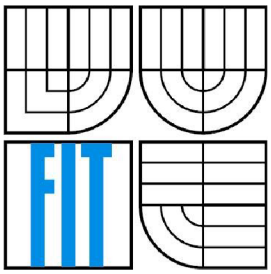
AUTOR PRÁCE
AUTHOR

Bc. RADOVAN SYNEK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# KLASIFIKACE TEXTU POMOCÍ METODY SVM

TEXT CLASSIFICATION WITH THE SVM METHOD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADOVAN SYNEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2010

# Abstrakt

Tato práce pojednává o dolování v textových datech. Zaměřuje se na problematiku klasifikace dokumentů a techniky s tím spojené, především předzpracování dat. Dále je představena metoda SVM, která byla zvolena pro samotnou klasifikaci, návrh a testování implementované aplikace.



# Abstract

This thesis deals with text mining. It focuses on problems of document classification and related techniques, mainly data preprocessing. Project also introduces the SVM method, which has been chosen for classification, design and testing of implemented application.



# Klíčová slova

Dolování v textu, dokument, kategorizace, klasifikace, předzpracování, SVM



# Keywords

Text mining, document, categorization, classification, preprocessing, SVM



# Citace

Radovan Synek: Klasifikace textu pomocí metody SVM, diplomová práce, Brno, FIT VUT v Brně, 2010

# Klasifikace textu pomocí metody SVM

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radovan Synek
Datum (15.5.2010)

## Poděkování

Děkuji Ing. Bartíkovi, Ph.D. za odborné vedení této práce, za jeho připomínky a rady, které mi při řešení velmi pomohly. Dále děkuji Mgr. Rychlému, Ph.D. za poskytnutí konzultace k architektonickému vzoru MVC.

# Obsah

# 1 Úvod

Moderní svět s sebou nese záplavu informací. Každá organizace v rámci své činnosti produkuje a zpracovává značné množství dokumentů a toto množství každým rokem narůstá. Jsou to firmy působící v sektoru bankovnictví a financí, veškeré součásti státní správy, zdravotnické subjekty. Příklady dokumentů jsou smlouvy s klienty, zdravotní záznamy, písemné žádosti a formuláře, vědecké články. Spousta dokumentů již vzniká v elektronické podobě a často se lze také setkat s řešením digitalizace papírových dokumentů pro jejich další zpracování i uskladnění.

Takové množství dokumentů samozřejmě musí být nějak organizováno do menších, přehlednějších celků. Většinou jsou to zaměstnanci, kdo se musí vypořádat s tříděním dokumentů, což je časově náročné, avšak existují už organizace, kde takové třídění probíhá automaticky. Uvážíme-li, že množství dokumentů bude i nadále narůstat, brzy bude prakticky nemožné, aby se s jejich tříděním někdo vypořádal manuálně. Přitom právě skutečnost, že dnes už většina dokumentů existuje v elektronické podobě, nahrává myšlence jejich automatického třídění a dost možná i dalšího zpracování.

Nejsou to však jen dokumenty spjaté s činností firem, využitím klasifikace dokumentů může být i filtrování spamu, který tvoří až 90 % z celkového objemu poslaných e-mailů. Možnosti uplatnění jsou široké, aby však lidská práce – i když mnohdy nadbytečná a rutinní – mohla být nahrazena počítačovým zpracováním, je nezbytné, aby byl počítač v této činnosti alespoň stejně úspěšný jako člověk. Chybná klasifikace totiž může vyústit v problém, který musí vyřešit člověk a škoda problémem způsobená může zastínit výhody automatické klasifikace.

Cílem v oblasti klasifikace dokumentů je tedy zkoumání a testování různých metod klasifikace a předzpracování, které přinesou co nejlepší výsledky na textových datech z různých domén.

V kapitole 2 jsou uvedeny nejčastější úlohy dolování v textu, blíže je rozebrána kategorizace dokumentů a její aplikace. Pro kategorizaci je znázorněn základní princip nejčastěji používaných klasifikátorů. V kapitole 3 je podrobně vysvětlena metoda SVM, její základní varianta pro čistě lineárně separovatelná data a úprava metody pro data, kde několik vzorků znemožňuje lineární rozdělení. Dále je uveden princip řešení při zcela nelineárních datech, který využívá jádrových funkcí pro mapování do nového prostoru vyšší dimenze. Nakonec jsou popsány některé metody řešení kvadratického optimalizačního problému a více rozebrán algoritmus SMO sloužící k trénování SVM klasifikátoru.

Kapitola 4 obsahuje návrhové diagramy a jejich popis. Pro znázornění struktury na nejvyšší úrovni slouží diagram balíčků, detailnější strukturu zachycují diagramy tříd, které pokrývají celou aplikaci. Diagramy sekvence byly uvedeny pouze pro nejzajímavější akce, které aplikace provádí. Na tuto kapitolu velmi těsně navazuje kapitola 5, která objasňuje některé postupy použité

při implementaci. Především je to popis zpracování souborů datasetu Reuters, který byl zvolen pro testování aplikace, dále princip řešení selekce rysů a výpočtu k tomu potřebných metrik.

Obsahem další kapitoly je testování aplikace. Nejprve jsou popsány použité nástroje a průběh testování. Implementovaný SVM klasifikátor je porovnáván s jiným klasifikátorem založeným na stejné metodě – SVM light. Rovněž je otestována úspěšnost jiných klasifikátorů, k čemuž byl použit dolovací nástroj Weka. Kromě srovnání různých klasifikačních metod byl také vyhodnocen vliv selekce rysů.

Poslední kapitola shrnuje výsledky této diplomové práce, význam metody SVM pro klasifikaci dokumentů a uvádí náměty na případná rozšíření implementované aplikace.

# 2 Dolování v textových datech

Obor zvaný datamining – dolování v datech, vznikl jako reakce na rozrůstající se objem dat uchovávaných v elektronické podobě. To s sebou přineslo i problém, jak v takovém rozsáhlém množství dat rozpoznat užitečné informace. Relační databázové systémy, které se staly dominantou trhu, se ukázaly samotné jako nedostačující. Do popředí se tedy dostávají algoritmy a metody, jak v datech nalézt netriviální znalosti, které budou pro uživatele zajímavé. Dolování v datech v sobě spojuje několik dalších oborů, jako je umělá inteligence, statistika a samozřejmě databázové technologie. Typickým zdrojem dat pro dolování je relační databáze, případně datový sklad.

Dolování v textových datech je speciálním případem dolování dat, má svoje specifika, kterými se od výše zmíněného odlišuje. Především zdrojem dat není relační databáze, ale množina dokumentů ve formátu čitelném pro člověka. To představuje data, která jsou strukturována velmi málo nebo vůbec. Stěžejní částí procesu dolování v textu se tak stává předzpracování, které z dokumentů vytvoří data použitelná jako vstup pro algoritmy vyhledávání asociačních pravidel, klasifikaci, apod.

## 2.1 Základní pojmy

V následujících podkapitolách jsou vysvětlené důležité pojmy z oblasti dolování v textových datech. Jsou to pojmy dokument a kolekce dokumentů, dále jsou rozebrány rysy, které dokument utvářejí.

### 2.1.1 Kolekce dokumentů

Kolekce dokumentů se rozlišují na statické a dynamické. Ve statické kolekci zůstávají všechny dokumenty nezměněné, zatímco v dynamické je třeba počítat se změnami - to značí úpravy, mazání i přidávání dokumentů.

### 2.1.2 Dokument

Dokumentem se rozumí konkrétní textová data obvykle spjatá se skutečným dokumentem v reálném světě. Dokument může být současně obsažen i ve více kolekcích. Na dokumenty lze nahlížet jako na strukturované objekty – zpravidla mají nadpis, autora, dělení do odstavců, atd. V takovém případě se jedná o dokumenty slabě strukturované. Příhodnější jsou dokumenty, pro něž je typické použití formátovacích značek nebo závislost na šablonách (HTML stránky, PDF dokumenty, e-maily, …). Takové dokumenty se označují jako polostrukturované.

### 2.1.3 Rysy dokumentu

Dokument je tvořen množinou rysů, jejíž kardinalita bývá značně vysoká (počet rysů dokumentu se může pohybovat v řádu desítek tisíc). Proto je snahou vybrat vhodnout podmnožinu, která bude dokument co nejlépe reprezentovat. Další typickou vlastností textových dat je řídkost rysů – každý dokument obsahuje jen zlomek z celkového množství rysů, které se nachází v celé kolekci.

Nejzákladnějšími rysy jsou jednotlivé znaky. Samotná existence znaků v dokumentu bez informace o jejich pozici nepřináší příliš užitku, používá se spíš znaková reprezentace dokumentu, která nějakou informaci o pozici znaků obsahuje. Tou může být například vyhledávání bigramů a trigramů. I tak jsou znaky jako rysy dokumentu používány zřídka.

Častější je využití celých slov, případně termů (zahrnují slova a víceslovné fráze). Pro detekci frází je obvykle nutné použít externí slovník, ale takto získaná reprezentace dokumentu je sémanticky bohatší.

Nejvýše v hierarchii rysů stojí koncepty. Jsou to rysy získané statisticky, ručně nebo pomocí předem definovaných pravidel, které zahrnují slova, fráze i větší celky. Na rozdíl od reprezentace tvořené slovy nebo termy, koncepty mohou obsahovat slova, která se v původním dokumentu nevyskytují. Použití konceptů jako rysů dokumentu není příliš rozšířené z důvodu jejich komplikovaného získávání.

## 2.2 Techniky předzpracování

Efektivnost dolování v textových datech je silně vymezena kvalitou předzpracování. K přípravě dat pro klasické dolování a pro dolování v textu se požadují velmi rozdílné techniky. Zde je základním úkolem předzpracování dát dokumentu nějakou strukturu. Na začátku je strukturovanost dokumentu velmi nízká a různé techniky předzpracování ji postupně obohacují. Z původního dokumentu vybírají důležité rysy, na konci zůstávají jen ty, které co nejlépe dokument reprezentují, zatímco ostatní rysy jsou zahozeny.

Rozlišují se techniky obecné, nezávislé na úloze, a techniky, které využívají konkrétní informace o dokumentech a váží se tak ke konkrétní úloze. Do první skupiny patří NLP (Natural Language Processing), do druhé skupiny kategorizace a IE (Information Extraction). Následuje krátký popis těchto technik, který čerpá z [1].

### 2.2.1 Techniky NLP

**Tokenizace**

Dokument může být rozdělen do kapitol, odstavců, vět, slov a slabik. Nejčastěji se však používá dělení do vět a slov. Rozdělení dokumentu do slov je triviálním úkolem, avšak při dělení do vět je nutné se vypořádat s rozlišením tečky jako konce věty nebo jako slovní zkratky.

**Part-of-Speech Tagging**

POS značkování přiděluje slovům značky na základě jejich významu ve větě. Je to známé dělení na jména, slovesa, příslovce, předložky, číslovky. Tyto značky jsou předem dány a můžou se v různých implementacích lišit.

**Syntaktická analýza**

Jedná se o syntaktickou analýzu dokumentu na základě předem definované gramatiky. Tato analýza může být buď úplná – což je ale výpočetně velmi náročné – nebo mělká. Mělká analýza si všímá pouze jasných a jednoznačných částí dokumentu, dokáže rozpoznat jednodušší fráze, zatímco komplexnější zůstávají bez povšimnutí. Toto pojetí je obvykle plně dostačující a výhodou je naopak významné snížení výpočetních nároků takové analýzy.

### 2.2.2 IE

Information Extraction patří k nejvýznamnějším technikám, které se váží k dolování v textových datech. Lze na ni pohlížet jako na součást techniky IR (Information Retrieval). IR na základě zadaného dotazu nalezne odpovídající dokument, ale IE jde o něco dál – z dokumentu vybere pouze související informace, které ve vhodně strukturované podobě zobrazí uživateli.

## 2.3 Klasifikace / kategorizace dokumentů

Klasifikace dokumentů je proces, který zařazuje dokumenty z kolekce do předem známých kategorií. Formálně ho lze definovat jako funkci přiřazení takto:

$$F : D \times C \rightarrow \{0, 1\} \quad , \tag{2.1}$$

kde $D$ je množina všech dokumentů, $C$ je množina všech kategorií. Potom $F(d, c)$ nabývá hodnoty 1 právě tehdy, když dokument $d$ patří do kategorie $c$, jinak nabývá hodnoty 0.

Rozlišuje se single-label a multi-label klasifikace. Při single-label klasifikaci může dokument patřit právě do jedné kategorie, zatímco v případě multi-label klasifikace může patřit i do více kategorií současně, tím se kategorie překrývají.

### 2.3.1 Reprezentace dokumentu

Jak již bylo řečeno, dokument je reprezentován množinou rysů, přesněji řečeno, tuto množinu lze chápat jako vektor rysů, který dokument určuje v prostoru rysů. Tyto rysy, ať už jsou to slova, kořeny slov nebo fráze, musíme nejprve převést do číselné podoby, než je předáme klasifikátoru k posouzení, do jaké kategorie dokument patří.

V případě použití slov jako rysů potom dostáváme dimenzi prostoru rysů rovnou počtu všech různých slov v celé kolekci. Nejjednodušší reprezentací je reprezentace binární – zda dokument daný rys obsahuje nebo ne. Lepší je použití schématu TF (Term Frequency) nebo TF-IDF (Term Frequency – Inverse Document Frequency), které se počítá podle vzorce:

$$TF-IDF = TF(w,d)\cdot\log\left(\frac{N}{DF(w)+1}\right) \quad , \tag{2.2}$$

kde $TF(w,d)$ je TF slova $w$ v dokumentu $d$, $N$ je počet všech dokumentů v kolekci a $DF$ je Document Frequency, tedy počet dokumentů, které obsahují slovo $w$.

### 2.3.2 Výběr rysů

Počet rysů v dokumentech je značný, přesto jen některé z nich mají užitek pro klasifikaci, zatímco většina rysů je irelevantní. Je proto žádoucí tyto rysy z dalšího zpracování odstranit. Obvykle to neznamená žádné zhoršení přesnosti, naopak lze očekávat zlepšení, protože se jedná o data obsahující šum, která mohou klasifikátor ovlivnit v negativním smyslu. Jejich odstraněním se získá další výhoda – snížení dimenzionality prostoru rysů bezpochyby urychlí další výpočty a tím i celý proces klasifikace.

Základní technikou selekce rysů je eliminace slov označovaných jako stop words. Jedná se o slova jako spojky, předložky nebo zájmena, tedy slova, která se nachází v každém dokumentu, často se opakují, jsou nezbytná pro větnou stavbu a zcela zbytečná pro klasifikaci. Rovněž libovolné číslovky lze eliminovat.

Sofistikovanějším přístupem je použití různých metrik, které určí, jak moc je slovo pro klasifikaci přínosné. Patří sem například výběr slov na základě Document Frequency, Information Gain nebo Chi-square.

Předchozí metody jsou založené na filtraci stávajících rysů, existuje ale metoda další, která místo filtrace vytváří zcela nové rysy a provádí tak trasnformaci z původního prostoru rysů do nového, který má typicky mnohem menší dimenzi. Děje se tak shlukováním původních slov na základě jejich sémantické blízkosti.

### 2.3.3 Proces klasifikace

Klasifikací se obecně rozumí proces, který určité objekty rozděluje do tříd. Tyto třídy jsou předem definovány. Aby klasifikátor dokázal vstupní objekty rozlišit, musí nejprve projít procesem učení. Rozlišujeme učení bez učitele, kdy si klasifikátor v datech sám najde závislosti, podle kterých bude rozhodovat, a učení s učitelem. V tomto případě poskytneme klasifikátoru množinu předem klasifikovaných dat, která slouží jako trénovací příklady. Klasifikátor si vytvoří pravidla, podle kterých nový objekt (s jistou tolerancí) přiřadí do nějaké třídy.

Po fázi trénování obvykle následuje fáze testování, jejímž cílem je ověřit úspěšnost naučeného klasifikátoru. Stejně jako u trénovacích dat, cílová třída každého příkladu je známa, ale slouží jen k vyhodnocení chybně klasifikovaných příkladů, klasifikátor ji ve svém rozhodování nesmí použít. Množina trénovacích dat a množina testovacích dat by měly být disjunktní. Obvykle máme k dispozici pouze jednu množinu dat a je třeba ji nejprve rozdělit.Tradiční přístup je náhodný výběr určitého procenta dat pro trénování a zbytek se použije pro testování. Jiným řešením je tzv. Cross-validation, cyklické trénování, kdy se původní množina rozdělí na *k* podmnožin, z nichž se vezme *k-1* podmnožin pro trénování a jedna podmnožina pro testování. V každém kroku se vymění jedna z podmnožin trénovacích za testovací, dokud se takto nevystřídají všechny podmnožiny.

### 2.3.4 Používané klasifikátory

Ke klasifikaci dokumentů existují dva přístupy: expertní systémy a strojové učení. V případě expertních systémů jsou klasifikační pravidla vytvořena manuálně skupinou expertů v dané oblasti. Z toho plyne jejich hlavní nevýhoda – vytvoření, stejně jako udržování takového systému, vyžaduje značné množství kvalifikovaných pracovníků. Příkladem expertního systému využívaného v oblasti klasifikace dokumentů je systém CONSTRUE používaný agenturou Reuters [1].

Naproti tomu v případě strojového učení vzniká klasifikátor automaticky – z kolekce předem klasifikovaných dokumentů, které slouží jako příklady pro učení. Expertní systémy nejsou dále v tomto textu uvažovány. Následující popisy principů klasifikačních metod čerpají z [3].

**Pravděpodobnostní klasifikátor**

Pro vstupní data (dokument) je určena pravděpodobnost, s jakou patří do jednotlivých tříd. Klasifikuje se do třídy, jejíž pravděpodobnost vyšla nejvyšší. Činnost klasifikátoru je založena na Bayesově vzorci, který má tvar:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \quad , \tag{2.3}$$

kde P(X) resp. P(Y) je pravděpodobnost jevu X resp. jevu Y, P(X | Y) je pravděpodobnost jevu X, jestliže víme, že nastal jev Y, analogicky P(Y | X).

Pro účely klasifikace dokumentů se za $Y$ dosazuje dokument $D$ tvořený termy $t_1 \ldots t_n$ a za $X$ kategorie $C$. Hledáme třídu $C$, pro kterou je maximální výraz $P(D \mid C)\, P(C)$, kde $P(D \mid C)$ znamená pravděpodobnost, že libovolně vybraný dokument třídy $C$ bude mít stejné hodnoty atributů (stejné termy) jako dokument $D$. $P(C)$ je pravděpodobnost, že náhodně vybraný dokument patří do kategorie $C$, což lze vyjádřit jako poměr počtu dokumentů patřících do kategorie $C$ a celkového počtu dokumentů všech kategorií.

$$P(D|C) = \prod_{i=1}^{n} P(t_i|C) \quad , \tag{2.4}$$

kde $P(t_i \mid C)$ je pravděpodobnost, že dokument, jehož $i$-tý term má hodnotu $t_i$, bude klasifikován do třídy $C$. Jelikož na termy lze pohlížet jako na diskrétní hodnoty, tato pravděpodobnost se dá vypočítat jako poměr počtu dokumentů patřících do kategorie $C$, jejichž $i$-tý term má hodnotu $t_i$ a počtu všech dokumentů patřících do kategorie $C$.

**Regrese**

Ačkoli se regrese řadí mezi metody predikce, lze ji rovněž použít pro klasifikaci dokumentů. Nejjednodušší je lineární regrese, která data aproximuje přímkou o rovnici:

$$Y = aX + b$$

Klíčovou otázkou je hledání parametrů $a$ a $b$, k čemuž se používá například metoda nejmenších čtverců (součet druhých mocnin odchylek skutečné a aproximované hodnoty y).

Lineární vícenásobná regrese se řídí rovnicí:

$$Y = a_0 + a_1 X_1 + a_2 X_2 + \ldots + a_n X_n ,$$

přičemž každý vzorek dat očekáváme ve tvaru $(x_1, x_2, \ldots, x_n, y)$, kde $y$ je kategorie a $x_i$ hodnoty jednotlivých atributů (termů). Cílem je nalézt koeficienty $a_0$ až $a_n$. Nejprve se z dat sestaví matice:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix} \quad \text{a} \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_m \end{pmatrix} \quad ,$$

dále hledaná matice:

$$A = \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_n \end{pmatrix}$$

, kterou lze vypočítat podle vzorce: $A=(X^T X)^{-1} X^T Y$.

Posledním případem je nelineární regrese, kdy má rovnice tvar:

$$Y = a_0 + a_1 X + a_2 X^2 + \ldots + a_n X^n$$

Toto lze dále řešit zavedením nových proměnných a substitucí:

$X_1 = X, X_2 = X^2, \ldots, X_n = X^n,$

což vede k rovnici pro lineární vícenásobnou regresi.

**Rocchio**

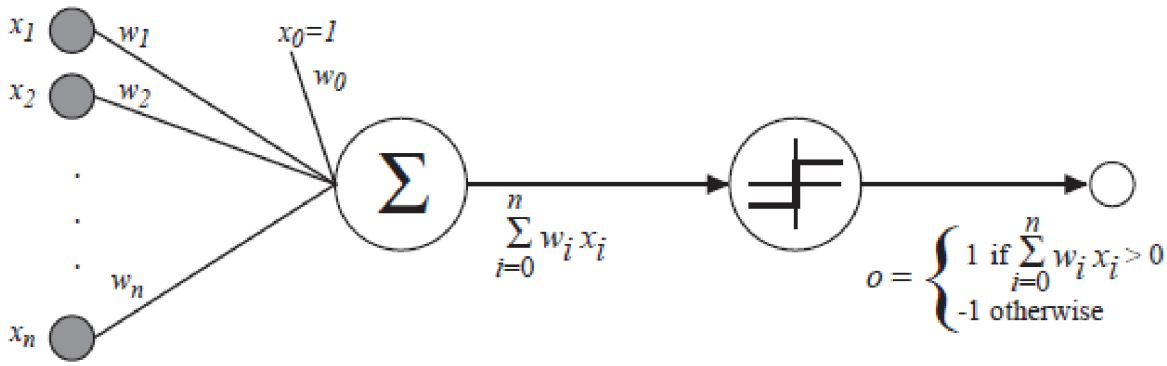
Tato metoda klasifikuje dokument podle výpočtu jeho vzdálenosti od referenčního vektoru každé kategorie. Referenční vektor se počítá na základě rysů dokumentů a počtu pozitivních a negativních dokumentů pro každou kategorii.

**Neuronové sítě**

Základem neuronové sítě je perceptron (obrázek 1), který je jakousi abstrakcí neuronu – základní buňky tvořící lidský mozek. Neuron se skládá z těla (soma), vstupů (dendrity) a výstupu (axon). Počet dendritů se pohybuje v řádu stovek až stovek tisíc, jedná se o výběžky dlouhé jen pár milimetrů, zatímco axon je pouze jeden a jeho délka může dosahovat až desítek centimetrů. Dendrity se napojují na axon jiného neuronu prostřednictvím synapse, což je rozhraní, které ovlivňuje přenos elektrických impulzů uvolňováním chemických látek.

Na vstupy ($x_1$ až $x_n$) perceptronu je přiveden vstupní vektor, z něhož se počítá vážená suma. Vstup $x_0$ je pevně stanoveným posunem. Vypočítaná suma potom vstupuje do výstupní funkce, na obrázku je znázorněna funkce, která přiřadí na výstup hodnotu +1, jestliže je suma kladná a -1, pokud je záporná. Velmi často se používá také sigmoidní funkce, která má tvar:

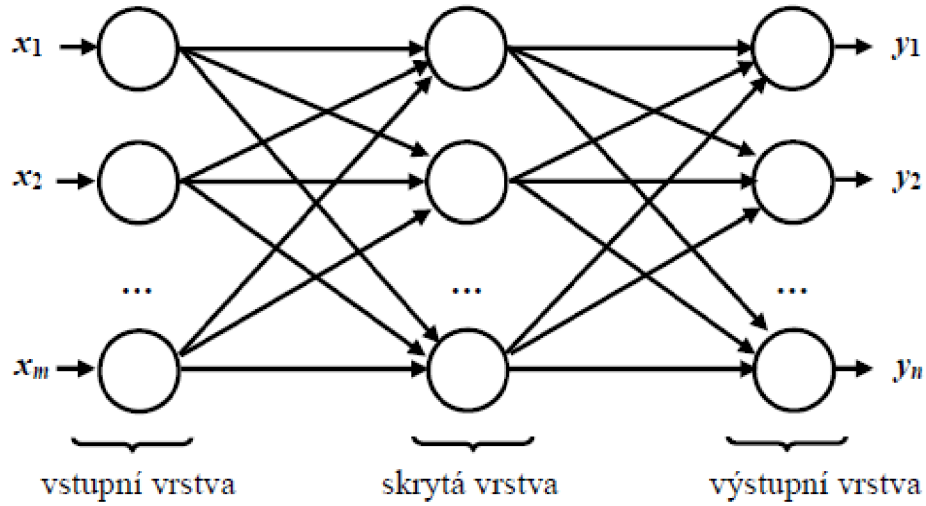
$$\sigma = \frac{1}{1 + e^{-x}} \quad . \tag{2.4}$$




*Obrázek 1: perceptron (převzato z [10]).*

Perceptrony se propojují do sítí různé topologie (dopředné sítě, rekurentní sítě, Kohonenovy mapy...), které mají zpravidla vstupní vrstvu, na kterou navazuje několik vrstev skrytých a výstupní vrstva. Existují rovněž neuronové sítě s proměnnou topologií.

Z hlediska učení patří neuronová síť mezi metody učení s učitelem (supervised learning), základem je postupná optimalizace vah na vstupech jednotlivých neuronů. Oblíbeným algoritmem trénování je algoritmus backpropagation, který – jak název napovídá – spočívá ve zpětné propagaci chyby klasifikace z výstupní vrstvy přes vrstvy skryté až po vrstvu vstupní.




*Obrázek 2: propojení neuronové sítě (převzato z [3]).*

Ke klasifikaci dokumentů stačí lineární klasifikátor, tvořený jediným perceptronem. Složitější klasifikátory vykazují pouze mírné zlepšení v přesnosti.

**KNN**

Klasifikátor k-nejbližšího sousedství si všímá podobnosti s již klasifikovanými dokumenty. Dokument je přiřazen do stejné třídy, kam patří majoritní část z jeho *k* nejbližších sousedů, tedy dokumentů, které jsou mu nejvíc podobné. Pro určení, který soused je nejbližší je nejprve nutné definovat metriku. Často využívanou metrikou je Euklidovská vzdálenost:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad . \tag{2.5}$$

Dále lze zmínit Mahanalobisovu metriku, metriku městského bloku nebo Minkowského metriku.

**Rozhodovací stromy**

Klasifikátor je grafem se stromovou strukturou. Vnitřní uzly představují test na hodnotu jednotlivých rysů a listové uzly reprezentují třídu, do které je dokument přiřazen. Strom lze snadno převést na klasifikační pravidla. Strom je vytvářen algoritmem, který v každém kroku vybírá atribut s největší rozhodovací schopností – takový atribut vlastně vytvoří nový rozhodovací uzel stromu.

Nejvhodnější atribut je vybírán na základě maximální hodnoty $Gain(A) = I(s_1,...,s_n) - E(A)$, kde $I(s_1,...,s_n)$ je očekávaná informace a $E(A)$ je entropie atributu $A$. Očekávanou informaci lze vypočítat podle vzorce:

$$I(s_{1j}, \ldots, s_{nj}) = -\sum_{i=1}^{n} p_{ij} \log_2(p_{ij}) \quad , \tag{2.6}$$

kde $p_{ij}$ je pravděpodobnost, že náhodně vybraný prvek z množiny $S_j$ bude klasifikován do $i$-té kategorie. Jestliže $a_1,...,a_n$ jsou všechny hodnoty atributu $A$, pak $S_j$ je množina všech trénovacích vzorků, jejichž atribut $A$ má hodnotu $a_j$. Dále $s_{ij}$ je počet vzorků z množiny $S_j$, které jsou klasifikovány do $i$-té třídy.

Entropii lze vypočítat podle vzorce:

$$E(A) = -\sum_{j=1}^{m} \frac{s_{1j} + \ldots + s_{nj}}{|S|} I(s_{1j}, \ldots, s_{nj}) \quad , \tag{2.7}$$

kde $S$ značí množinu všech trénovacích vzorků.

Během vytváření stromu vznikají i takové větve, které nejen že dělají strom zbytečně složitějším, ale často i zhoršují jeho klasifikační úspěšnost. Je proto žádoucí tyto větve odstranit, k čemuž se využívá dvou metod. Prepruning je metoda, kdy se tyto větve vůbec do stromu negenerují, zatímco u metody postpruning se nejprve vytvoří celý strom a následně se odstraní nevhodné větve. Obě metody mají svoje výhody i nevýhody a proto se v praxi kombinují.

**SVM (Support Vector Machine)**

Tato metoda si získává čím dál větší oblibu nejen v klasifikaci dokumentů, ale i v oblasti rozpoznávání vzorů v obrazech – například rozpoznávání obličejů. Protože tato metoda byla zvolena pro implementaci nástroje na klasifikaci dokumentů, jejímu vysvětlení se věnuje celá kapitola 3.

### 2.3.5 Aplikace

**Třídění dokumentů**

Jedná se o typický problém třídění dokumentů do několika kategorií, toto třídění většinou probíhá průběžně po jednotlivých dokumentech. Zvláštním případem je filtrace textu, kdy existují pouze dvě kategorie – relevantní dokumenty a dokumenty, které chceme odfiltrovat. Jako příklad lze uvést filtrování spamu v e-mailech.

Ačkoli většinou mají chyby v přesnosti a odpovědi stejný význam, u filtrování textu tomu tak není. Právě na zmíněném příkladu filtrace spamu je patrný rozdíl, protože relevantní e-mail, který je vyhodnocen jako spam, způsobí více škody, než jeden spam, který pronikne mezi e-maily.

**Indexování**

Indexování dokumentů nachází uplatnění v IR systémech. Tyto systémy vyhledávají na základě klíčových slov přiřazených ke každému dokumentu v prohledávané kolekci. Právě tato množina klíčových slov (označovaná jako slovník) může být chápána jako množina kategorií, do kterých chceme dokumenty klasifikovat. Tento proces může probíhat plně automaticky nebo poloautomaticky – uživateli se předloží seznam pravděpodobných klíčových slov a je na něm, která z nich vybere.

**Hierarchická klasifikace webových stránek**

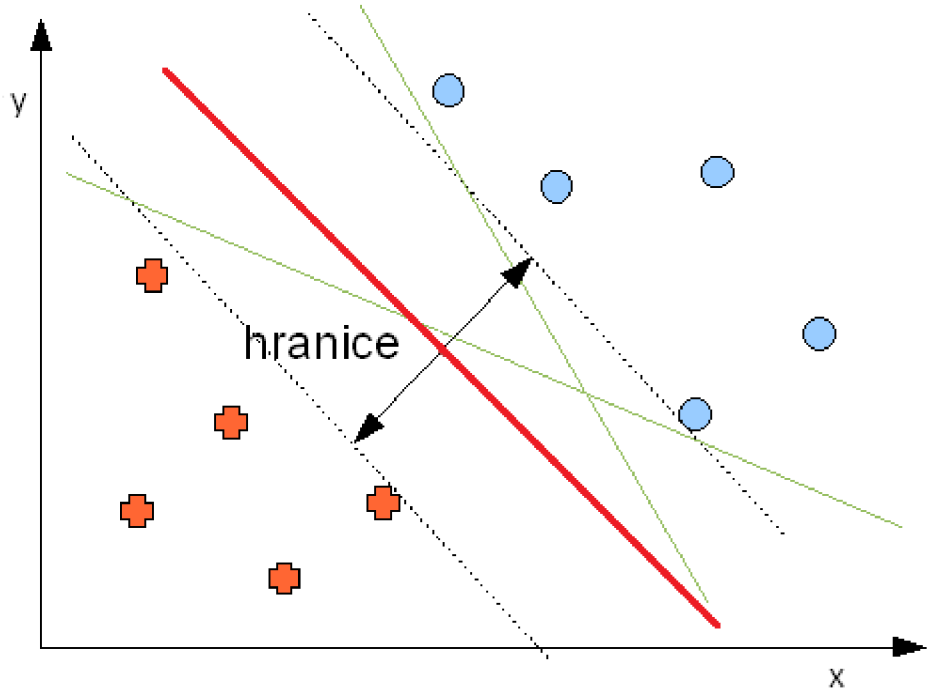
Řada internetových vyhledávačů používá hierarchické katalogy webových stránek. I zde nachází klasifikace textu uplatnění, avšak je třeba se vypořádat s narůstající velikostí kategorií, aby nedocházelo k poklesu přehlednosti. To se zpravidla řeší omezením velikosti každé kategorie a při dosažení této hranice jejím rozdělením na několik podkategorií. Systém, který klasifikaci provádí musí zvládat přídávání nových a mazání starých kategorií.

# 3 Klasifikace metodou SVM

Autorem metody SVM je Vladimir Vapnik, který ji prezentoval roku 1979 [6]. Metoda je založena na hledání dělící nadroviny v prostoru rysů mezi daty různých tříd. To samo o sobě není nic výjimečného, jedinečnost této metody spočívá v tom, že hledá vždy optimální dělící nadrovinu, jinak řečeno, snaží se o maximalizaci šířky hranice, která data rozděluje.

## 3.1 Lineární SVM

Metoda SVM je ve své původní podobě lineární. Dokáže od sebe odlišit dvě třídy objektů (nazvěme je pozitivní a negativní) tak, že je hranice mezi nimi maximální. Na obrázku 3 jsou vyznačena data dvou tříd v dvourozměrném prostoru. Zelené přímky představují libovolné dělící nadroviny, červená je optimální dělící nadrovina, čárkované přímky znázorňují hranice dat obou tříd.




*Obrázek 3: lineární SVM - optimální dělící nadrovina*

### 3.1.1 Lineárně separovatelný problém

Rovnice dělící nadroviny má tvar:

$$0 = \vec{w} \cdot \vec{x} - b \tag{3.1}$$

kde $\vec{w}$ je normálový vektor nadroviny, $\vec{x}$ je vstupní vektor rysů a $b$ je posunutí.

Vzdálenost mezi nadrovinou a vzorky dat z pozitivní i negativní třídy je $\frac{1}{\|\vec{w}\|}$, mezi daty obou tříd tedy $\frac{1}{\|\vec{w}\|} + \frac{1}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$. Tuto vzdálenost chceme maximalizovat, což se dá zapsat jako optimalizační problém:

$$min \frac{1}{2}\|\vec{w}\|^2 \quad \text{vzhledem k} \quad y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \forall i \quad , \tag{3.2}$$

kde $\vec{x}_i$ je $i$-tý vstupní vektor z trénovací množiny, $y_i \in \{-1, +1\}$ je výstup klasifikátoru pro $i$-tý vstupní vektor. Jedná se o hledání vyhovujících parametrů $\vec{w}$ a $b$, aby byly splněny výše zmíněné požadavky. To představuje kvadratický optimalizační problém, který lze řešit použitím lagrangianu následovně:

$$L_P = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i \cdot y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 + \sum_{i=1}^{n} \alpha_i \quad , \tag{3.3}$$

kde $\alpha_i$ je Lagrangeův koeficient.

S podmínkami:

$$\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \tag{3.4}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.5}$$

Rovnici (3.4) můžeme dosadit do (3.3) a dostáváme duální formu:

$$L_D = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j + \sum_{i=1}^{n} \alpha_i \tag{3.6}$$

vzhledem k podmínkám:

$$\alpha_i \geq 0, \forall i \quad \text{a} \tag{3.7}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.8}$$

V této podobě už není lagrangian závislý na normálovém vektoru $\vec{w}$ a posunutí $b$, ale pouze na koeficientech $\alpha$, pro které platí podmínky (3.7) a (3.8).

### 3.1.2 Lineárně neseparovatelný problém

V případě, že problém není lineárně separovatelný, musíme zavést chybovou míru $\xi$, která udává, jak daleko se trénovaný vzorek dostane mimo hranice vytyčené dělící nadrovinou. Je-li $\xi = 0$, pak je vzorek správně klasifikován, je-li $0 < \xi < 1$, pak je sice klasifikován správně, ale nachází se uvnitř hranice kolem dělící nadroviny. Konečně, je-li $\xi \geq 1$, vzorek dat byl klasifikován chybně. Potom $\sum \xi_i$ znamená maximální počet chybně klasifikovaných vzorků. Zároveň se stanoví konstanta $C$, která představuje penalizaci za každý chybně klasifikovaný vzorek trénovacích dat.

Toto rozšíření si neporadí s problémem, který ze své podstaty není lineárně separovatelný, ale vyřeší šum v datech, který by znemožňoval nalezení nadroviny pro čistě lineárně oddělitelná data. Minimalizační úloha se mění do podoby:

$$min \frac{1}{2} \|\vec{w}\|^2 + C \cdot \sum_i \xi_i \tag{3.9}$$

Duální forma lagrangianu zůstává stejná (3.6) a mění se pouze podmínky:

$$0 \leq \alpha_i \leq C, \forall i \quad \text{a} \tag{3.10}$$

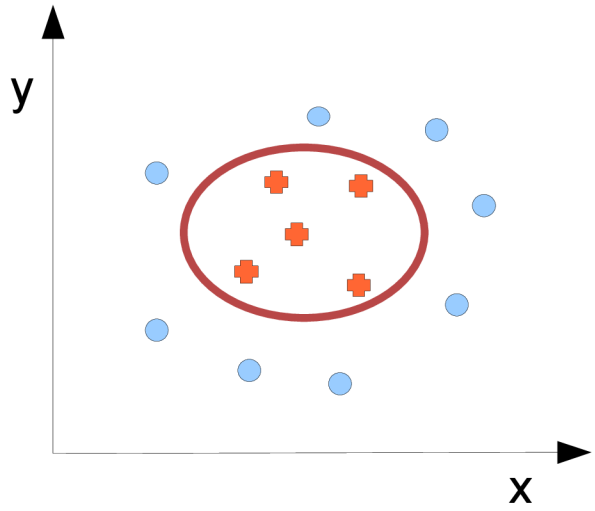
$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3.11}$$

Klasifikace vstupního vektoru $\vec{x}$ probíhá výpočtem funkce:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} - b \tag{3.12}$$

Je-li její výsledek kladný, patří vzorek do pozitivní třídy, jinak patří do třídy negativní.

## 3.2 Nelineární SVM




*Obrázek 4: příklad nelineárního problému*

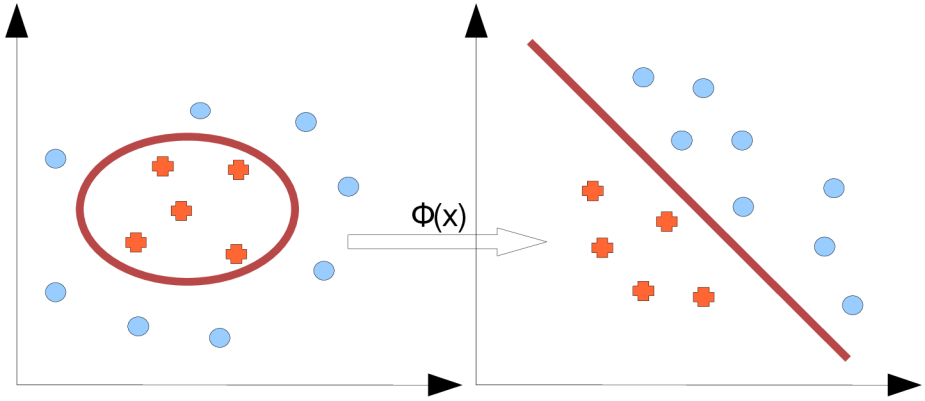
Metoda SVM se později dočkala dalšího rozšíření, které umožňuje řešit nelineární problémy, se kterými by si klasifikátor představený v předchozích podkapitolách neporadil (obrázek 4). Princip spočívá v mapování vstupních dat do jiného prostoru za použití jádrových funkcí.

### 3.2.1 Mapování do nového prostoru

Pokud data nejsou lineárně separovatelná, lze je namapovat do nového prostoru, zpravidla vyšší dimenze, kde už lineární separace možná je. Toto mapování lze zapsat:

$$\Phi : R^d \rightarrow H \quad , \tag{3.13}$$

kde $H$ je nový euklidovský prostor [6]. Nevýhodou mapování je skutečnost, že s tím, jak vzroste dimenzionalita, vzroste i výpočetní náročnost skalárních součinů vektorů.




*Obrázek 5: mapování mezi dvěma prostory*

### 3.2.2 Jádrové funkce

Jádrové funkce využívají skutečnosti, že se vstupní data vyskytují v lagrangianu pouze jako skalární součin dvojice různých vektorů. Jestliže provedeme mapování do jiného prostoru podle předchozí kapitoly, dostáváme lagrangian:

$$L_D = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j)+\sum_{i=1}^{n}\alpha_i \tag{3.14}$$

Pro odstranění zmíněné nevýhody mapování potřebujeme, aby existovala funkce, pro kterou platí:

$$K(\vec{x}_i,\vec{x}_j) = \Phi(\vec{x}_i)\cdot\Phi(\vec{x}_j) \tag{3.15}$$

Nyní můžeme skalární součin vektorů mapovaných do nového prostoru nahradit touto jádrovou funkcí, čímž dostává lagrangian tvar:

$$L_D = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\vec{x}_i,\vec{x}_j)+\sum_{i=1}^{n}\alpha_i \tag{3.16}$$

Podmínky se nijak neliší a zůstávají v podobě rovnic (3.10) a (3.11) – použití mapování a jádrových funkcí neznamená, že rozšíření představené v kapitole 3.1.2 je už zbytečné. I po převedení do nového prostoru se můžou vyskytnout vzorky dat, které znemožní čistě lineární separaci.

**Mezi často používané jádrové funkce patří:**

- Polynomiální jádrová funkce $K(\vec{x}_i,\vec{x}_j) = (\vec{x}_i\cdot\vec{x}_j + c)^n$
- RBF (Radial Basis Function) jádrová funkce $K(\vec{x}_i,\vec{x}_j) = \exp\left(\frac{-\|\vec{x}_i-\vec{x}_j\|^2}{2\sigma^2}\right)$
- Sigmoidová jádrová funkce $K(\vec{x}_i,\vec{x}_j) = \tanh(\kappa\,\vec{x}_i\cdot\vec{x}_j-\delta)$

### 3.2.3 Klasifikace nelineárního SVM

Klasifikace nelineárního SVM se od lineárního liší tím, že v původním prostoru rysů nemusí existovat vektor, který by po mapování odpovídal vektoru $\vec{w}$ v novém vysoce dimenzionálním prostoru. Proto nelze využít rovnice (3.12) a hodnota klasifikační funkce se vypočítá [6]:

$$f(\vec{x}) = \sum_{i=1}^{N_s}\alpha_i y_i K(\vec{s}_i,\vec{x})-b \quad , \tag{3.17}$$

kde $\vec{s}_i$ jsou podpůrné vektory – jsou to ty vektory trénovacích dat, které leží na hranicích dělící nadroviny. $N_s$ je počet podpůrných vektorů.

## 3.3 Metody učení

Učení SVM klasifikátoru znamená vyřešit kvadratický optimalizační problém (viz rovnice 3.16). K tomu existuje celá řada nástrojů a knihoven, avšak většinou používají řešení, které vyžaduje matici o velikosti druhé mocniny počtu trénovacích dat. Takové paměťové nároky jsou v případě tisíců trénovacích příkladů nepřijatelné. Existují proto metody, které se snaží rozdělit kvadratický optimalizační problém na menší podproblémy a ty řešit samostatně.

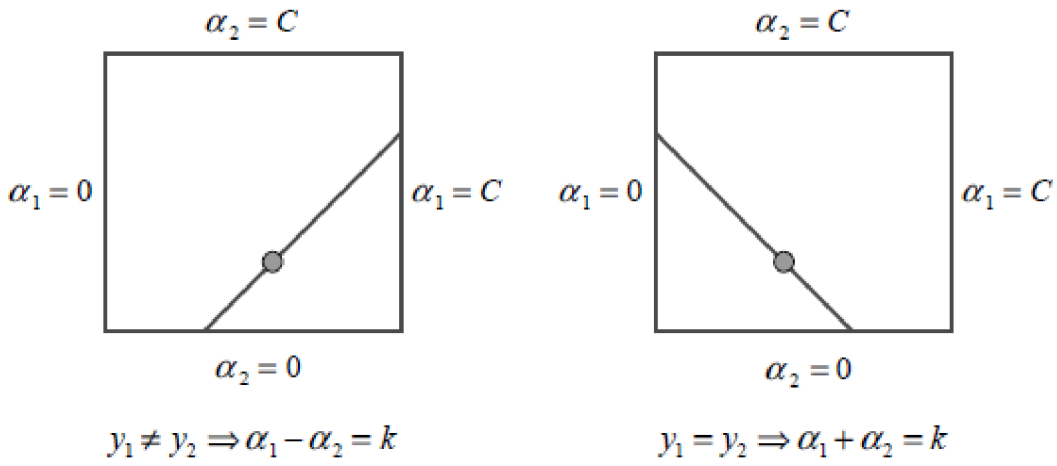
### 3.3.1 Dřívější metody

Sám Vapnik navrhl metodu „chunking“, která je založena na poznatku, že z výše popsané matice lze odebrat řádky a sloupce, které odpovídají nulovým Lagrangeovým koeficientům. Původní kvadratický optimalizační problém se rozpadá na řešení skupiny menších podproblémů. V každém kroku algoritmu se řeší podproblém skládající se z nenulových Lagrangeových koeficientů z minulého kroku a z $n$ příkladů, které nejvíc porušují KKT podmínky. V posledním kroku se vyřeší původní kvadratický problém. Někdy ovšem ani toto dělení na podproblémy nemusí stačit (u velmi rozsáhlých a vysoce dimenzionálních dat).

Osunův algoritmus opět dělí původní kvadratický problém na podproblémy. Pracuje s maticí konstantní velikosti, ze které se v každém kroku jistý počet prvků odebere a stejný počet prvků se do ní zase přidá.

### 3.3.2 SMO

Algoritmus Sequential Minimal Optimization (SMO) nepotřebuje žádnou matici, řeší kvadratický problém po nejmenších podproblémech – optimalizuje v každém kroku právě dva Lagrangeovy koeficienty. Předchozí algoritmy řeší problém pomocí numerických metod, které je velmi obtížné naprogramovat tak, aby nevznikaly potíže s přesností výpočtu [5]. SMO používá analytické řešení – což je možné právě díky tomu, že v každém kroku optimalizuje pouze dva koeficienty. Díky tomu je algoritmus snadnější na implementaci a rychlejší v trénování klasifikátoru.




*Obrázek 6: podmínky (3.10) a (3.11) omezující koeficienty (převzato z [5])*

Podmínky kladené na celý optimalizační problém platí i pro dvojice koeficientů. Podmínka nerovnosti (3.10) vymezuje koeficienty do čtverce, podmínka rovnosti (3.11) způsobuje, že leží na diagonále. Algoritmus má dvě části – optimalizaci dvojice koeficientů a heuristiku pro výběr této dvojice.

**Heuristika pro výběr koeficientů**

Nejprve je třeba zmínit se o Karush-Kuhn-Tuckerových (KKT) podmínkách, které představují při řešení kvadratického optimalizačního problému nutnou i dostačující podmínku nalezení optimálního bodu. Kvadratický problém je vyřešen tehdy a jen tehdy když:

$$\begin{aligned} \alpha_i = 0 &\Leftrightarrow y_i u_i \geq 1, \\ 0 < \alpha_i < C &\Leftrightarrow y_i u_i = 1, \\ \alpha_i = C &\Leftrightarrow y_i u_i \leq 1 \end{aligned} \tag{3.18}$$

kde $u_i$ je výstup klasifikátoru pro $i$-tý vstupní vektor.

Pro oba koeficienty se používá odlišná heuristika. Vnější smyčka algoritmu nejprve proběhne nad celou trénovací množinou a zjistí, které její prvky porušují KKT podmínky a tím jsou vhodné pro optimalizaci. Dále proběhne nad těmi prvky, jejichž Lagrangeovy koeficienty neleží na hranici, tj. nejsou nulové ani nemají hodnotu $C$. Opět se vyhodnocuje porušení KKT podmínek. Jakmile nejsou nalezeny žádné nehraniční prvky nesplňující KKT podmínky, smyčka opět kontroluje celou trénovací množinu. První koeficient se tedy přednostně vybírá z nehraničních prvků množiny. Ve chvíli, kdy všechny prvky splňují KKT podmínky, algoritmus končí.

Druhý koeficient se k prvnímu vybere tak, aby se maximalizoval optimalizační krok. Vybírá se na základě klasifikační chyby, která se pro každý koeficient ukládá.

**Optimalizace dvojice koeficientů**

Nejprve se stanoví minimální a maximální možná hodnota koeficientu $\alpha_2$, výpočet se liší podle toho, zda se třídy obou koeficientů shodují:

- y1 ≠ y2, potom

$$L = max(0, \alpha_2 - \alpha_1), \; H = min(C, C + \alpha_2 - \alpha_1) \tag{3.19}$$

- y1 = y2, potom

$$L = max(0, \alpha_2 + \alpha_1 - C), \; H = min(C, \alpha_1 + \alpha_2) \tag{3.20}$$

Dále se vypočítá nová hodnota koeficientu $\alpha_2$:

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2)} \;, \tag{3.21}$$

kde $E_i = y_i - u_i$ je chyba $i$-tého trénovacího příkladu. Nyní se nová hodnota koeficientu musí ořezat, aby ležela na úsečce (obrázek 6).

$$\alpha_2^{new,clipped} = \begin{cases} H \Leftrightarrow \alpha_2^{new} \geq H; \\ \alpha_2^{new} \Leftrightarrow L < \alpha_2^{new} < H; \\ L \Leftrightarrow \alpha_2^{new} \leq L. \end{cases} \tag{3.22}$$

Nyní může být spočítána i hodnota koeficientu $\alpha_1$:

$$\alpha_1^{new} = \alpha_1 + y_1 y_2 (\alpha_2 - \alpha_2^{new,clipped}). \tag{3.23}$$

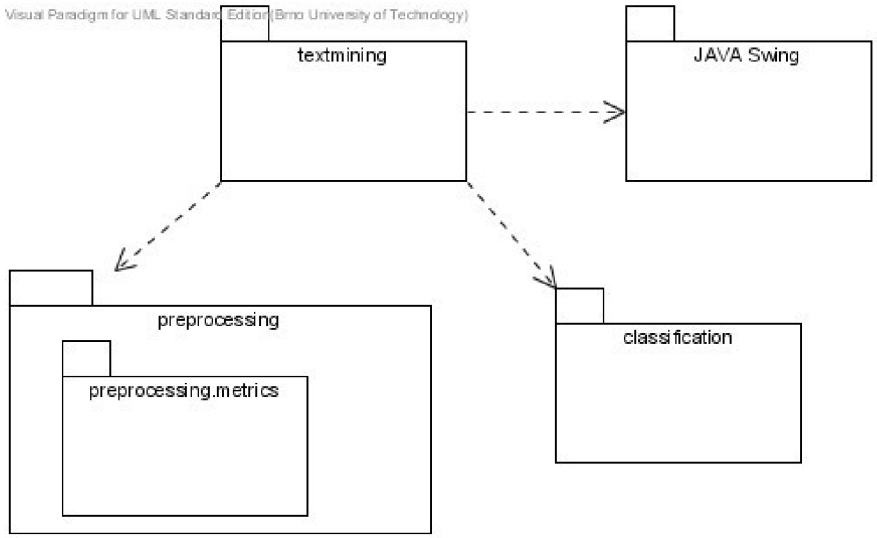
# 4 Návrh aplikace

Následující podkapitoly obsahují návrh aplikace pro klasifikaci dokumentů. Uveden je diagram balíčku (kapitola 4.1), který dává přehled o struktuře aplikace na nejvyšší úrovni, dále diagramy tříd (kapitola 4.2) pro každý balíček zvlášť a diagramy sekvence typických akcí programu (kapitola 4.3). V návrhu je kladen důraz na budoucí rozšiřitelnost aplikace, což je řešeno mimo jiné oddělením důležitých komponent stabilním rozhraním.

Pro implementaci byl zvolen programovací jazyk Java hned z několika důvodů. Jedná se o objektově orientovaný jazyk poskytující dostatečnou míru abstrakce a komfortu. Dále díky běhovému prostředí poskytuje nezávislost na platformě. Další výhodou je existence implementace Porter stemmeru v tomto jazyce (viz [11]) a podpora některých návrhových vzorů. Pro tvorbu uživatelského prostředí jsou použity komponenty knihovny Swing.

## 4.1 Diagram balíčků

Aplikace se skládá ze dvou významných částí – předzpracovánía klasifikace, což je zohledněno v rozdělení na balíčky. Tyto balíčky potom využívá vrstva aplikační logiky v návaznosti na GUI – balíček textmining.



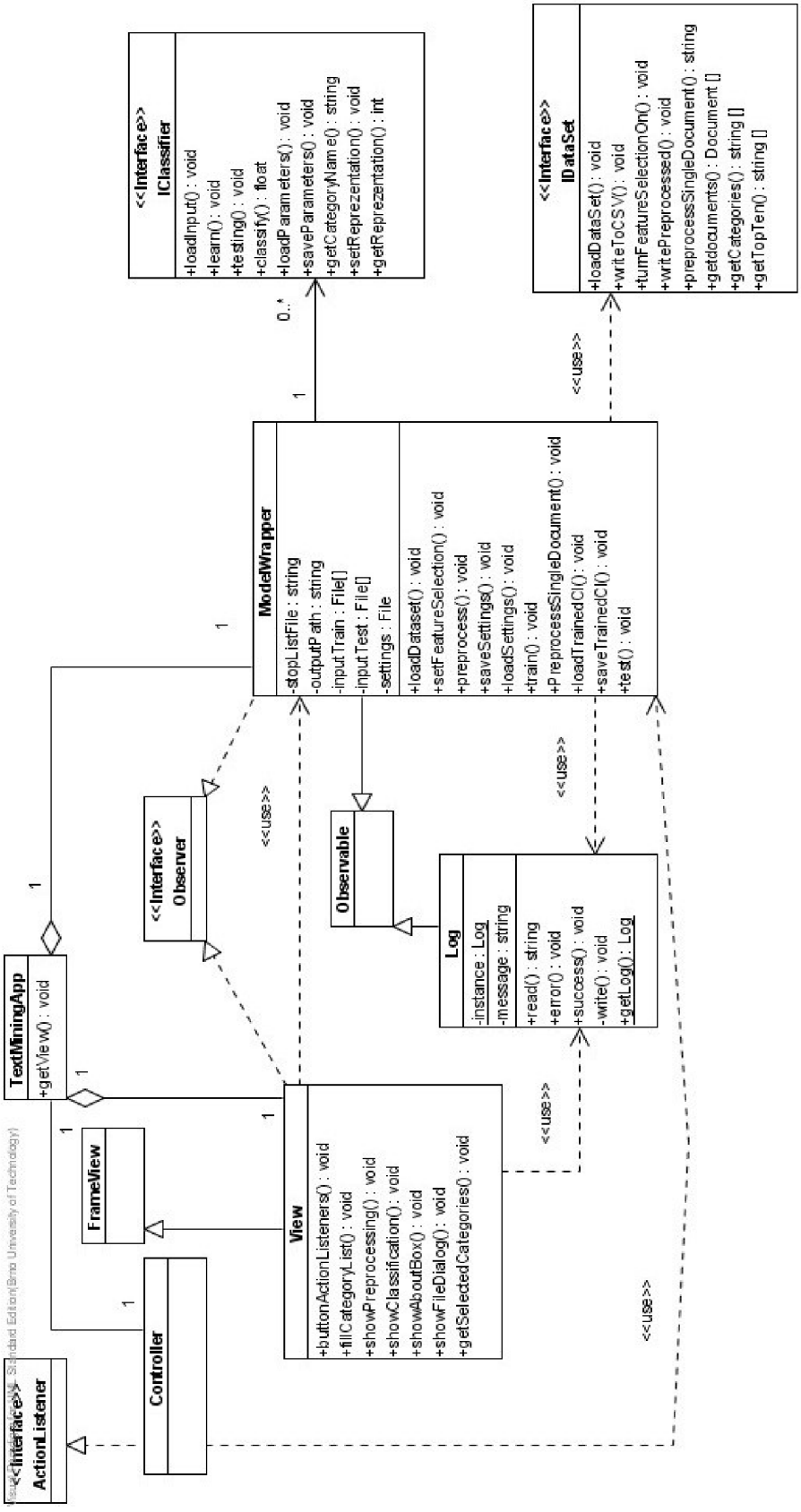

*Obrázek 7: diagram balíčků*

## 4.2 Diagramy tříd

### 4.2.1 GUI a aplikační logika

Aplikace je koncipována do architektonického vzoru MVC. Třída `View` zajišťuje zobrazení, je vlastníkem veškerých použitých ovládacích prvků knihovny Swing a několika dialogů (např. dialog pro otevření souboru). Třída `ModelWrapper` zastřešuje moduly předzpracování a klasifikace připojené přes rozhraní `IDataSet` a `IClassifier`. Z tohoto pohledu se jedná o využití návrhového vzoru fasáda, dále v sobě tato třída nese nastavení aplikace a stará se o jejich načítání a ukládání. Třída `Controller` řídí zobrazování a provádění změnv modelu, reaguje na události od uživatele. Jakmile uživatel stiskne tlačítko nebo vybere položku menu, tato událost se přepošle objektu třídy `Controller`, který rozhodne o akci, která bude provedena. Pro zajištění komunikace je použit návrhový vzor Observer, který má v jazyce Java oporu v třídě `Observable` a rozhraní `Observer`. Vlastníkem tříd `ModelWrapper`, `View` a `Controller` je třída `TextMiningApp`.

Funkci hlášení chybových stavů i úspěšně dokončených akcí pro lepší interakci s uživatelem na sebe bere třída `Log`, která je navržena jako Singleton. Ta je dostupná z libovolného místa aplikace a rovněž je spjata vzorem Observer s třídou `ModelWrapper`. Oznámení o chybě nebo úspěšně provedené akci potom probíhá následovně: zpráva je zapsána do logu, ten ji předá objektu třídy `ModelWrapper`, který o změně stavu vyrozumí objekt třídy `View`. Ten si následně zprávu vyzvedne a zobrazí ji do textového pole.



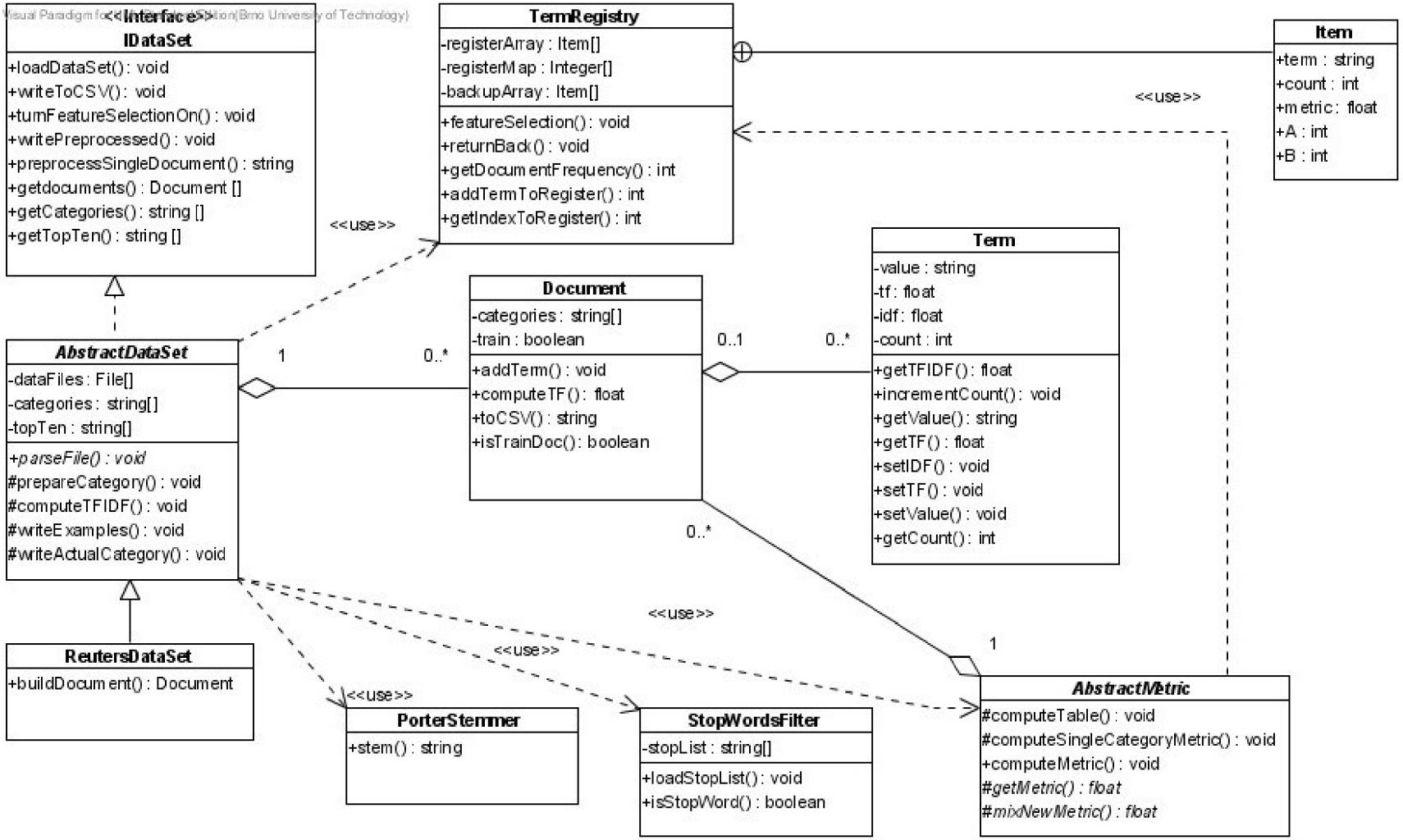

*Obrázek 8: diagram tříd aplikační logiky*

### 4.2.2 Balíček preprocessing

Balíček předzpracování obsahuje reprezentaci dokumentu pomocí termů. Jednotlivé termy jsou reprezentovány třídou `Term`. Vlastnostmi této třídy jsou hodnoty TF a IDF a dále index do centrálního registru termů. Dále obsahuje pouze vhodné přístupové metody. Objekty této třídy se sdružují do kolekce, která spolu s kolekcí kategorií fakticky tvoří jednotlivé dokumenty. Kategorie jsou reprezentovány pouze jako textový řetězec, jelikož není třeba udržovat o nich další údaje. Další informací, kterou v sobě dokument nese, je příznak, zda je určen pro testování nebo trénování (aplikace se snaží respektovat původní rozdělení dle autora datasetu). Zbytek třídy `Document` tvoří metody pro přidání nového termu, získání textové reprezentace dokumentu a výpočet a nastavení hodnoty TF všech obsažených termů.

Třída `AbstractDataSet` implementuje většinu metod rozhraní `IDataSet`, ale některé činnosti deleguje na třídu `ReutersDataSet` – parsování jednotlivých souborů a sestavení nového objektu třídy `Document`. Třída `AbstractDataSet` v sobě nese kolekci dokumentů a je zodpovědná za výpočet hodnot TFIDF (je-li tato reprezentace zvolena) a přípravu dokumentů pro klasifikaci. Přípravou je míněno vytvoření trénovacích a testovacích souborů, což v sobě obnáší rozdělení dokumentů podle vybraných kategorií a náhodném přiřazení pozitivních a negativních trénovacích resp. testovacích příkladů. Rovněž obsahuje metody, které převedou dokumenty do zvolené reprezentace (binární, TF, TFIDF) a umožní jejich výpis v řídkém formátu.

Ve fázi načítání datasetu třída `AbstractDataSet` používá třídy `PorterStemmer` a `StopWordsFilter`. Třída `TermRegistry` udržuje souhrnné informace (je centrálním registrem termů) pro každý term vyskytující se v datasetu (vnitřní třída `Item`) a pro zbytek aplikace v podstatě vymezuje prostor rysů. Ten se může měnit v rámci selekce rysů, na které se podílí některá z metrik.

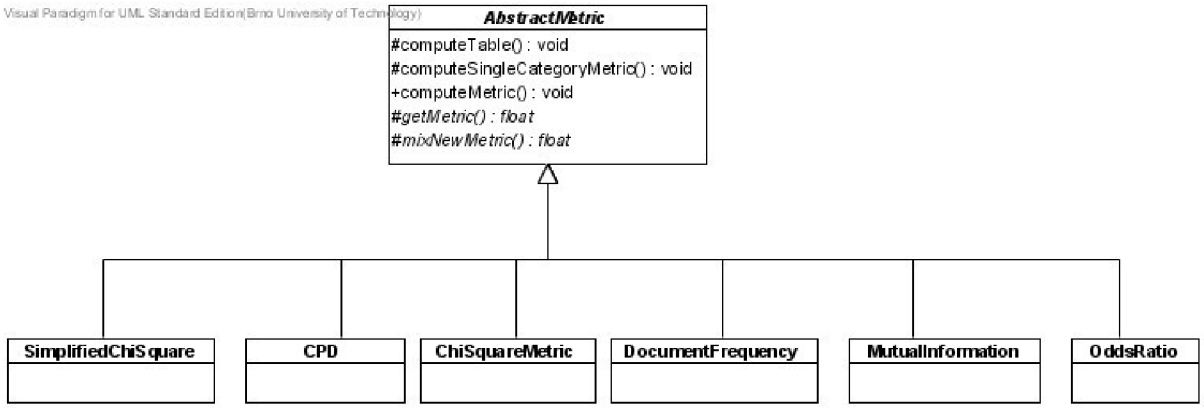



*Obrázek 9: diagram tříd předzpracování*

### 4.2.3 Balíček preprocessing.metric

Třída `AbstractMetric` reprezentuje metriku, podle které se provádí seřazení prostoru rysů a jejich následná selekce. Potomci třídy `AbstractMetric` jsou potom konkrétními metrikami používanými pro selekci rysů při dolování v textu nebo při dolování v datech obecně. Popis jednotlivých metrik lze nalézt v kapitole 5.2.2 a podrobněji v [8].

Navenek je viditelná pouze metoda `computeMetric()`, v rámci níž je volána metoda `computeSingleCategoryMetric()`. Ta si dále volá metodu `computeTable()` sloužící k výpočtu tabulky, ze které všechny metriky vycházejí, a metody `getMetric()` a `mixNewMetric()`. Poslední dvě zmíněné metody jsou zároveň jedinou částí výpočtu, kterou se od sebe jednotlivé metriky liší.



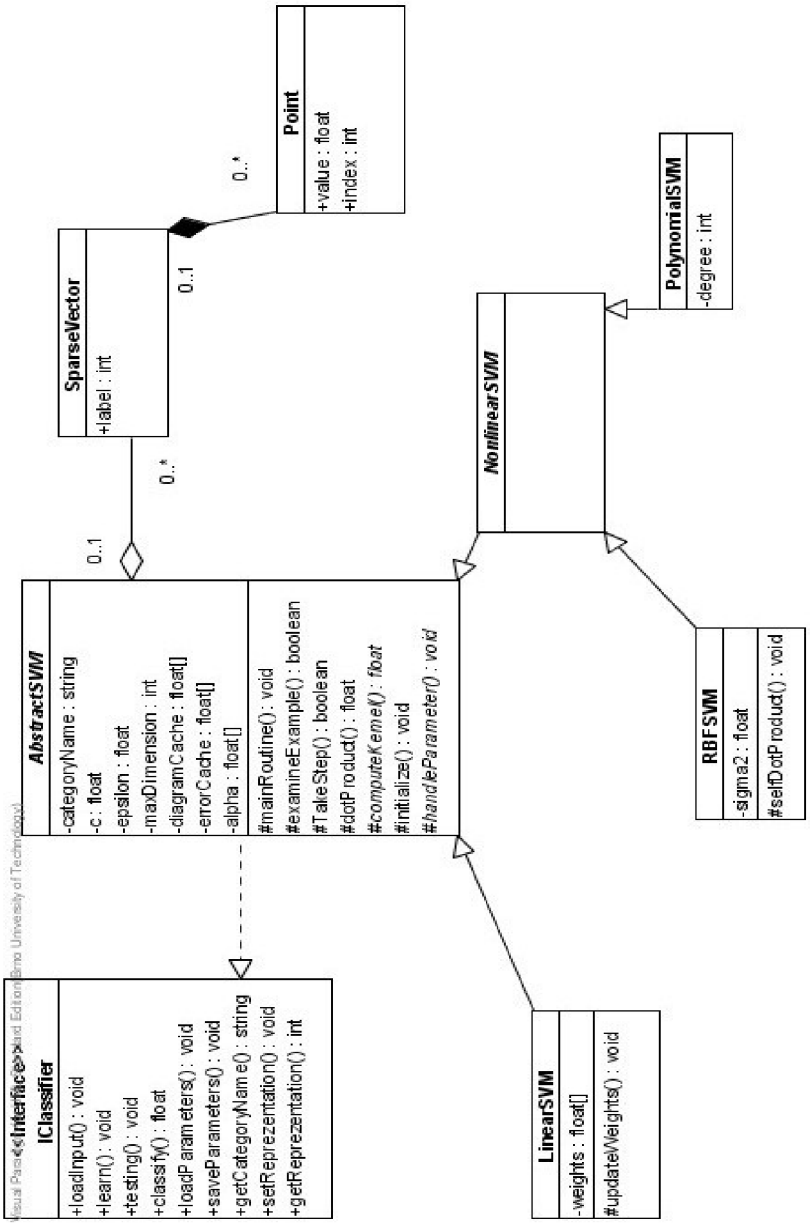

*Obrázek 10: diagram tříd metrik pro selekci rysů*

### 4.2.4 Balíček classification

Balíček klasifikace obsahuje implementaci klasifikátoru SVM, tedy přes rozhraní `IClassifier` poskytuje navenek metody pro trénování a testování klasifikátoru a klasifikaci jednoho dokumentu. Dalšími podporovanými operacemi je ukládání a načítání již naučeného klasifikátoru. Díky struktuře tříd a rozhraní není problém do aplikace začlenit další klasifikátory. Vstupní parametry, které ovlivňují průběh trénování, jsou předávány přes konstruktor.

Třída `AbstractSVM` v sobě nese implementaci algoritmu trénování SMO, tedy metody pro výběr vhodných Lagrangových koeficientů a jejich následnou optimalizaci. Je rovněž nositelem vstupních dat – kterými je trénovací resp. testovací množina dokumentů. Tato data jsou načítána ze souborů a ukládána do kolekce řídkých vektorů, které zastupuje třída `SparseVector`. Ta je tvořena kolekcí objektů třídy `Point`, která obsahuje vždy jednu hodnotu a její index.

Potomky třídy `AbstractSVM` jsou třídy `LinearSVM` a `NonlinearSVM`, `LinearSVM` obsahuje pole vah a metodu pro jejich aktualizaci během trénování. Dále implementuje metodu `classify()`, protože výpočet výsledku klasifikace se u lineárního SVM liší, a metodu `computeKernel()`, která vrací hodnotu jádrové funkce, jež je v tomto případě pouhým skalárním součinem vektorů. `NonlinearSVM` je abstraktní třídou, která sdružuje společné metody pro SVM s RBF a polynomiální jádrovou funkcí. Tyto klasifikátory reprezentují třídy `RBFSVM` a `PolynomialSVM`, které opět implementují metodu `computeKernel()`, jejíž výpočet se řídí vzorci uvedenýmiv kapitole 3.2.2.




*Obrázek 11: diagram tříd klasifikátoru*

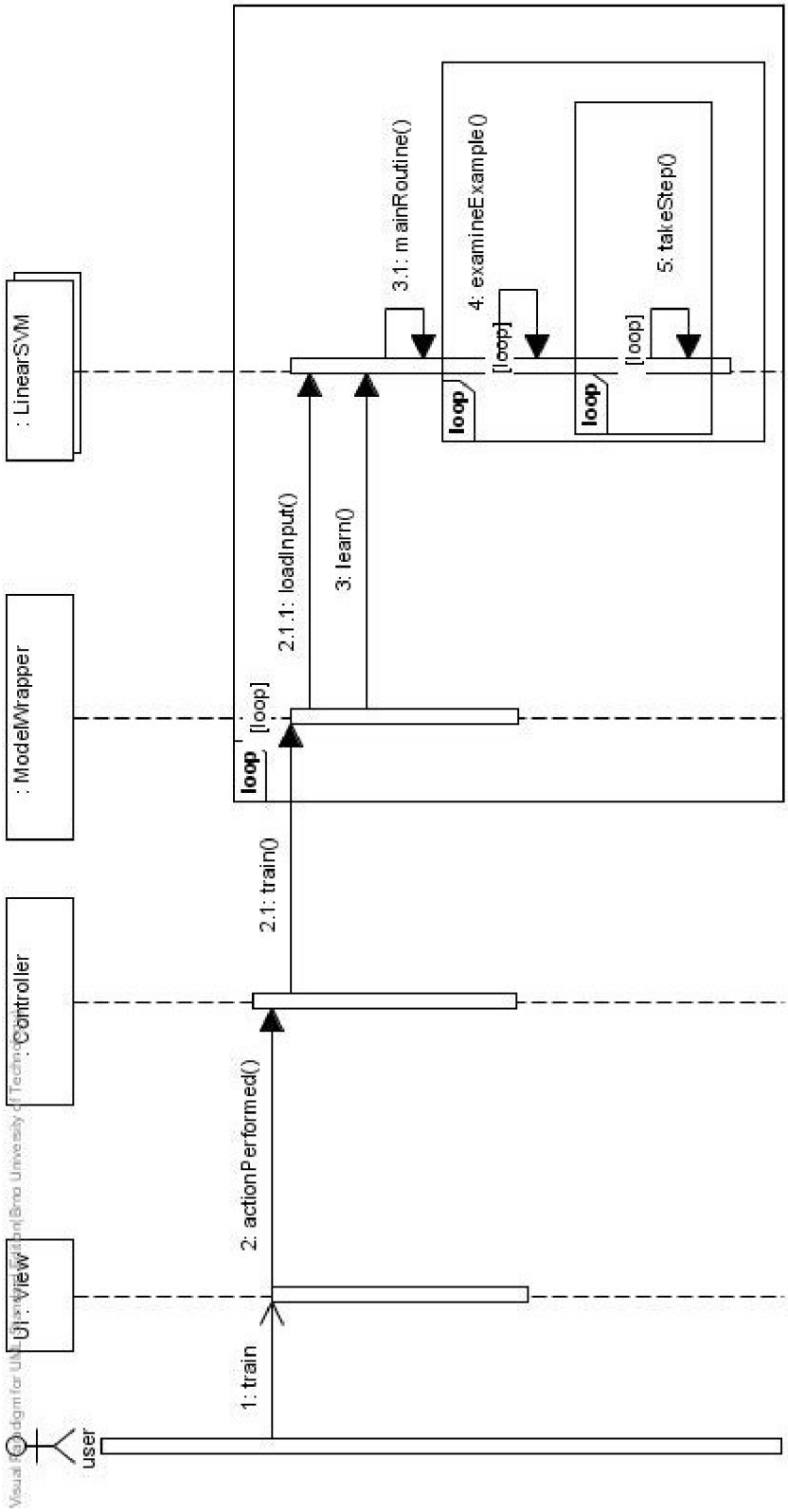
## 4.3 Diagramy Sekvence

Následující diagramy sekvence spolu s jejich popisem podávají přesnější představu o fungování implementované aplikace. Obsah této podkapitoly není na sto procent shodný s tím, jak je aplikace reálně implementována, jejím účelem je znázornit a vysvětlit vybrané operace.

### 4.3.1 Trénování klasifikátoru

Třída `ModelWrapper` je vlastníkem třídy implemetující rozhraní `IClassifier`, přesněji řečeno obsahuje pole instancí takové třídy, protože je umožněno trénování několika kategorií v dávce – každý klasifikátor se učí jednu kategorii. Po vytvoření instancí klasifikátorů (a předání parametrů přes konstruktory) se pro každou instanci provede trénování na souboru s trénovacími daty. Klasifikátor si uloží aktuální kategorii, kterou se bude učit (lze ji zjistit ze jména vstupního souboru nebo z jeho prvního řádku) a následně se spustí proces učení metodou `learn()`.

Ten se skládá z inicializace, kdy je vytvořena paměť chyb a paměť hodnot jádrové funkce a dále z volání metody `mainRoutine()`. Tato metoda v cyklu prochází trénovací množinou a vybírá první vhodný příklad k optimalizaci. Index nalezeného příkladu předá metodě `examineExample()`, která se pomocí heuristiky popsané v kapitole 3.3.2 snaží nalézt druhý co nevhodnější příklad. Indexy těchto dvou vybraných příkladů jsou předány metodě `TakeStep()`, která provádí samotnou optimalizaci sestávající z výpočtu nových hodnot Lagrangeových koeficientů, aktualizace posunutí, paměti chyb a v případě lineárního SVM i aktualizace vah.



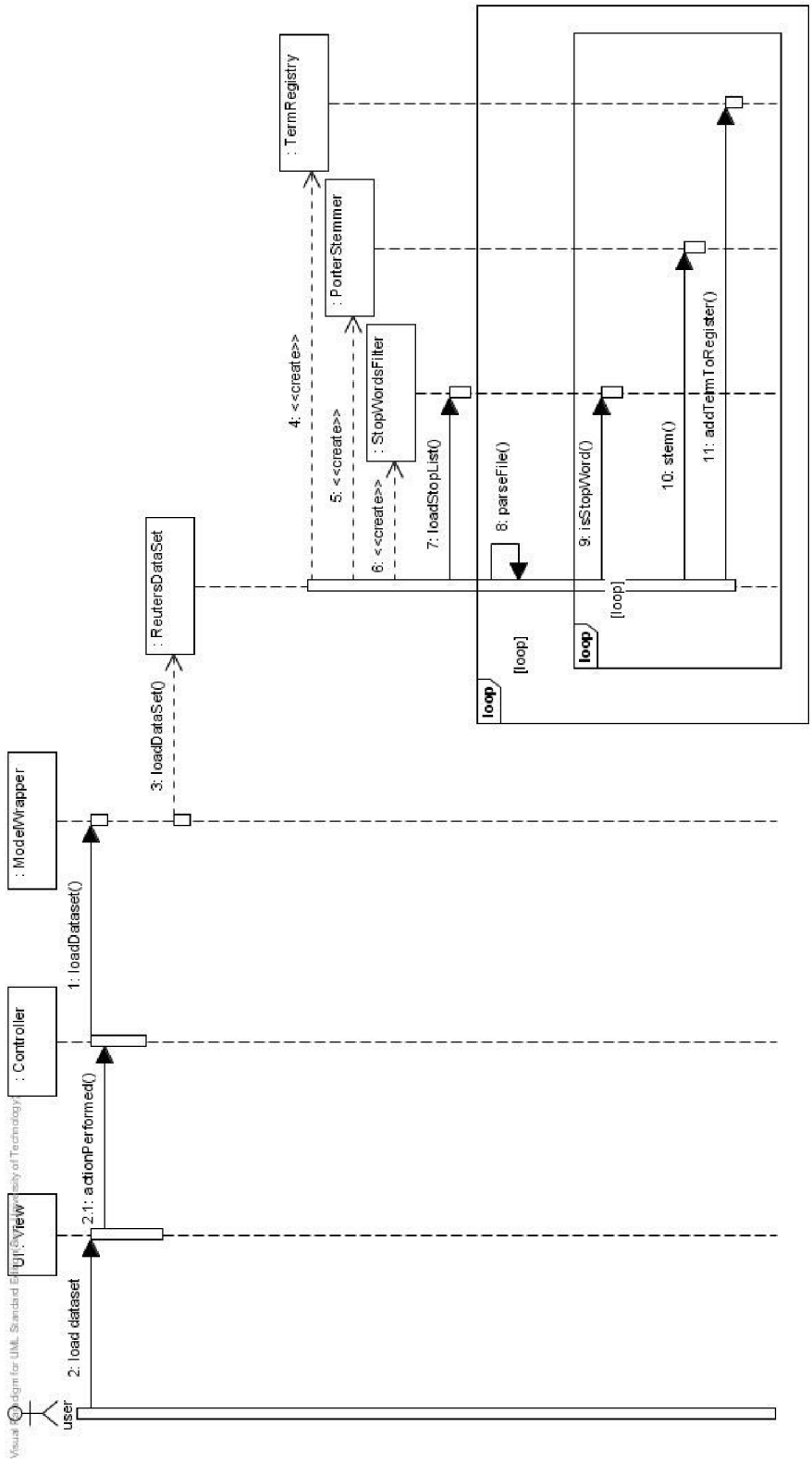

*Obrázek 12: diagram sekvence trénování klasifikátoru*

### 4.3.2 Načtení datasetu

Po volbě načtení datasetu je nejprve vytvořen objekt třídy implementující rozhraní `IDataSet`, aktuálně je to třída `ReutersDataSet`. Ta dále vytváří objekty dalších tříd, které využívá pro svoji činnost. Jsou jimi `StopWordsFilter` – třída, která ze zadaného souboru načte seznam stop slov, dále `PorterStemmer`, poskytující aplikaci stemming a nakonec `TermRegistry` – centrální registr termů.

Protože se dataset skládá z více souborů, následuje cyklické volání metody `parseFile()` provádějící načtení jednoho souboru. V rámci načítání jsou postupně detekovány jednotlivé dokumenty – pro každý je vytvořena nová instance třídy `Document`. Zatím se obsah dokumentu vyskytuje pouze ve formě jednoho textového řetězce, následuje jeho rozdělení na jednotlivá slova, poté je každé ověřeno vůči seznamu stop slov metodou `isStopWord()`. Projde-li slovo filtrem a má-li více než dva znaky, je získán jeho slovní kořen metodou `stem()` třídy `PorterStemmer` a uloží se v centrálním registru termů. Index do tohoto registru se uloží v nové instanci třídy `Term`, která se přidá do kolekce termů uvnitř objektu třídy `Document`.

Zvlášť se extrahují kategorie přiřazené k dokumentu a příznak, zda je dokument určen k testování nebo k trénování. Příznak se dokumentu předává přes konstruktor. Výše popsané volání metod a provádění dalších akcí se nachází uvnitř metody `BuildDocument()` třídy `ReutersDataSet`. Tato metoda je zodpovědná za sestavení jednoho dokumentu, který se následně vloží do kolekce všech dokumentů



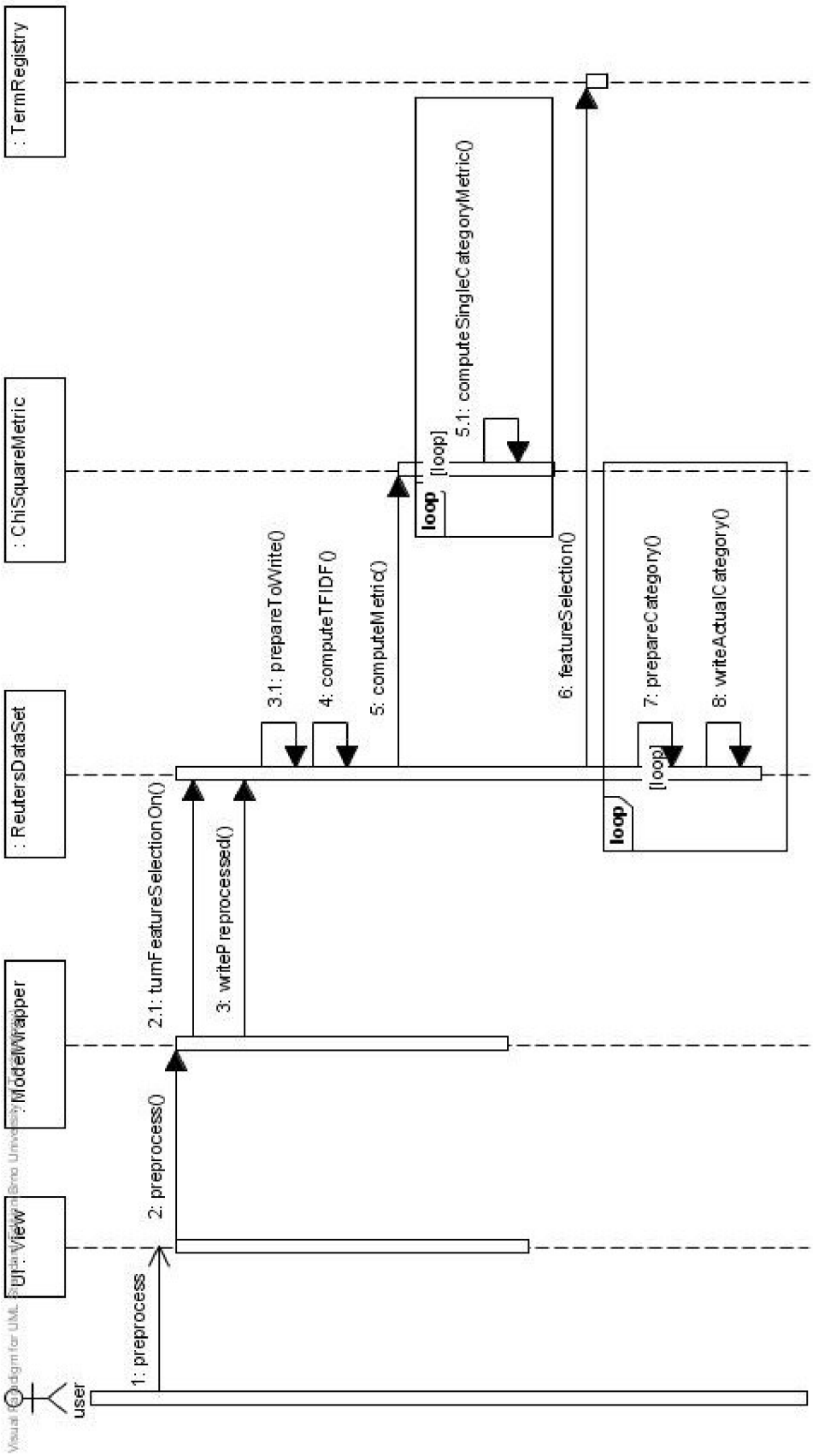

*Obrázek 13: diagram sekvence načtení datasetu*

### 4.3.3 Předzpracování

Diagram sekvence předzpracování znázorňuje situaci, kdy je už načtený dataset, uživatel vybral kategorie, nastavil parametry a úlohou aplikace je vytvořit trénovací a testovací soubory.

Vytvoření souborů začíná výpočtem reprezentace dokumentu – tedy hodnot TF nebo TFIDF (na binární reprezentaci netřeba nic počítat). Dále se vypustí z dokumentů všechny kategorie, které nebyly uživatelem vybrány. Dalším krokem je realizace selekce rysů, tedy nejprve výpočet požadované metriky, jejíž výběr se odrazí v hodnotě parametru předanému metodě `turnFeatureSelectionOn()` třídy `AbstractDataSet`. Zde je vytvořena instance odpovídající třídy, která výpočet zvolené metriky implementuje. Výpočet provádí metoda `computeMetric()`, přičemž pro získání celkové hodnoty metriky všech kategorií každého termu se musí vypočítat hodnota pro konkrétní kategorii. Děje se tak v metodě `computeSingleCategoryMetric()`.

Následuje volání metody `featureSelection()` třídy `TermRegistry`, která provede selekci rysů. Veškeré popsané operace se provádí uvnitř metody `PrepareToWrite()`. Nyní přichází na řadu příprava pozitivních a negativních příkladů do trénovací a testovací množiny, což je úkolem metody `prepareCategory()`. Nakonec jsou dokumenty zapsány do souborů metodou `writeActualCategory()`.




*Obrázek 14: diagram sekvence předzpracování*

# 5 Popis řešení

Jako doplnění a upřesnění předchozí kapitoly následuje popis implementace vybraných částí projektu. Je zmíněna struktura datasetu Reuters a jednotlivých článků, dále jsou detailně popsány určité postupy předzpracování, především metody selekce rysů. Rovněž je uvedeno několik poznámek ohledně implementace klasifikace.

## 5.1 Dataset Reuters-21578

Dataset Reuters-21578 je často používaným datasetem pro testování a porovnávání metod klasifikace dokumentů a IR. Tvoří ho nasbírané články agentury Reuters uspořádané do kolekce 22 souborů ve formátu sgml. Jak název napovídá, kolekce obsahuje celkem 21 578 dokumentů, jimž je přiřazeno celkem 135 různých kategorií. Každému dokumentu může být přiřazeno i několik kategorií nebo také žádná.

Každý dokument je uveden párovým tagem:

<REUTERS TOPICS=? LEWISSPLIT=? CGISPLIT=? OLDID=? NEWID=?>, kde otazníky značí hodnoty jednotlivých atributů, jak bude popsáno dále. Atribut TOPICS informuje, zda má článek přiřazenou nějakou kategorii, nabývá hodnot „YES“, „NO“ a „BYPASS“, což znamená, že tento článek nebyl vůbec indexován. Atribut LEWISSPLIT resp. CGISPLIT reprezentuje již použité rozdělení článků na trénovací a testovací příklady. Může mít hodnotu „TRAIN“, „TEST“ nebo „NOT-USED“, resp. „TRAINING-SET“ nebo „PUBLISHED-TESTSET“. Zbylé dva atributy jsou identifikátory článků staré a novější verze datasetu.

Témata článku se nachází uvnitř párového tagu <TOPICS>, každé téma oddělené tagem <D>. Vlastní obsah článku je uzavřen v párovém tagu <TEXT> a dále rozčleněn mezi tagy <AUTHOR>, <DATELINE>, <TITLE> a <BODY>.

V literatuře se lze často setkat s prezentací úspěšnosti klasifikace na deseti nejčastěji se vyskytujících kategoriích tohoto datasetu, jsou jimi: earn, acquisition, money-fx, grain, crude, trade, interest, ship, wheat, corn.

## 5.2 Předzpracování

### 5.2.1 Načítání dokumentů

Dokument udržovaný v paměti počítače obsahuje kategorie, termy a příznak, zda se bude používat k trénování nebo testování klasifikátoru. Tento příznak je určen podle dělení „ModApte“, které je založeno na hodnotách atributů LEWISSPLIT a TOPICS:

- LEWISSPLIT = TRAIN, TOPICS = „YES“ – jedná se o trénovací dokument.
- LEWISSPLIT = TEST, TOPICS = „YES“ – jedná se o testovací dokument.
- LEWISSPLIT = NOT-USED, TOPICS = „YES“
  - nebo TOPICS = „NO“
  - nebo TOPICS = „BYPASS“ – dokument bude vynechán z dalšího zpracování

Jednotlivé termy jsou před přidáním do dokumentu ověřeny vůči seznamu stop slov a podrobeny stemmingu, k čemuž je použit volně dostupný Porter stemmer. Jak z důvodu úspory paměti, tak z důvodů, které budou popsány dále, se termy neukládají v dokumentech přímo, ale pouze ve formě indexů do centrálního registru.

### 5.2.2 Prostor rysů a jejich selekce

Veškeré různé slovní kořeny v celém datasetu tvoří prostor rysů. Jeho rozměr se pohybuje nad číslem 17 000 v závislosti na použitém seznamu stop slov. Tento prostor rysů je seřazen tak, jak se slovní kořeny postupně vyskytovaly v datasetu. To je další důvod zavedení centrálního registru termů, protože každý term je jakousi souřadnicí prostoru vysoké dimenze, je nutné udržet konzistenci – každý dokument musí respektovat umístění svých termů v tomto prostoru.

Pro selekci rysů je nutné nejprve vypočítat nějakou metriku, která stanoví, jak je který slovní kořen přínosný pro klasifikaci dokumentu. Základem většiny běžně používaných metrik je výpočet tabulky pro každý slovní kořen, která má následující podobu:

| | c | not c |
|---|---|---|
| w | A | B |
| not w | C | D |

*Tabulka 1: základ pro výpočet metrik.*

kde „c“ resp. „not c“ značí, že dokument patří resp. nepatří do dané kategorie a „w“ resp. „not w“ znamená, že tento dokument daný slovní kořen obsahuje resp. neobsahuje. Číslo „A“ tedy představuje počet dokumentů, které obsahují slovní kořen „w“ a zároveň patří do kategorie „c“.

Pro každý slovní kořen se v centrálním registru ukládají hodnoty A a B, díky čemuž lze zmíněnou tabulku vypočítat v jednom průchodu přes kolekci dokumentů a jejich termy. Dále se ukládá výsledná metrika a počet dokumentů, ve kterých se daný term vyskytuje (DF je vypočítáno hned při načtení datasetu).

Takto lze vypočítat metriku pro jednu kategorii, pokud je požadováno zpracování několika kategorií naráz, musí se spočítat pro každou zvlášť a výsledná metrika je dána buď součtem metrik dílčích nebo maximální hodnotou z nich. Registr termů je podle vypočtené metriky sestupně seřazen a určité procento nejméně významných termů je označeno jako vypuštěné. Původní stav registru je zálohován, aby se kdykoli mohla provést selekce rysů podle jiné metriky, aniž by se musel znovu načítat celý dataset. Následuje krátký popis metrik implementovaných v této práci.

Document Frequency – není třeba počítat, viz výše. S využitím uvedené tabulky by se dalo spočítat DF = A + B. Jedná se o základní techniku selekce rysů, kdy jsou filtrovány termy, které se vyskytují v příliš málo dokumentech (porovnání vůči fixně dané hranici). Vylepšení výsledků klasifikace lze očekávat, pokud zřídka se vyskytující termy jsou šumové. Pokud je požadována agresivnější selekce rysů, DF zde uplatnění nenajde, protože tyto zřídka se vyskytující termy nemusí být jen šumem, ale nositeli důležité informace (tento předpoklad pochází z oblasti IR systémů).

Categorical Proportional Difference (CPD) udává, jakou měrou term přispívá ke vzájemné odlišnosti kategorií. Může nabývat hodnot z intervalu (-1, 1>, přičemž hodnota blížící se k -1 znamená, že daný term se nachází přibližně ve stejném množství dokumentů pro každou kategorii. Naopak hodnota blížící se k 1 značí, že daný term se objevuje v dokumentech jen jediné kategorie. Z jednotlivých hodnot pro každou kategorii se vybírá maximální.

$$CPD = \frac{A-B}{A+B} \quad . \tag{5.1}$$

Mutual Information (MI) ukazuje vzájemnou souvislost mezi termem a kategorií. Opět se vybírá maximální hodnota ze všech kategorií.

$$MI = \log \frac{A\ N}{(A+B)(A+C)}, \quad \text{kde N = A + B + C + D.} \tag{5.2}$$

Odds Ratio (OR) porovnává pravděpodobnost, že term do kategorie patří, s pravděpodobností, že do ní nepatří. Celkovou hodnotou je suma přes dílčí hodnoty jednotlivých kategorií.

$$\mathrm{OR} = \frac{A\ D}{B\ C} \quad . \tag{5.3}$$

Chi square je často používanou metrikou pro selekci rysů v oblasti dolování dat obecně, nejen textu. Chi square znázorňuje vzájemnou závislost dvou proměnných – v tomto kontextu termu a kategorie. Celkovou hodnotou je maximum z hodnot dílčích.

$$\chi^2 = \frac{N\ (AD-BC)^2}{(A+C)(B+D)(A+B)(C+D)} \quad . \tag{5.4}$$

Simplified chi square je zjednodušenou variantou chi square, i pro tuto metriku se celková hodnota získá z maxima hodnot dílčích.

$$s-\chi^2 = \frac{AD-BC}{N^2} \quad . \tag{5.5}$$

### 5.2.3 Příprava dat pro klasifikátor

Dokument je na výstupu předzpracování transformován do podoby řídkého vektoru, jehož indexy odpovídají pozicím jednotlivých termů v seřazeném centrálním registru a hodnoty jsou reprezentací těchto termů v podobě binární, TF nebo TF-IDF.

Program zapisuje data pro klasifikátor do dvou souborů pro každou kategorii – zvlášť trénovací a testovací. Výsledná podoba výstupu předzpracování je ovlivněna zvolenou kategorií, maximálním počtem trénovacích a testovacích příkladů a selekcí rysů. Pro každou kategorii se vyberou související dokumenty jako množina pozitivních příkladů, všechny ostatní dokumenty slouží jako výchozí množina negativních příkladů. Mezi negativní příklady patří i dokumenty, jejichž atribut TOPICS má sice hodnotu „YES“, ale párový tag TOPICS žádné kategorie neobsahuje. Množiny pozitivních i negativních příkladů se dále upraví, aby obsahovaly stejný počet prvků, menší nebo roven zadanému maximálnímu počtu.

Co se týká formátu výstupních souborů předzpracování, byl zvolen formát kompatibilní s klasifikátorem SVM light. Každý řádek reprezentující jeden dokument je tvořen sloupci oddělenými mezerou. První sloupec má hodnotu 1 resp. -1, tj. kategorie dokumentu, následující sloupce jsou ve tvaru index:hodnota, což umožňuje data ukládat jako řídké vektory. Na prvním řádku soubory obsahují dimenzi prostoru rysů a odpovídající kategorii, která nástroji umožní spárovat správné trénovací a testovací soubory, pokud jich uživatel vybere víc najednou. Následující obrázek ukazuje fragment souboru pro kategorii trade.

```
1 #dimension=19190 category=trade
2 +1 22:0.007715704895080535 28:0.01091048609400389 30:0.1758097
3 +1 11:0.020751316688595634 22:0.2389995831574417 25:0.05573561
4 +1 11:0.08571733938555005 22:0.004912187379235463 28:0.0163882
5 +1 22:0.006584421380677323 118:0.041396921657012496 122:4.6805
6 +1 3:0.012797737230041096 18:0.009386045200585707 22:0.0010782
7 +1 22:0.02906627613932746 28:0.08303643063892008 57:0.01258101
8 +1 12:0.03917560423097007 18:0.03409867053883668 22:0.00391731
```

*Obrázek 15: příklad výstupních dat předzpracování na kategorii trade*

## 5.3 Klasifikace

### 5.3.1 Implementace klasifikace metodou SVM

Implementace metody SVM se přísně drží algoritmu a popisu v [5]. Pro zvýšení rychlosti výpočtů při učení klasifikátoru existuje několik optimalizací. Jednou z nich je použití vah u lineárního klasifikátoru, dále jsou to různé možnosti uložení již vypočítaných výsledků. Velmi často se počítá hodnota jádrové funkce pro dvojice vstupních vektorů. Dobu výpočtu lze znatelně zkrátit uložením této hodnoty. Pro soubor čítající 2 000 trénovacích příkladů to dělá 4 000 000 hodnot, což je pro současné prostředky výpočetní techniky snesitelné.

Toto opatření se ukázalo jako nutné při implementaci nelineárního klasifikátoru. Algoritmus jeho učení je výpočetně náročnější, než je tomu u klasifikátoru lineárního – je to právě ještě častějším vyhodnocováním jádrové funkce. V důsledku toho by byl proces učení bez ukládání hodnot jádrové funkce ukončen v rozumném čase jen pro menší soubory trénovacích dat.

### 5.3.2 Ukládání a načítání naučeného klasifikátoru

Jakmile je klasifikátor pro nějaká typická data natrénován, jeví se užitečné mít možnost uložení a načtení parametrů potřebných pro následnou klasifikaci. Je proto dobrým zvykem tuto funkčnost v klasifikačních nástrojích podporovat, protože doba trénování (u rozsáhlé množiny vysoce dimenzionálních dat) může být dosti dlouhá.

Nejprve si musíme položit otázku, jaká data musí mít klasifikátor k dispozici, aby mohl vypočítat odpovídající třídu dokumentu. Odpověď dávají rovnice 3.12 a 3.17, přičemž řešení se pro lineární a nelineární klasifikaci rozchází. Lineární klasifikátor kromě vstupního vektoru, který reprezentuje aktuálně klasifikovaný dokument, potřebuje znát hodnoty vektoru vah a hodnotu prahu. Oproti tomu nelineární klasifikátor potřebuje ukládat Lagrangeovy koeficienty, podpůrné vektory (vektory trénovacích dat, které leží na hranicích dělící nadroviny), odpovídající třídu (1 resp. -1) jednotlivých podpůrných vektorů a hodnotu prahu. Rovněž je třeba uložit parametry potřebné k výpočtu hodnoty jádrové funkce (například stupeň polynomu u polynomiální funkce). Pokud je implementováno více jádrových funkcí, musí se také ukládat typ použité jádrové funkce.

Dále se ukládají i ostatní parametry zadané při trénování, aby uživatel měl představu za jakých podmínek byl klasifikátor naučen, když si jeho parametry ze souboru načte. Kromě toho může nastat jistá problematická situace. Uživatel uloží parametry klasifikátoru, který se naučil například kategorii crude v reprezentaci TF. Později načte tyto parametry ze souboru a jako testovací data zvolí zcela jinou kategorii nebo reprezentaci dokumentů. Z tohoto důvodu se ukládá i kategorie a reprezentace trénovacích dat.

Na obrázku 16 je vidět příklad fragmentu souboru s parametry. Jedná se o lineární klasifikátor trénovaný na kategorii acqusition s binární reprezentací dokumentů. Slovem „weights“ se uvádí následující řádek, který obsahuje jednotlivé hodnoty vektoru vah oddělené čárkou.

```
1 kernel=LINEAR
2 category=acq
3 reprezentation=BINARY
4 epsilon=0.0010
5 c=0.5
6 b=0.9368874
7 dimension=19190
8 weights
9 0.0,0.0,-0.039537594,-0.14836687,0.0,0.13141909,-0.12730637,0.0298
10
```

*Obrázek 16: příklad uložení parametrů pro lineární klasifikátor*

Další obrázek ukazuje uložení parametrů pro nelineární klasifikátor, který používá RBF jádrovou funkci a byl naučen na kategorii interest opět s binární reprezentací dokumentů. Slovo „alpha“ značí, že následující řádek obsahuje Lagrangeovy koeficienty oddělené čárkou. Stejně tak slovní spojení „support vectors“ označuje pozici v souboru, kde na dalším řádku začínají podpůrné vektory. Jejich formát uložení je shodný s formátem trénovacích dat po předzpracování. Tedy každý řádek reprezentuje jeden vektor, kde hodnoty jsou oddělené čarkou, přičemž první sloupec je třída tohoto vektoru.

```
 1 kernel=RBF
 2 category=interest
 3 reprezentation=BINARY
 4 epsilon=0.0010
 5 c=250.0
 6 b=0.064552665
 7 sigma=3.0
 8 dimension=19190
 9 alpha
10 1.0644821,1.0647972,1.0648315,1.0646733,1.0622575,1.0652249,
11 support vectors
12 1 6:1.0 22:1.0 36:1.0 41:1.0 72:1.0 122:1.0 124:1.0 125:1.0
13 1 22:1.0 38:1.0 54:1.0 122:1.0 141:1.0 186:1.0 226:1.0 230:1
14 1 6:1.0 18:1.0 22:1.0 25:1.0 28:1.0 30:1.0 38:1.0 40:1.0 44:
15 1 22:1.0 38:1.0 41:1.0 49:1.0 97:1.0 122:1.0 226:1.0 234:1.0
16 1 22:1.0 36:1.0 38:1.0 49:1.0 122:1.0 230:1.0 238:1.0 247:1.
```

*Obrázek 17: příklad uložení parametrů pro RBF nelineární klasifikátor*

# 6 Testování

Pro ověření úspěšnosti klasifikátoru bylo třeba najít jiný SVM klasifikátor – byl zvolen velmi kvalitní SVM light. Dále pro srovnání mezi metodu SVM a jinými metodami klasifikace posloužil dolovací nástroj Weka. Konkrétně byla použita verze 3.6.2, která již podporuje shodný formát vstupních souborů jako SVM light (a tím i shodný s formátem implementovaného klasifikačního nástroje).

## 6.1 Postup testování

První část testů se zaměřuje na porovnání různých klasifikátorů a různých reprezentací dokumentů bez selekce rysů. Druhá část potom ukazuje vliv vybraných metod selekce rysů na úspěšnost klasifikace. Testování probíhá na deseti nejčastěji se vyskytujících kategoriích datasetu Reuters s nastaveným maximálním množstvím 2 000 trénovacích příkladů a 600 příkladů testovacích.

### 6.1.1 Testování v programu Text SVM

Vzhledem k tomu, že nelineární klasifikaci se nepodařilo implementovat tak dobře, aby s jedinou výjimkou (viz kapitola 6.2.1.1) podávala kvalitní výsledky (úspěšnost se pohybuje mírně nad hranicí 50%), budou výsledky prezentovány pouze pro lineární klasifikátor. Testování proběhlo s nastavením parametrů C = 0,5; epsilon 0,001 pro všechny kategorie. Výhodou je možnost trénování a testování několika kategorií naráz. Bližší informace k použití této aplikace lze nalézt v manuálu (příloha 1).

### 6.1.2 Testování v klasifikátoru SVM light

Klasifikátor SVM light se skládá ze dvou programů: svm_learn.exe provede trénování klasifikátoru a vytvoří soubor s jeho parametry, který využívá svm_classify.exe pro otestování. Parametry pro lineární klasifikaci byly ponechány v původním nastavení (program si sám stanovuje hodnotu parametru C na základě velikosti vektorů vstupních dat). Pro klasifikaci nelineární byla použita jádrová funkce RBF a parametr sigma byl nastaven na hodnotu 0,003. Podrobnější popis SVM light lze nalézt v [13].

### 6.1.3 Testování v dolovacím nástroji Weka

Původním záměrem bylo provést testování na školní instalaci dolovacího nástroje SAS Enterprise Miner, bohužel tento nástroj nepodporuje řídký formát dat.

Nástroj Weka poskytuje několik uživatelských rozhraní, na první pohled se jako nejlepší z nich jeví Knowledge Flow. Ovšem po bližším prozkoumání je patrné, že toto prostředí je ještě ve fázi vývoje a pro další testování nevhodné, proto bylo zvoleno prostředí Explorer.

Nástroj obsahuje celou řadu klasifikačních metod, ze které byly nakonec vybrány tři – conjunctive rule, naive Bayes a voted perceptron. Většina ostatních klasifikátorů neskončila učení v rozumném čase nebo skončila výjimkou pro nedostatek paměti (nástroji byl přidělen 1 GB).

Jako problém se ukázalo rozdělení dat na soubor trénovací a testovací, protože tento nástroj dovoluje rozdělení pouze v rámci svého nativního formátu dat. Východiskem bylo spojení obou souborů a následná volba, aby si data na množinu trénovací a testovací nástroj rozdělil sám náhodným výběrem. Tím bylo bohužel znemožněno použít původní rozdělení datasetu Reuters, ale poměr objemu trénovacích a testovacích dat zůstal přibližně zachován. Bližší informace o nástroji Weka lze nalézt v [12].

## 6.2 Výsledky

Pro menší rozměr tabulek je použito jen první písmeno názvů sloupců – P znamená přesnost (Precision), O odpověď (Recall) a Ú celková úspěšnost klasifikátoru (Accuracy). Tyto ukazatele, hojně využívané v oblasti IR systémů, jsou pro potřeby klasifikace definované následovně:

$$Recall = \frac{tp}{tp + fn} \quad , \tag{6.1}$$

$$Precision = \frac{tp}{tp + fp} \quad , \tag{6.2}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad , \tag{6.3}$$

kde *tp* znamené true positive – počet dokumentů korektně klasifikovaných do třídy +1, *tn* je true negative – počet dokumentů korektně klasifikovaných do třídy -1, dále *fp* resp. *fn* jsou false positive resp. false negative, tedy počet dokumentů chybně klasifikovaných do třídy +1 resp. -1.

### 6.2.1 Srovnání klasifikátorů

Následují tabulky s výsledky pro každou reprezentaci dokumentů a jednotlivé klasifikátory.

#### 6.2.1.1 Binární reprezentace

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 95,8 | 91,33 | 93,67 | 98,54 | 90 | 94,33 | 98,9 | 89,67 | 94,33 |
| Acquisition | 89,68 | 92,67 | 91 | 94,88 | 92,67 | 93,83 | 94,88 | 92,67 | 93,83 |
| Money-fx | 90,41 | 80,98 | 86,32 | 95,1 | 83,44 | 89,67 | 95,14 | 84,05 | 89,97 |
| Grain | 94,89 | 89,04 | 91,84 | 96,27 | 88,36 | 92,2 | 96,27 | 88,36 | 92,2 |
| Crude | 91,07 | 84,07 | 86,38 | 92,4 | 86,81 | 88,54 | 92,4 | 86,81 | 88,54 |
| Trade | 91,07 | 87,17 | 89,32 | 96,4 | 91,45 | 94,02 | 95,54 | 91,45 | 93,59 |
| Interest | 91,43 | 76,19 | 83,61 | 95,24 | 79,37 | 86,97 | 95,19 | 78,57 | 86,55 |
| Ship | 100 | 86,36 | 92,68 | 100 | 86,36 | 92,68 | 100 | 88,64 | 93,9 |
| Wheat | 97,01 | 91,55 | 94,87 | 98,48 | 91,55 | 95,51 | 98,48 | 91,55 | 95,51 |
| Corn | 90,38 | 83,93 | 87,93 | 95,83 | 82,14 | 89,66 | 95,83 | 82,14 | 89,66 |
| Průměr | 93,17 | 86,33 | 89,76 | 96,31 | 87,22 | 91,74 | 96,26 | 87,39 | 91,81 |

*Tabulka 2: binární reprezentace dokumentu, klasifikace metodou SVM*

| Kategorie | conjunctive rules | | | naive Bayes | | | voted perceptron | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 87 | 83,9 | 83,95 | 92,3 | 91,0 | 90,97 | 95,4 | 95,3 | 95,32 |
| Acquisition | 77,6 | 61,9 | 61,87 | 90,7 | 90,5 | 90,47 | 88,9 | 88,6 | 88,63 |
| Money-fx | 80,1 | 67,7 | 67,74 | 87,2 | 87,1 | 87,1 | 92,3 | 92,3 | 92,26 |
| Grain | 83,4 | 75,6 | 75,57 | 83,3 | 82,8 | 82,82 | 87,1 | 87 | 87,02 |
| Crude | 89,7 | 89,6 | 89,62 | 84,2 | 83,1 | 83,08 | 88,3 | 88,1 | 88,08 |
| Trade | 91 | 90,9 | 90,91 | 86,7 | 86,4 | 86,36 | 91,8 | 91,8 | 91,82 |
| Interest | 79,1 | 77,3 | 77,25 | 82 | 82 | 81,99 | 88,7 | 88,6 | 88,63 |
| Ship | 84,3 | 79,4 | 79,39 | 93,2 | 93,1 | 93,13 | 89,8 | 89,3 | 89,31 |
| Wheat | 95,8 | 95,4 | 95,38 | 89,5 | 89,2 | 89,23 | 89,2 | 89,2 | 89,23 |
| Corn | 89,6 | 87 | 87,04 | 92,1 | 91,7 | 91,67 | 89,8 | 89,8 | 89,81 |
| Průměr | 85,76 | 80,87 | 80,87 | 88,12 | 87,69 | 87,68 | 90,13 | 90 | 90,01 |

*Tabulka 3: binární reprezentace dokumentu, ostatní metody klasifikace*

Během různých experimentů s implementovaným klasifikátorem bylo dosaženo až překvapivě výborného výsledku (průměrně 97,8 % úspěšnosti) při nastavení parametrů: C = 250; epsilon = 0,001; sigma = 3. Ačkoli na ostatních reprezentacích dokumentů má tento nelineární klasifikátor velmi slabé výsledky, následující výjimka je poměrně zajímavá (tabulka 4).

| Kategorie | text svm – RBF | | |
|---|---|---|---|
| | P [%] | O [%] | Ú [%] |
| Earn | 100 | 82,33 | 91,17 |
| Acquisition | 99,34 | 99,67 | 99,5 |
| Money-fx | 100 | 93,25 | 96,65 |
| Grain | 100 | 98,63 | 99,29 |
| Crude | 100 | 96,7 | 98,14 |
| Trade | 95,9 | 100 | 97,86 |
| Interest | 100 | 93,65 | 96,64 |
| Ship | 100 | 97,73 | 98,78 |
| Wheat | 100 | 100 | 100 |
| Corn | 100 | 100 | 100 |
| Průměr | 99,52 | 96,2 | 97,8 |

*Tabulka 4: binární reprezentace dokumentu, RBF klasifikátor*


*Obrázek 18: graf úspěšnosti klasifikátorů na binární reprezentaci dokumentu*

Je patrné, že všechny SVM klasifikátory se s binární reprezentací dokumentu vyrovnaly velmi dobře (kolem 90 %). Z ostatních klasifikátorů vykazuje srovnatelný výkon pouze voted perceptron, nicméně všechny se dostaly přes hranici 80 %. Nejhůře dopadl klasifikátor conjunctive rules, ačkoli na některých kategoriích (trade, wheat) si vedl srovnatelně dobře s metodou SVM.

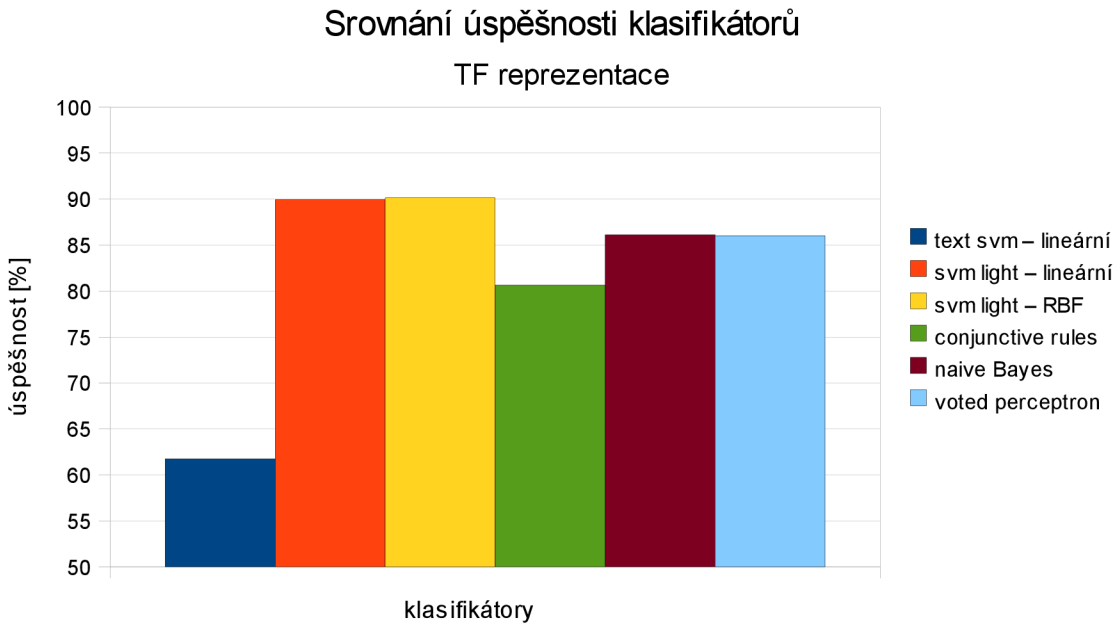
#### 6.2.1.2 TF reprezentace

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 99,47 | 63,0 | 81,33 | 97,35 | 85,67 | 91,67 | 97,35 | 85,67 | 91,67 |
| Acquisition | 61,37 | 98,0 | 68,17 | 96,80 | 90,67 | 93,83 | 96,80 | 90,67 | 93,83 |
| Money-fx | 56,32 | 95,70 | 62,02 | 85,89 | 85,89 | 86,35 | 86,96 | 85,89 | 86,94 |
| Grain | 64,22 | 95,89 | 70,63 | 98,47 | 88,36 | 93,36 | 97,73 | 88,36 | 93,01 |
| Crude | 0,0 | 0,0 | 44,0 | 92,98 | 87,36 | 89,23 | 94,08 | 87,36 | 89,85 |
| Trade | 58,88 | 99,15 | 64,81 | 85,50 | 95,73 | 89,70 | 84,96 | 96,58 | 89,7 |
| Interest | 89,22 | 72,22 | 80,43 | 90,0 | 78,57 | 83,83 | 91,07 | 80,95 | 85,53 |
| Ship | 0,0 | 0,0 | 46,99 | 90,22 | 94,32 | 91,57 | 90,22 | 94,32 | 91,57 |
| Wheat | 100,0 | 5,63 | 49,62 | 92,75 | 90,14 | 90,98 | 92,65 | 88,73 | 90,23 |
| Corn | 48,70 | 100,0 | 49,57 | 92,16 | 83,93 | 88,89 | 92,16 | 83,93 | 88,89 |
| Průměr | 57,82 | 62,4 | 61,76 | 92,21 | 88,06 | 89,94 | 92,4 | 88,25 | 90,12 |

*Tabulka 5: TF reprezentace dokumentu, klasifikace metodou SVM*

| Kategorie | conjunctive rules | | | naive Bayes | | | voted perceptron | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 86 | 83,3 | 83,28 | 89 | 89 | 88,96 | 87,5 | 86,3 | 86,29 |
| Acquisition | 77,1 | 61,7 | 61,71 | 86,3 | 86,3 | 86,29 | 88 | 88 | 87,96 |
| Money-fx | 77,3 | 74,8 | 74,84 | 87,1 | 87,1 | 87,1 | 86,7 | 86,5 | 86,45 |
| Grain | 82,5 | 73,7 | 73,66 | 83,6 | 83,6 | 83,6 | 80,1 | 79,4 | 79,4 |
| Crude | 92,5 | 92,3 | 92,31 | 88,1 | 88,1 | 88,08 | 89,2 | 88,8 | 88,85 |
| Trade | 87,8 | 87,7 | 87,73 | 86,8 | 86,8 | 86,82 | 90 | 89,5 | 89,55 |
| Interest | 77,3 | 75,4 | 75,36 | 86,5 | 86,3 | 86,26 | 81,4 | 81 | 81,04 |
| Ship | 84,5 | 77,9 | 77,86 | 89,5 | 89,3 | 89,31 | 92 | 91,6 | 91,6 |
| Wheat | 95,8 | 95,4 | 95,38 | 82,4 | 81,5 | 81,54 | 85,6 | 84,6 | 84,62 |
| Corn | 87 | 84,3 | 84,26 | 83,8 | 83,3 | 83,33 | 85 | 84,3 | 84,26 |
| Průměr | 84,78 | 80,65 | 80,64 | 86,31 | 86,13 | 86,13 | 86,55 | 86 | 86 |

*Tabulka 6: TF reprezentace dokumentu, ostatní metody klasifikace*




*Obrázek 19: graf úspěšnosti klasifikátorů na TF reprezentaci dokumentu*

S touto reprezentací se implementovaný klasifikátor vypořádal poměrně špatně, u ostatních klasifikátorů lze pozorovat mírné snížení úspěšnosti. Klasifikátor SVM light předvedl nejlepší výsledky (opět kolem 90 %), přičemž použití RBF jádrové funkce přineslo oproti lineární variantě jen stěží znatelné zlepšení.

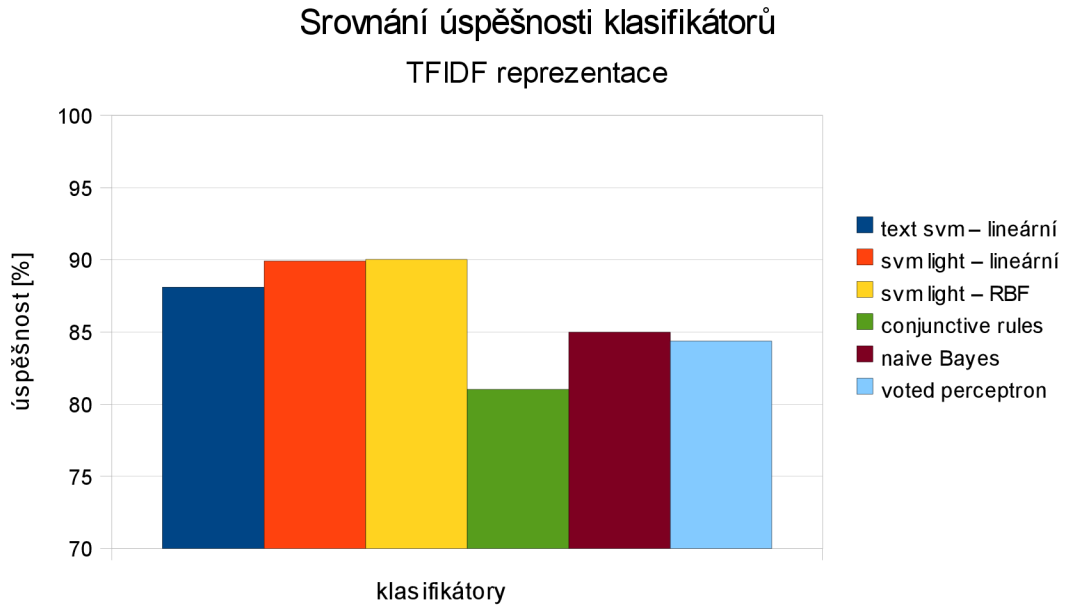
#### 6.2.1.3 TFIDF reprezentace

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 98,63 | 72,00 | 85,50 | 97,23 | 82,00 | 89,83 | 97,28 | 83,33 | 90,5 |
| Acquisition | 93,20 | 91,33 | 92,33 | 93,86 | 91,67 | 92,83 | 93,86 | 91,67 | 92,83 |
| Money-fx | 87,65 | 87,12 | 88,01 | 87,20 | 87,73 | 88,01 | 87,2 | 87,73 | 88,01 |
| Grain | 96,82 | 83,56 | 89,96 | 94,78 | 86,99 | 90,68 | 95,45 | 86,3 | 90,68 |
| Crude | 96,18 | 82,97 | 88,72 | 95,76 | 86,81 | 90,55 | 95,81 | 87,91 | 91,16 |
| Trade | 84,85 | 95,73 | 89,54 | 92,50 | 94,87 | 93,72 | 91,74 | 94,87 | 93,31 |
| Interest | 97,61 | 65,08 | 81,07 | 88,07 | 76,19 | 82,30 | 88,79 | 75,4 | 82,3 |
| Ship | 98,73 | 88,64 | 93,04 | 98,73 | 88,64 | 93,04 | 97,5 | 88,64 | 92,41 |
| Wheat | 88,57 | 87,32 | 88,11 | 89,04 | 91,55 | 90,21 | 89,04 | 91,55 | 90,21 |
| Corn | 81,36 | 85,71 | 84,80 | 90,20 | 82,14 | 88,00 | 90,38 | 83,93 | 88,8 |
| Průměr | 92,36 | 83,95 | 88,11 | 92,74 | 86,86 | 89,92 | 92,71 | 87,13 | 90,02 |

*Tabulka 7: TFIDF reprezentace dokumentu, klasifikace metodou SVM*

| Kategorie | conjunctive rules | | | naive Bayes | | | voted perceptron | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 86,2 | 83,4 | 83,44 | 86,5 | 86,5 | 86,45 | 93,3 | 93,1 | 93,14 |
| Acquisition | 77,1 | 61,7 | 61,71 | 83,4 | 83,4 | 83,44 | 86,4 | 86,1 | 86,12 |
| Money-fx | 81 | 77,1 | 77,1 | 85,9 | 85,8 | 85,81 | 86,2 | 85,8 | 85,81 |
| Grain | 83,6 | 76 | 75,95 | 85,3 | 85,1 | 85,11 | 83 | 80,9 | 80,92 |
| Crude | 89,7 | 89,6 | 89,62 | 85,8 | 85,8 | 85,77 | 83,4 | 83,1 | 83,08 |
| Trade | 88,7 | 88,6 | 88,64 | 86,2 | 85,9 | 85,91 | 86,4 | 86,4 | 86,36 |
| Interest | 79,4 | 76,3 | 76,3 | 80,1 | 80,1 | 80,09 | 81,4 | 81 | 81,04 |
| Ship | 86,3 | 80,2 | 80,15 | 87,8 | 87,8 | 87,79 | 83,2 | 83,2 | 83,21 |
| Wheat | 93,7 | 93,1 | 93,1 | 87 | 86,9 | 86,92 | 86,6 | 85,4 | 85,38 |
| Corn | 87,9 | 84,3 | 84,26 | 84 | 82,4 | 82,41 | 80,2 | 78,7 | 78,7 |
| Průměr | 85,36 | 81,03 | 81,03 | 85,2 | 84,97 | 84,97 | 85,01 | 84,37 | 84,38 |

*Tabulka 8: TFIDF reprezentace dokumentu, ostatní metody klasifikace*




*Obrázek 20: graf úspěšnosti klasifikátorů na TFIDF reprezentaci dokumentu*

Výsledky jsou velmi podobné jako u minulé reprezentace, tentokrát si ale implementovaný klasifikátor s daty poradil dobře. Je zde vidět dominance metody SVM nad ostatními klasifikátory. Klasifikátor conjunctive rules opět zaostává za ostatními, ale je zajímavé, že má téměř stejné průměrné výsledky u všech reprezentací (viz předchozí tabulky a grafy).

### 6.2.2 Vliv selekce rysů

Následující podkapitoly ukazují vliv selekce rysů na úspěšnost klasifikace. Testování probíhalo jen na SVM klasifikátorech (opět lineární text SVM a SVM light jak lineární, tak i s RBF jádrovou

funkcí). Pro porovnání byly zvoleny metriky chi-square, document frequency a mutual information. Pro každou z těchto metrik byly provedeny dva experimenty, kdy se lišila pouze míra selekce rysů. V prvním experimentu bylo vybráno 80 % a v druhém jen 50 % nejlepších rysů.

#### 6.2.2.1 Chi-square

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 98,71 | 76,33 | 87,67 | 97,67 | 83,67 | 90,83 | 98,41 | 82,67 | 90,67 |
| Acquisition | 95,36 | 89 | 92,33 | 94,08 | 90,0 | 92,17 | 94,08 | 90 | 92,17 |
| Money-fx | 90,51 | 87,73 | 89,06 | 88,54 | 85,28 | 86,88 | 91,56 | 86,5 | 89,06 |
| Grain | 94,44 | 81,51 | 88,07 | 94,57 | 83,56 | 89,12 | 93,13 | 83,56 | 88,42 |
| Crude | 98,05 | 82,97 | 89,67 | 95,68 | 85,16 | 89,67 | 96,79 | 82,97 | 89,06 |
| Trade | 84,21 | 95,72 | 89,12 | 86,05 | 94,87 | 89,96 | 88,89 | 95,73 | 92,05 |
| Interest | 95,06 | 61,11 | 79,05 | 90,57 | 76,19 | 84,19 | 89,91 | 77,78 | 84,58 |
| Ship | 98,68 | 85,23 | 90,73 | 97,53 | 89,77 | 92,72 | 92,94 | 89,77 | 90,07 |
| Wheat | 91,18 | 87,32 | 88,72 | 89,86 | 87,32 | 87,97 | 91,3 | 88,73 | 89,47 |
| Corn | 95,12 | 69,64 | 83,19 | 93,48 | 76,79 | 85,84 | 95,35 | 73,21 | 84,96 |
| Průměr | 94,13 | 81,66 | 87,76 | 92,8 | 85,26 | 88,94 | 93,24 | 85,09 | 89,05 |

*Tabulka 9: vliv selekce rysů: chi-square 80 %*

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 95,85 | 61,67 | 79,5 | 92,67 | 84,33 | 88,83 | 94,44 | 79,33 | 87,33 |
| Acquisition | 95,85 | 86 | 90,67 | 92,31 | 88,0 | 90,33 | 92,31 | 88 | 90,33 |
| Money-fx | 96,09 | 75,46 | 86,73 | 90,91 | 85,89 | 89,09 | 91,16 | 82,21 | 87,61 |
| Grain | 95,45 | 71,92 | 83,92 | 93,13 | 83,56 | 88,45 | 92,54 | 84,93 | 88,81 |
| Crude | 98,28 | 62,64 | 78,72 | 94,77 | 79,67 | 86,32 | 92,95 | 79,67 | 85,41 |
| Trade | 87,39 | 88,89 | 87,5 | 85,37 | 89,74 | 86,61 | 83,33 | 89,74 | 85,27 |
| Interest | 98,08 | 40,48 | 70,54 | 90,63 | 69,05 | 81,40 | 91,21 | 65,87 | 80,23 |
| Ship | 100 | 63,64 | 79,22 | 97,4 | 85,23 | 90,26 | 97,47 | 87,5 | 91,56 |
| Wheat | 96,22 | 71,83 | 83,94 | 94,12 | 90,14 | 91,97 | 95,31 | 85,92 | 90,51 |
| Corn | 96,97 | 57,14 | 77,88 | 91,3 | 75,0 | 84,07 | 89,13 | 73,21 | 82,3 |
| Průměr | 96,02 | 67,97 | 81,86 | 92,26 | 83,06 | 87,73 | 91,99 | 81,64 | 86,94 |

*Tabulka 10: vliv selekce rysů: chi-square 50 %*

#### 6.2.2.2 Document Frequency

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 96,97 | 74,67 | 86,17 | 96,08 | 81,67 | 89,17 | 95,72 | 82 | 89,17 |
| Acquisition | 94,43 | 90,33 | 92,5 | 92,47 | 90 | 91,33 | 92,47 | 90 | 91,33 |
| Money-fx | 84,02 | 87,11 | 85,67 | 86,67 | 87,73 | 87,46 | 86,06 | 87,12 | 86,87 |
| Grain | 90,85 | 88,36 | 90,48 | 91,91 | 85,62 | 89,84 | 90,65 | 86,3 | 89,52 |
| Crude | 97,44 | 83,51 | 88,67 | 96,86 | 84,62 | 89 | 96,18 | 82,97 | 87,67 |
| Trade | 83,09 | 95,58 | 88,61 | 88,89 | 95,73 | 91,98 | 86,61 | 94,02 | 89,87 |
| Interest | 97,37 | 58,73 | 76,42 | 94,12 | 76,19 | 84,28 | 94 | 74,6 | 83,41 |
| Ship | 96,34 | 89,77 | 92,9 | 93,02 | 90,91 | 91,72 | 94,12 | 90,91 | 92,31 |
| Wheat | 95,08 | 81,69 | 87,5 | 92,54 | 87,32 | 89,06 | 92,42 | 85,92 | 88,28 |
| Corn | 97,3 | 64,29 | 80,19 | 95,56 | 76,79 | 85,85 | 97,67 | 75 | 85,85 |
| Průměr | 93,29 | 81,4 | 86,91 | 92,81 | 85,66 | 88,97 | 92,59 | 84,88 | 88,43 |

*Tabulka 11: vliv selekce rysů: DF 80 %*

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 99,12 | 74,67 | 87 | 98,03 | 83 | 90,67 | 9,2 | 82,67 | 91 |
| Acquisition | 93,64 | 88,33 | 91,17 | 93,4 | 89,67 | 91,67 | 93,4 | 89,67 | 91,67 |
| Money-fx | 89,17 | 85,89 | 88,27 | 86,34 | 85,28 | 86,51 | 88,39 | 84,05 | 87,1 |
| Grain | 96,15 | 85,62 | 90,55 | 94,03 | 86,3 | 89,82 | 92,7 | 86,99 | 89,45 |
| Crude | 96,1 | 81,32 | 87,54 | 94,51 | 85,16 | 88,79 | 95,65 | 84,62 | 89,1 |
| Trade | 81,95 | 93,16 | 86,61 | 84,5 | 93,16 | 88,28 | 83,08 | 92,31 | 87,03 |
| Interest | 93,67 | 58,73 | 75,11 | 90,38 | 74,6 | 81,66 | 91,26 | 74,6 | 82,1 |
| Ship | 96,3 | 88,64 | 91,28 | 92,68 | 86,36 | 87,92 | 91,67 | 87,5 | 87,92 |
| Wheat | 96,72 | 83,1 | 88,62 | 92,65 | 88,73 | 89,43 | 95,38 | 87,32 | 90,24 |
| Corn | 97,37 | 66,07 | 81,31 | 94 | 83,93 | 88,79 | 95,74 | 80,36 | 87,85 |
| Průměr | 94,02 | 80,55 | 86,75 | 92,05 | 85,62 | 88,35 | 83,65 | 85,01 | 88,35 |

*Tabulka 12: vliv selekce rysů: DF 50 %*

#### 6.2.2.3 Mutual Information

| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 99,54 | 73 | 86,33 | 98,43 | 83,33 | 91 | 96,88 | 82,67 | 90 |
| Acquisition | 93,68 | 89 | 91,5 | 94,74 | 90 | 92,5 | 94,74 | 90 | 92,5 |
| Money-fx | 90,91 | 85,89 | 88,92 | 87,58 | 86,5 | 87,43 | 90,73 | 84,05 | 88,02 |
| Grain | 95,45 | 86,3 | 91,22 | 96,21 | 86,99 | 91,89 | 95,45 | 86,3 | 91,22 |
| Crude | 96,69 | 80,22 | 87,31 | 96,15 | 82,42 | 88,24 | 96,1 | 81,32 | 87,62 |
| Trade | 85,27 | 94,02 | 89,17 | 92,37 | 93,16 | 92,92 | 91,74 | 94,87 | 93,33 |
| Interest | 95,95 | 56,35 | 75,32 | 92,08 | 73,81 | 82,55 | 95,96 | 75,4 | 85,11 |
| Ship | 97,3 | 81,82 | 88,31 | 95,06 | 87,5 | 90,26 | 95,06 | 87,5 | 90,26 |
| Wheat | 98,39 | 85,92 | 92,62 | 96,97 | 90,14 | 93,96 | 97,01 | 91,55 | 94,63 |
| Corn | 100 | 75 | 86,79 | 100 | 78,57 | 88,68 | 100 | 76,79 | 87,74 |
| Průměr | 95,32 | 80,75 | 87,75 | 94,96 | 85,24 | 89,94 | 95,37 | 85,05 | 90,04 |

*Tabulka 13: vliv selekce rysů: MI 80 %*

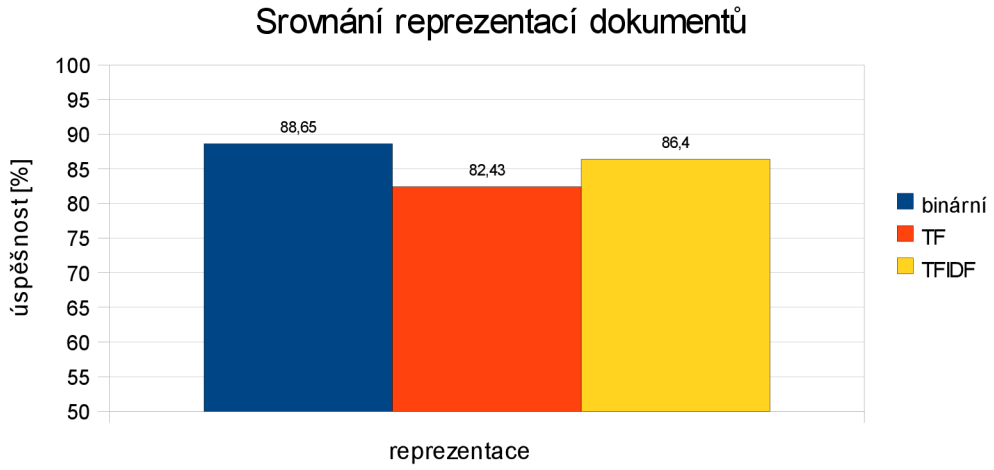
| Kategorie | text svm – lineární | | | svm light – lineární | | | svm light – RBF | | |
|---|---|---|---|---|---|---|---|---|---|
| | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] | P [%] | O [%] | Ú [%] |
| Earn | 98,03 | 66,33 | 82,5 | 95,74 | 75 | 85,83 | 93,55 | 77,33 | 86 |
| Acquisition | 96,8 | 80,66 | 89 | 97,33 | 85 | 91,33 | 97,33 | 85 | 91,33 |
| Money-fx | 65,94 | 92,63 | 72,89 | 86,99 | 77,91 | 83,43 | 86,99 | 77,91 | 83,43 |
| Grain | 100 | 71,23 | 85,37 | 100 | 82,88 | 91,29 | 98,43 | 85,62 | 91,99 |
| Crude | 97,64 | 68,13 | 81,63 | 97,47 | 84,62 | 90,36 | 94,58 | 86,26 | 89,76 |
| Trade | 54,81 | 97,44 | 61,2 | 74,66 | 93,16 | 82 | 74,83 | 94,02 | 82,4 |
| Interest | 67,27 | 88,1 | 71,95 | 75,81 | 74,6 | 74,8 | 73,28 | 76,19 | 73,58 |
| Ship | 100 | 46,59 | 69,68 | 97,5 | 88,64 | 92,26 | 94,19 | 92,05 | 92,26 |
| Wheat | 96,72 | 83,1 | 90,41 | 95,52 | 90,14 | 93,15 | 94,2 | 91,55 | 93,15 |
| Corn | 100 | 57,14 | 78,38 | 97,62 | 73,21 | 85,59 | 97,73 | 76,79 | 87,39 |
| Průměr | 87,72 | 75,14 | 78,3 | 91,86 | 82,52 | 87 | 90,51 | 84,27 | 87,13 |

*Tabulka 14: vliv selekce rysů: MI 50 %*

### 6.2.3 Shrnutí

V první části výsledků se ukázalo, že metoda SVM je jednou z nejlepších klasifikačních metod, je vidět převaha nad ostatními klasifikátory, které ve všech testech zaostávaly v úspěšnosti o několik procent. Zvláště SVM light podalo vynikající výsledky, navíc podpořené faktem, že doba trénování se po většinu experimentů pohybovala pod hranicí jedné sekundy. Dokonce i při použití nelineární jádrové funkce doba trénování nepřesáhla několik sekund.

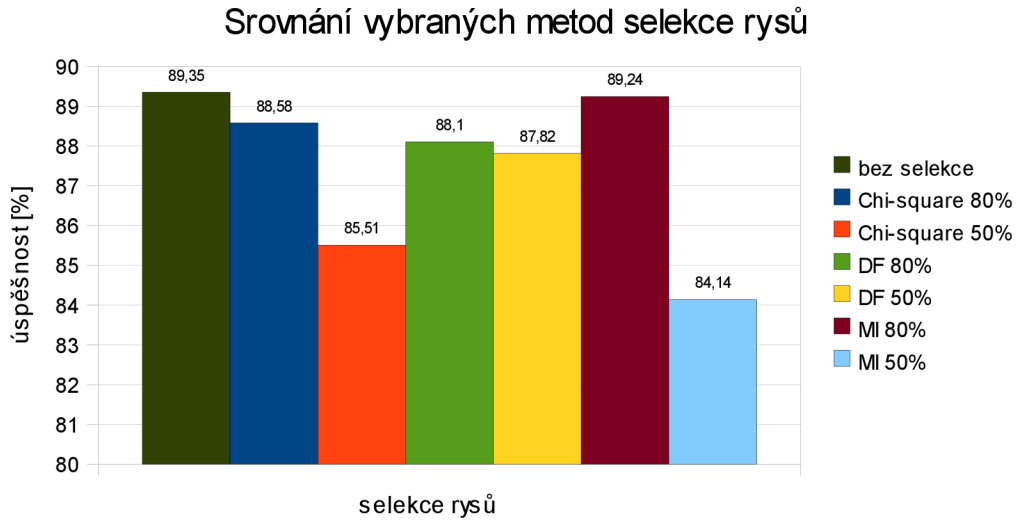
Srovnání různých reprezentací dokumentů dopadlo poměrně překvapivě. Ačkoli se obecně TFIDF reprezentace považuje za nejlepší, na následujícím grafu (obrázek 21) lze jasně vidět, že nejlépších výsledků klasifikátory dosáhly s binární reprezentací. Není ale možné z toho vyvozovat nějaké obecné závěry, jelikož testování proběhlo jen na datasetu Reuters, takže na jiných datasetech může být dosaženo opačného výsledku.




*Obrázek 21: srovnání různých reprezentací dokumentů*

Obecně se soudí, že selekce rysů by neměla znatelně zhoršit úspěšnost klasifikace a v některých případech ji dokonce vylepšit. Její význam je spíš ve zrychlení procesu učení i samotné klasifikace. Otázkou ovšem zůstává, jakou metodu selekce rysů zvolit a s jakou razancí selekci provést. Graf (obrázek 22) zachycuje porovnání různých hodnot selekce rysů. Jednotlivé sloupce jsou průměrné hodnoty úspěšnosti použitých klasifikátorů.

Při zachování 80 % rysů došlo k mírnému snížení úspěšnosti, nejlépe z testu vychází metoda selekce založená na výpočtu mutual information. Ovšem při selekci 50 % rysů úspěšnost klesá citelně s výjimkou selekce podle document frequency. Předpoklad, že by se úspěšnost klasifikace mohla zlepšit, se potvrdil ojediněle u některých kategorií – příkladem je kategorie wheat, kde došlo ke zlepšení až o několik procent.




*Obrázek 22: vliv selekce rysů na úspěšnost klasifikace*

Pro posouzení celkové úspěšnosti této práce je uvedena tabulka s již existujícím srovnáním klasifikačních metod na datasetu Reuters:

| | Findsim [%] | Naive Bayes [%] | BayesNets [%] | Trees [%] | Linear SVM [%] |
|---|---|---|---|---|---|
| earn | 92,9 | 95,9 | 95,8 | 97,8 | 98 |
| acq | 64,7 | 87,8 | 88,3 | 89,7 | 93,6 |
| money-fx | 46,7 | 65,6 | 58,8 | 66,2 | 74,5 |
| grain | 67,5 | 78,8 | 81,4 | 85 | 94,6 |
| crude | 70,1 | 79,5 | 79,6 | 85 | 88,9 |
| trade | 65,1 | 63,9 | 69 | 72,5 | 75,9 |
| interest | 63,4 | 64,9 | 71,3 | 67,1 | 77,7 |
| ship | 49,2 | 85,4 | 84,4 | 74,2 | 85,6 |
| w heat | 68,9 | 69,7 | 82,7 | 92,5 | 91,8 |
| corn | 48,2 | 65,3 | 76,4 | 91,8 | 90,3 |
| Průměr | 63,67 | 75,68 | 78,77 | 82,18 | 87,09 |

*Tabulka 15: porovnání klasifikátorů (tabulka převzata z [7]).*

Tabulka srovnává úspěšnost klasifikačních metod rovněž na deseti nejčastějších kategoriích datasetu Reuters. Z uvedených klasifikačních metod (v tabulce 15) lze porovnat s provedenými testy pouze dvě – naivní Bayesův klasifikátor a lineární SVM. Pro lepší představu je potřeba uvést i způsob předzpracování, který autoři zvolili. Původní popis lze nalézt v [7]. Dokumenty byly rovněž rozděleny na testovací a trénovací podle implicitního dělení ModApte, dále byla vyloučena slova, která se vyskytují pouze v jednom dokumentu a provedena další selekce rysů, tentokrát na základě Mutual Information, kdy bylo vybráno pouze 300 nejlepších slov pro každou kategorii. Byla zvolena binární reprezentace dokumentů, tedy budeme srovnávat s výsledky v kapitole 6.2.1.1 v tabulkách 2 a 3.

Na první pohled je patrné, že výsledky jsou v jednotlivých kategoriích dosti nevyrovnané, jako by se autoři zaměřili na co nejlepší úspěšnost v kategorii earn, kde je výsledek opravdu vynikající. Naproti tomu ve třech kategoriích klesla úspěšnost pod 80 % (kategorie money-fx jen 74,5 %). Ve výsledcích získaných v rámci této práce je vidět větší vyrovnanost. Sice si nejlepší kategorie vedou o několik procent hůře (earn 93,7 %, acquisition 91 %), ale nejhorším dosaženým výsledkem je 83,61 % u kategorie interest. Přes hranici 90 % se v obou srovnávaných experimentech dostalo pět kategorií. Průměrná hodnota úspěšnosti na těchto deseti nejčastějších kategoriích činí 87,09 % v tabulce 15 resp. 89,76 % v tabulce 2. Pokud porovnáme výsledky naivních Bayesovských klasifikátorů, je na tom lépe klasifikátor z nástroje Weka, a to v průměru dokonce o téměř 9 %. Jelikož ale nejsou známy bližší informace o Bayesovském klasifikátoru z tabulky 15, je toto srovnání jen ilustrativní.

# 7 Závěr

Cílem této diplomové práce bylo prostudovat techniky dolování v textových datech, především kategorizaci dokumentů, a klasifikační metodu Support Vector Machine, která se v této oblasti stále častěji využívá. Dále navrhnout, implementovat a otestovat aplikaci pro klasifikaci dokumentů pomocí této metody. Jako zdroj testovacích dat posloužil dataset zpráv agentury Reuters, který se stal pro výzkum v oboru kategorizace dokumentů jakýmsi nepsaným standardem.

Celkově se metoda SVM ukázala jako dobrou volbou pro klasifikaci dokumentů pro svoji rychlost a poměrně vysokou úspěšnost. Potvrdil se předpoklad, že SVM klasifikátor předstihl ostatní testované klasifikační metody v úspěšnosti o několik procent. Ačkoli tato skutečnost nebyla řádně testována a zdokumentována, vykazoval SVM klasifikátor oproti ostatním metodám lepší výsledky i v rychlosti trénování a klasifikace. Lineární a nelineární varianta dávají přibližně stejné výsledky, což je způsobeno tím, že prostor rysů má už tak dost vysokou dimenzi a výhoda mapování do jiného prostoru se tak příliš neprojeví. Navíc lineární SVM je výpočetně nenáročnější.

Případné vylepšení implementovaného nástroje pro klasifikaci dokumentů by se týkalo v první řadě lepšího vyřešení nelineárního klasifikátoru, jehož úspěšnost je prozatím na rozsáhlých datech slabá. Druhým cílem by byla podpora dalších datasetů, které se pro testování kategorizace dokumentů používají a realizace důkladnějších metod v oblasti předzpracování, které má patrně na výsledek klasifikace zásadní vliv. Zajímavým rozšířením by byla například detekce různých slovních spojení s využitím slovníku nebo implementace některé z metod extrakce rysů.

# Literatura


[1] Feldman, R., Sanger, J.: The Text Mining Handbook. Cambridge University Press, 2007.

[2] Krontorád, J.: Implementace algoritmu SVM v FPGA, diplomová práce, FIT VUT v Brně, 2009

[3] Zendulka, J. a kol.: Získávání znalostí z databází – studijní opora. FIT VUT v Brně, 2006.

[4] Gunn, S.R.: Support Vector Machine for Classification and Regression. University of Southampton, 1998

[5] Platt, J.C.: Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Dokument dostupný na URL http://research.microsoft.com/en-us/um/people/jplatt/smoTR.pdf (květen 2010)

[6] Burges, Ch.: A Tutorial on Support Vector Machines for Pattern Recognition, 1998. Dokument dostupný na URL http://www.umiacs.umd.edu/~joseph/support-vector-machines4.pdf (květen 2010)

[7] Hearst M. A.: Support Vector Machines. Dokument dostupný na URL http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/hearst98-SVMtutorial.pdf (květen 2010)

[8] Simeon M., Hilderman R.: Categorical Proportional Difference: A Feature Selection Method for Text Categorization. Dokument dostupný na URL http://crpit.com/confpapers/CRPITV87Simeon.pdf (květen 2010)

[9] Lewis, D. D.: Readme k Reuters 21578. Dokument dostupný na URL http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt (květen 2010)

[10] Janoušek V.: Machine Learning and Neural Networks – materiály k přednášce, 2007

[11] Porter, M.: The Porter Stemming Algorithm. Dokument dostupný na URL http://tartarus.org/martin/PorterStemmer (květen 2010)

[12] The University of Waikato: Weka Machine Learning Project. Dokument dostupný na URL http://www.cs.waikato.ac.nz/~ml/index.html (květen 2010)

[13] Joachims T.: SVM light. Dokument dostupný na URL http://svmlight.joachims.org (květen 2010)

# Seznam příloh

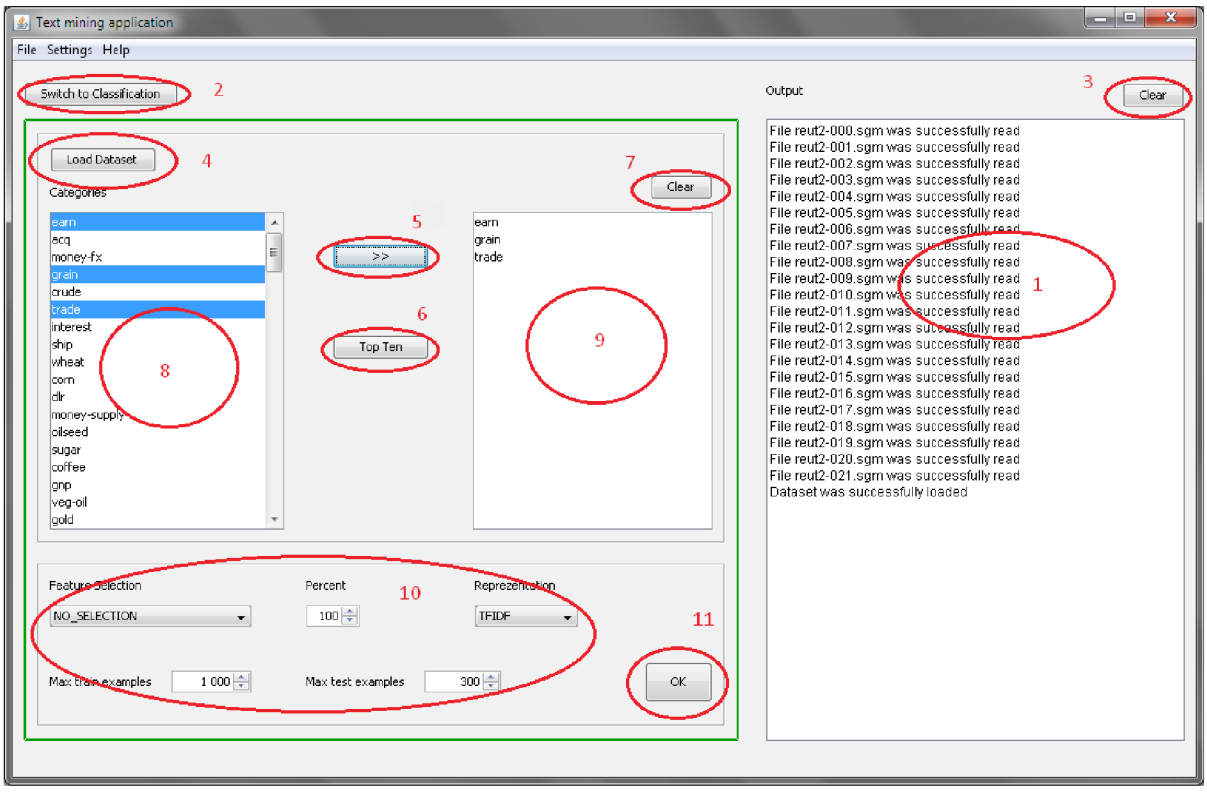
# Příloha 1

TEXT SVM

NÁSTROJ PRO KATEGORIZACI DOKUMENTŮ

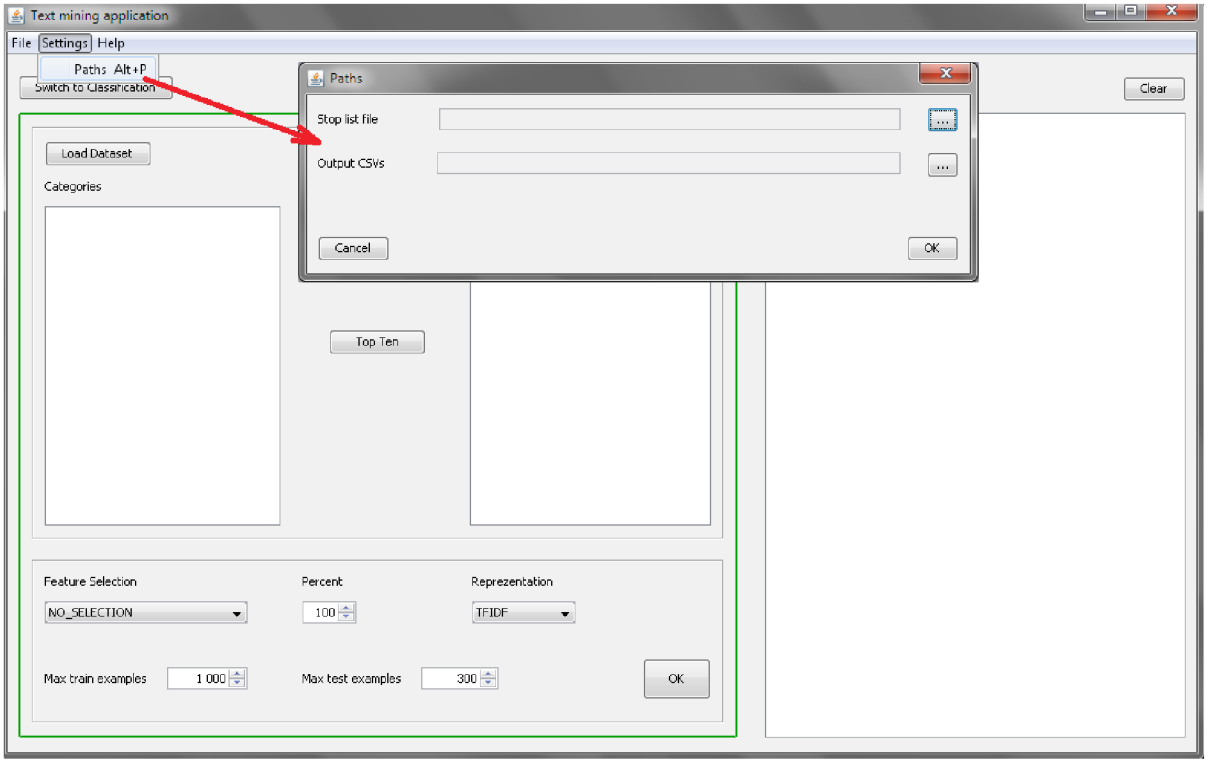
NÁVOD K OBSLUZE

# Předzpracování




1. Konzole pro komunikaci s uživatelem
2. Tlačítko pro přepnutí do režimu klasifikace
3. Tlačítko pro smazání výpisu
4. Tlačítko pro načtení datasetu
5. Tlačítko pro výběr kategorií
6. Tlačítko pro výběr deseti nejčastěji se vyskytujících kategorií
7. Tlačítko pro vymazání vybraných kategorií
8. Seznam načtených kategorií
9. Seznam vybraných kategorií
10. Nastavení parametrů předzpracování
11. Tlačítko pro spuštění předzpracování

## Nastavení seznamu stop slov a výstupu předzpracování




V menu Settings vyberte položku Paths, zobrazí se dialog, ve kterém zadáte cestu k seznamu stop slov a cestu ke složce, kam budou uloženy výstupní soubory předzpracování.

## Načtení datasetu a výběr kategorií

Po stisknutí tlačítka (4) se zobrazí dialog pro výběr souborů datasetu. Po vybrání začne jeho načítání. Jakmile tento proces skončí, ze seznamu (8) přesuňte pomocí tlačítek (5,6) do seznamu (9) kategorie dokumentů, které chcete předzpracovat. Tlačítkem (7) lze výběr zrušit.

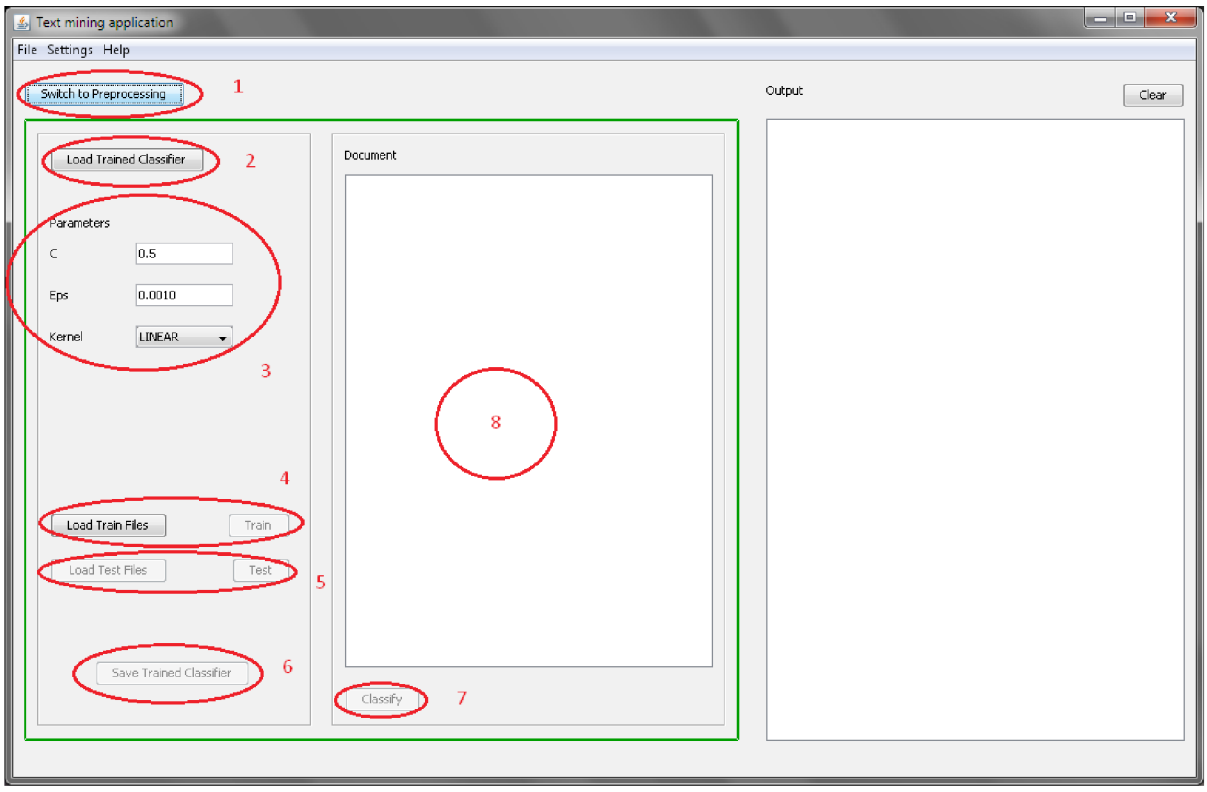
## Nastavení parametrů předzpracování

Nastavte parametry předzpracování (10):

- vyberte metodu selekce rysů a množství rysů, které zůstanou zachovány (v procentech)
- vyberte reprezentaci dokumentů
- zvolte maximální velikosti trénovacího a testovacího souboru

Předzpracování začne po stisku tlačítka (11). Výstupní soubory naleznete ve složce, kterou jste zadali (viz předchozí kapitola návodu).

# Klasifikace




1. Tlačítko pro přepnutí do režimu předzpracování
2. Tlačítko pro načtení naučeného klasifikátoru
3. Nastavení parametrů klasifikace
4. Nastavení trénovacích souborů a spuštění trénování
5. Nastavení testovacích souborů a spuštění testování
6. Uložení naučeného klasifikátoru
7. Klasifikace samotného dokumentu
8. Textové pole pro zadání obsahu dokumentu

## Nastavení parametrů

Zadejte hodnoty parametrů C a epsilon a vyberte jádrovou funkci (3). Po výběru nelineární funkce se zobrazí pole pro zadání dalšího parametru (stupeň u polynomiální funkce a sigma u RBF funkce).

## Výběr trénovacích a testovacích dat

Pomocí tlačítek (4) vyberte trénovací soubory a spusťte proces učení (tlačítko pro spuštění učení se povolí až po výběru trénovacích souborů). Jakmile skončí proces učení, povolí se tlačítko pro výběr testovacích souborů a po jejich výběru i tlačítko pro spuštění testování (5).

## Uložení / načtení naučeného klasifikátoru

Tlačítkem (2) lze načíst již naučený klasifikátor ze souboru. V takovém případě lze rovnou přistoupit k testování. Pomocí tlačítka (6) lze naučený klasifikátor uložit do souboru.

## Klasifikace samotného dokumentu

Tato funkce je dostupná až po načtení datasetu v části předzpracování a po naučení klasifikátoru. Do pole (8) zadejte text dokumentu a stiskněte tlačítko (7). Na výstupní konzoli se zobrazí kategorie, do které byl zadaný dokument přiřazen.