VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLASIFIKACE DOKUMENTŮ PODLE TÉMATU

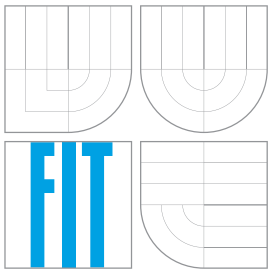
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ MAREK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLASIFIKACE DOKUMENTŮ PODLE TÉMATU

DOCUMENT CLASSIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ MAREK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUBOMÍR OTRUSINA

BRNO 2013

Abstrakt

Tato práce se zabývá problematikou klasifikace textových dokumentů, a to především metodami klasifikace textu. Hlavním cílem této práce je rozebrat dva algoritmy pro klasifikaci dokumentů, implementovat je a následně porovnat. Byl zvoleny algoritmy Bayesovského klasifikátoru a klasifikátoru založeného na metodě support vector machines (SVM), které jsou v této práci podrobně analyzovány a popsány. Jedním z cílů této práce bylo optimálně vytvořit a vybrat příznaky, které by co nejvíce napomohly klasifikaci textu. V závěru práce je provedeno množství testů, ukazujících účinnost obou klasifikátorů za různých podmínek.

Abstract

This thesis deals with a document classification, especially with a text classification method. Main goal of this thesis is to analyze two arbitrary document classification algorithms to describe them and to create an implementation of those algorithms. Chosen algorithms are Bayes classifier and classifier based on support vector machines (SVM) which were analyzed and implemented in the practical part of this thesis. One of the main goals of this thesis is to create and choose optimal text features, which are describing the input text best and thus lead to the best classification results. At the end of this thesis there is a bunch of tests showing comparison of efficiency of the chosen classifiers under various conditions.

Klíčová slova

klasifikace, Bayesovský klasifikátor, Bayesovský teorém, SVM klasifikátor, SVM, jaderné metody, volba parametrů SVM, simulované žíhání, klasifikace přirozeného jazyka, strojové učení, zpracování přirozeného jazyka, příznaky

Keywords

classification, Bayes classifier, Bayes' theorem, SVM classifier, SVM, kernel methods, choosing SVM classifier parameters, simulated annealing, natural language classification, machine learning, natural language processing, features

Citace

Tomáš Marek: Klasifikace dokumentů podle tématu, diplomová práce, Brno, FIT VUT v Brně, 2013

Klasifikace dokumentů podle tématu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Lubomíra Otrusiny.

.....
Tomáš Marek
19. května 2013

Poděkování

V prvé řadě bych chtěl poděkovat panu Ing. Lubomíru Otrusinovi za vedení této práce a podnětné připomínky a návrhy, které mi poskytl. Dále pak bych chtěl poděkovat všem členům své rodiny za podporu a porozumění při psaní této práce. Zejména pak Kubovi, který nikdy neřekl ne, když jsem potřeboval konzultaci s nějakým matematickým problémem a své mamce, která mi moc pomohla s konečnými úpravami jazyka (a bylo to věru potřeba).

Obsah

Kapitola 1

Úvod

Klasifikace dokumentů podle tématu je jednou z úloh oboru *zpracování přirozeného jazyka (Natural Language Processing – NLP)*. Zpracování přirozeného jazyka je oborem aplikace výpočetních modelů pro řešení úkolů, které se určitým způsobem vztahují k textu libovolného přirozeného jazyka. Historie zpracování přirozeného jazyka začíná v padesátých letech 20. století a od té doby doznal obor zpracování přirozeného jazyka díky vysokému nárůstu výpočetního výkonu velkých změn. V současné době patří mezi nejčastěji používané metody zpracování přirozeného jazyka statistické metody a metody *strojového učení (machine learning*).

Významnou úlohou řešenou v oboru zpracování přirozeného jazyka je klasifikace textu. Tato úloha spadá do výše zmíněné skupiny statistických metod a metod strojového učení. Metody klasifikace textu lze použít pro velké množství úloh, které se týkají zpracování přirozeného jazyka, a to především díky jejich flexibilitě a již velmi dobře zpracované teorii.

V této práci se nejprve zabýváme obecnými přístupy, používanými pro řešení úloh zpracování přirozeného jazyka a následně představujeme dvě zvolené klasifikační metody – jednu probabilistickou a jednu neprobabilistickou.

Tou první, probabilistickou metodou, která je v této práci podrobně popsána, je metoda nazývaná *Bayesovský klasifikátor*, využívající ke klasifikaci textu Bayesova teorému. Bayesovský klasifikátor, nebo také Bayesovský filtr, jak je občas nazýván, je v praxi často používán e-mailovými klienty pro zjišťování, zda je e-mailová zpráva vyžádanou, či nevyžádanou poštou.

Druhou zvolenou metodou je neprobabilistická klasifikační metoda *SVM (support vector machines)*, jež pomocí namapování vstupního textu do n-dimenzionálního prostoru a následným rozdělením tohoto prostoru na dva poloprostory dokáže vstupní text klasifikovat.

Obě metody jsou v této práci podrobně rozebrány, a to jak z teoretického, tak z praktického hlediska.

Součástí této práce je mimo teoretické části také část praktická a část implementační. Praktická část se nachází ve druhé polovině tohoto dokumentu a popisuje implementaci a přístupy k řešení problémů, které nastaly při vytváření implementační části. Také se v ní nacházejí výsledky testů, které byly nad implementovaným programem provedeny a jejich interpretace. V rámci implementační části byl vytvořen samotný program, v němž jsou implementovány obě výše zmíněné klasifikační metody. Součástí praktické části je také množina trénovacích a testovacích dat, která jsou potom používána v testech.

V závěru této práce jsou prezentovány výsledky, získané porovnáním obou výše zmíněných implementovaných klasifikátorů a je popsán vliv optimální volby příznaků na klasifikační schopnosti implementovaných klasifikátorů.

1.1 Zadání práce a motivace

Zadáním této práce bylo seznámit se s problematikou klasifikace dokumentů dle tématu a zvolit si dvě metody, kterými se budu podrobněji zabývat. Následně tyto dvě metody analyzovat a implementovat program, který pomocí těchto metod bude klasifikovat vstupní dokumenty do určitých tříd. U obou metod jsem měl dle zadání dbát na výběr vhodných příznaků z textu.

Pro potřeby této práce jsem anotoval velkou sadu dokumentů, které budu následně používat pro trénování a testování mnou vytvořeného programu. Na těchto datech následně porovnám dvě zvolené metody klasifikace pomocí standardních metrik pro hodnocení klasifikátorů.

Téma diplomové práce jsem si zvolil, jelikož jsem téměř dva roky pracoval na projektu M-Eco[1] ve skupině NLP na VUT FIT, kde jsem se zabýval klasifikací tweetů[2]. Praktické využití Bayesovského klasifikátoru při této práci mne motivovalo k tomu, abych se klasifikačními metodami zabýval dále. Této práce bych proto chtěl využít k tomu, abych prohloubil své znalosti o problematice klasifikace dokumentů a porozumění klasifikačním metodám. V rámci klasifikačních metod se chci zaměřit především na klasifikátor založený na SVM, jelikož metoda SVM je velmi dobře variabilní a lze ji použít pro řešení velkého množství různých klasifikačních, ale například i regresních úkolů.

1.2 Návaznost práce na semestrální projekt

Základ této práce byl rozpracován v semestrálním projektu [19], jenž byl vytvořen v rámci stejnojmenného předmětu na FIT VUT.

Semestrální projekt obsahoval úvodní kapitolu o problematice zpracování přirozeného jazyka a popis problematiky Bayesovského klasifikátoru, jenž byl pro tuto práci také implementován (viz kapitola 3). Byla vytvořena malá testovací sada pro trénování a testování tohoto klasifikátoru. Na této datové sadě byly pro Bayesovský klasifikátor následně provedeny testy, v nichž byl zhodnocen vliv zvolených příznaků (viz kapitola 5) na klasifikační schopnosti implementovaného Bayesovského klasifikátoru. V semestrálním projektu ještě nebyly implementovány speciální příznaky (viz kapitola 5.1).

[1] http://www.meco-project.eu/

[2] Tweety – příspěvky zveřejňované na sociální síti Twitter (http://twitter.com/)

Kapitola 2

Úvod do klasifikace dokumentů

V této kapitole se budeme stručně zabývat historií (viz kapitola 2.1) oboru *zpracování přirozeného jazyka (natural language processing, dále též NLP)*, poté hlavními úkoly, které se v oboru zpracování přirozeného jazyka řeší (viz kapitola 2.2), dále popíšeme typické klasifikační úlohy (viz kapitola 2.3) a na závěr se budeme věnovat problematice klasifikace dokumentů, jež je hlavním tématem této práce.

2.1 Historie

Obor zpracování přirozeného jazyka se vyvíjí souběžně s historií vývoje výpočetní techniky. Po roce 1945, kdy společensko-politická situace ve světě umožnila změnu hlavních dosavadních směrů výzkumu v oblasti výpočetní techniky od původně převážně vojenského využití (např. kryptografie, kryptoanalýza, atd.) k dalším vědním oborům. Díky tomu se začal rozvíjet také obor zpracování přirozeného jazyka.

Jedním z prvních významných milníků v historii zpracování přirozeného jazyka byl článek Alana Turinga s názvem „Computing Machinery and Intelligence“ [24], v němž Turing publikoval takzvaný *Turingův test* jakožto kritérium inteligence počítačových programů. Aby počítačový program prošel *Turingovým testem*, nesmí nestranný soudce poznat z obsahu konverzace mezi programem a člověkem (konverzace probíhá v reálném čase), která strana je která.

Jednou z prvních řešených úloh v oboru zpracování přirozeného jazyka bylo vytvoření automatických překladačů, tedy programů, které bez lidského přispění dokážou přeložit vstupní text z jednoho přirozeného jazyka do druhého. Uspokojivé řešení tohoto úkolu však v té době nebylo nalezeno, a ani v roce 1966 ještě výzkumníci nebyli nikterak blízko k vyvinutí takového softwaru. V roce 1966 proto vydal americký vládní výbor „The Automatic Language Processing Advisory Committee“ (ALPAC) zprávu shrnující dosavadní výsledky výzkumu a konstatující, že: „Nebyl vyvinut žádný strojový překladač obecných vědeckých textů a žádný takový překladač nebude vyvinut ani v blízké budoucnosti“. Takto formulovaný závěr zprávy zapříčinil výrazné škrty v rozpočtech vědeckých týmů, zabývajících se výzkumem zpracování přirozeného jazyka a překladačů.

Do 80. let 20. století používala drtivá většina systémů pro zpracování přirozeného jazyka velmi složitá ručně zadávaná pravidla. V 80. letech však byly poprvé představeny metody strojového učení, které díky stále rostoucímu výkonu výpočetní techniky umožňovaly generovat tato pravidla automaticky a dosahovat tak při zpracování přirozeného jazyka stále lepších výsledků. S nástupem strojového učení se odvětví zpracování přirozeného jazyka za-

čalo zaměřovat na statistické metody, jež k řešení problémů přistupovaly jinak, než metody dosavadní. Jejich hlavním principem byla práce s pravděpodobnostními modely a váhami jednotlivých rozhodnutí. Tímto způsobem bylo možné efektivněji řešit většinu NLP úkolů. Vzhledem k tomu, že obor zpracování přirozeného jazyka pracuje s daty vytvořenými člověkem, jež mohou obsahovat různé chyby, pracují statistické metody značně spolehlivěji než dřívější ručně psaná pravidla.

V současné době je výzkum zpracování přirozeného jazyka orientován především na vytvoření autonomních a semiautonomních učících algoritmů, tedy algoritmů, schopných učit se z dat, která předtím nebyla ručně anotována, a nebo z kombinace anotovaných a neanotovaných dat.

2.2 Hlavní úkoly řešené v NLP

Obor zpracování přirozeného jazyka je velmi rozsáhlý a existuje v něm mimo klasifikace textu velké množství dalších úloh k řešení. Nyní velmi stručně popíšeme několik hlavních úkolů, řešených v oboru zpracování přirozeného jazyka:

- *Automatická sumarizace* – úkolem automatické sumarizace je redukovat vstupní text nebo sadu textů do několika slov nebo krátkého odstavce, popisujícího sémantický obsah vstupního textu.
- *Generování přirozeného jazyka* – generování přirozeného jazyka má vytvořit výstup v přirozeném jazyce z interní reprezentace v počítači.
- *Odpovídání na otázky* – v oblasti odpovídání na otázky se programátoři pokoušejí vytvořit program, který by dokázal korektně odpovědět na uživatelem zadanou vstupní otázku formulovanou v přirozeném jazyce.
- *Odstraňování víceznačnosti slov v textu* – odstraňování víceznačnosti slov v textu je významnou úlohou, napomáhající správnému porozumění textu. Víceznačná slova jsou v textu identifikována a poté je vyhledáván jejich správný význam v kontextu vstupního textu.
- *Strojový překlad* – je úloha zpracování přirozeného jazyka, která převádí vstupní text v určitém vstupním jazyce na výstupní text v jazyce jiném.
- *Klasifikační úlohy* – mají za úkol libovolné vstupy přiřadit do předem určených tříd. Tyto třídy mohou být buď dvě (takzvaná binární klasifikace), nebo jich může být více (takzvaná multilabel klasifikace).

2.3 Klasifikační úlohy

V této části kapitoly se budeme podrobněji zabývat následujícími dvěma klasifikačními úlohami:

- Klasifikace tématu
- Filtrování spamu

2.3.1 Klasifikace tématu

Klasifikace tématu je úloha přiřazování názvů témat ke vstupním textovým dokumentům. Typicky daný vstupní text pokrývá větší množství témat. Metody pro klasifikaci tématu mohou být založeny například na skrytých Markovových modelech. Výstupem klasifikátoru tématu pro vstupní text bývá velmi často kromě seznamu tříd témat, kterých se zřejmě vstupní text týká, také seznam pravděpodobností definujících míru náležitosti do těchto jednotlivých tříd.

2.3.2 Filtrování spamu

Zájem o klasifikační problém filtrování spamu v posledních letech velmi výrazně vzrostl, a to především kvůli množství nevyžádané pošty (spamu), kterou uživatelé dostávají do svých e-mailových schránek. Jsou dvě možnosti jak úloha filtrování spamu může fungovat. Buď jsou zprávy filtrovány na základě obsahu (ať už textového nebo jiného), nebo na základě metainformací v hlavičce zprávy. Metody zabývající se filtrováním na základě obsahu e-mailu velmi často využívají toho, že většina zpráv obsahuje nějakou textovou informaci. Podle ní potom klasifikují, zda je zpráva nevyžádanou poštou.

V této práci se budeme zabývat metodami klasifikace textových dokumentů, které tím, že označují relevantní a nerelevantní textové vstupy vzhledem k danému tématu, mohou být s výhodou využity i pro filtrování spamu.

2.4 Klasifikace dokumentů

Jedním z úkolů této práce je získat přehled o klasifikaci textových dokumentů, seznámit se s metodami klasifikace a aplikovat je na vytvořenou datovou sadu. V této části textu bude podrobně popsána problematika klasifikace dokumentů.

2.4.1 Definice klasifikace

Definice: Klasifikace je činnost, která rozděluje objekty do tříd (kategorií) podle jejich společných vlastností.

Třída v kontextu zpracování přirozeného jazyka je množina objektů, vyznačujících se určitou společnou vlastností nebo vlastnostmi, která/-ré danou třídu popisuje/-jí. Klasifikace je potom činnost, která přiřazuje objekty do daných tříd. Nadále se v této práci budeme zabývat klasifikací textu.

Obecně mějme tedy prostor objektů X a množinu tříd Y. Operace klasifikace potom odpovídá klasifikační funkci f:

$$f : X \to Y. \tag{2.1}$$

Klasifikační funkce f tedy přiřadí jeden objekt z množiny objektů právě do jedné třídy. Může ovšem nastat situace, kdy jeden objekt odpovídá kritériím pro zařazení do více tříd. Například budeme klasifikovat textovou zprávu, jež může z hlediska obsahu zapadnout do více tříd, například do třídy osobních zkušeností (pisatel píše o vlastní zkušenosti) a do třídy relevantní k tématu zdraví (*„Bolí mě hlava.“*). V tomto případě musíme modifikovat klasifikační funkci f následovně:

$$f : X \to 2^Y, \tag{2.2}$$

kde 2^Y označuje potenční množinu všech tříd. Tato forma klasifikace se také označuje jako *multi-label klasifikace*. Mimo multi-label klasifikace ovšem existuje druhá forma klasifikace – takzvaná *binární klasifikace*, u které se text klasifikuje právě do dvou tříd. V této práci se nadále budeme zabývat právě binární klasifikací.

Při klasifikaci se ke každé třídě může přiřadit reálné číslo $p \in < 0, 1 >$, které definuje míru náležitosti klasifikovaného objektu do přiřazených tříd. Tomuto se také říká *soft klasifikace*. Naproti tomu, když je při klasifikaci přiřazen objekt do třídy „napevno“ (ano, patří tam / ne, nepatří tam), pak tento způsob klasifikace nazýváme *hard klasifikace*.

2.4.2 Klasifikační přístupy

Existují dva hlavní přístupy při řešení klasifikačních úloh:

- Probabilistické
- Neprobabilistické

Probabilistické klasifikátory

Jedním ze způsobů, jakým lze vytvořit funkční klasifikátor, je využít teorie pravděpodobnosti. Klasifikátory fungující na bázi pravděpodobnostních výpočtů určují pravděpodobnosti, se kterými daný objekt spadá do některé třídy. Do které třídy respektive kterých tříd objekt spadá, je následně určeno pomocí předem definovaného *prahu (threshold)*.

Mezi probabilistické klasifikační metody patří například metoda založená na Bayesově teorému – Bayesovský klasifikátor. Bayesův teorém a jeho použití pro klasifikaci je podrobně popsán v kapitole 3.

Mimo Bayesovského klasifikátoru samozřejmě existují další probabilistické klasifikátory, například Fuzzy klasifikátory [4], nebo klasifikační stromy [5], jimiž se v této práci ale nebudeme zabývat.

Neprobabilistické klasifikátory

Kromě probabilistických metod existují také neprobabilistické metody klasifikace. Neprobabilistické klasifikátory většinou pracují tak, že vymodelují klasifikační funkci a nevyužívají pro zařazení objektů do tříd pravděpodobnostní výpočty. Asi nejznámější neprobabilistickou metodou je metoda SVM (support vector machines), která vytváří takovou nadrovinu v prostoru příznaků, která bude rozdělovat trénovací data. Ideální nadrovina rozděluje data z trénovací množiny tak, že body v prostoru leží v opačných poloprostorech a vzdálenosti všech bodů od roviny jsou co největší. Mimo SVM klasifikátoru existují další neprobabilistické klasifikátory, jako například klasifikátory založené na neuronových sítích – perceptron [11], či back propagation [27]. V této práci se však budeme podrobněji zabývat pouze metodou SVM, která bude dále podrobněji popsána (viz kapitola 4).

2.4.3 Klasifikace a shlukování

Ještě relativně nedávno byla klasifikace chápána jako podmnožina úlohy nazývané shlukování. Je sice pravda, že klasifikace a shlukování k sobě mají velmi blízko a také mnoho technik používaných v klasifikaci je možné použít i ve shlukování, nicméně rozdíl mezi těmito dvěma úlohami je v tom, že při klasifikaci známe předem třídy, do kterých budeme

vstup klasifikovat. U shlukování tomu tak není. Shlukování dělí vstupní text podle významných společných příznaků ve vstupních datech. Z definice klasifikace (viz kapitola 2.4.1) víme, že klasifikační úloha je definována klasifikační funkcí f, která vstupnímu vzorku x podle jeho příznaků přiřadí označení třídy y, do níž vzorek spadá. Klasifikační úlohy v NLP se snaží tuto klasifikační funkci f co nejpřesněji aproximovat, aby simulovaly její výsledek. Naproti tomu shlukování nemá podobnou funkci jako klasifikace, která by určovala, jak má vypadat výsledek. Výsledná struktura tříd shlukování je vytvořena za běhu metody na základě znaků podobnosti vzorků z množiny vstupů x.

Strojové učení

Jelikož pro vytvoření funkčního klasifikátoru je nutné, aby klasifikátor byl naučen pomocí správné trénovací množiny dat, je strojové učení s tématem klasifikace velmi úzce spojeno. Strojové učení je jedním z podtémat oboru umělé inteligence, zabývající se vytvořením algoritmů, umožňujících počítačovým programům se učit. Algoritmy strojového učení se dělí do následujících tří kategorií:

- *Učení s učitelem (supervised learning)* – nejjednodušší způsob učení. Problematičnost tohoto přístupu spočívá v tom, že veškerá data, která se program naučí, musejí být ručně anotována člověkem, z čehož vyplývá jeho velká časová náročnost.
- *Učení bez učitele (unsupervised learning)* – při učení bez učitele je učícímu se programu předána sada trénovacích dat, ve které si učící se program sám hledá význačné vlastnosti, na jejichž základě poté vytvoří vlastní klasifikační funkci. Jedním z možných přístupů k řešení této metody je metoda shlukování (viz kapitola 2.4.3).
- *Přístup na pomezí metod učení s učitelem a učení bez učitele (semi-supervised learning)* – je metoda založená na učení pomocí vstupních trénovacích dat, kterých následně metoda využije k automatickému vytvoření dalších trénovacích dat.

Boosting

V roce 1988 vyslovil Michael Kearns v práci s názvem „Thoughts on hypothesis boosting“ [15] otázku, zda lze z množiny slabých klasifikátorů (takových, které mají nízké hodnoty korelace testovacích a výstupních dat) vytvořit jeden klasifikátor, který by měl výrazně lepší výsledky. Ve své práci [15] dokázal, že tomu tak opravdu může být. Od té doby bylo vyvinuto více algoritmů pro boosting klasifikace, nicméně většina z nich se zakládá na iterativním učení množiny slabých klasifikátorů a jejich následném sdružení do jednoho přesného klasifikátoru. Historicky nejvýznamnějším boostingovým algoritmem je zřejmě algoritmus *AdaBoost (adaptive boosting)*, vytvořený Yoavem Freundem a Robertem Schapirem, který publikovali v práci „A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting“[10].

Kapitola 3

Bayesovský klasifikátor

Bayesovský klasifikátor (naivní Bayesovský klasifikátor) je klasifikátor založený na zjednodušeném Bayesově teorému. Zjednodušení spočívá v tom, že existence nebo naopak neexistence jednoho příznaku (textového nebo speciálního – viz kapitola 5) není závislá na existenci nebo neexistenci jiného příznaku. Tudíž, i když existence několika příznaků na sobě závisí, Bayesovský klasifikátor s těmito příznaky pracuje jako se zcela nezávislými událostmi. Tato vlastnost Bayesovského klasifikátoru je nevýhodou pro klasifikování přirozeného jazyka především proto, že nedokáže pracovat se slovními spojeními a kontextem slov, což by mohlo napomoci klasifikaci. Ačkoliv neschopnost pracovat s kontextem slov Bayesovský klasifikátor omezuje, jeho využití na reálných textech se osvědčilo a je hojně používán. V posledních letech ale již existují jiné a lepší metody pro klasifikaci, například metoda *support vector machines (SVM)*, která je v této práci podrobně popsána (viz kapitola 4).

V následujícím textu bude představena teorie Bayesovského klasifikátoru pro binární klasifikaci textu, tedy pro klasifikaci do pozitivní a negativní třídy.

3.1 Matematický model

Bayesovský klasifikátor využívá Bayesova teorému, který zní následovně:

Věta: Mějme dva náhodné jevy A a B s pravděpodobnostmi $P(A)$ a $P(B)$, přičemž $P(B) > 0$. Potom platí:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}. \tag{3.1}$$

Důkaz: Dle podmíněných pravděpodobností platí, že pravděpodobnost dvou událostí A a B – $P(A \bigcap B)$ se rovná pravděpodobnosti A krát pravděpodobnost B za předpokladu, že nastalo A – $P(B|A)$.

$$P(P(A \cap B)) = P(A) \cdot P(B|A) \tag{3.2}$$

Dále také platí, že pravděpodobnost A a B se rovná pravděpodobnosti B krát pravděpodobnost A za předpokladu, že nastalo B:

$$P(P(A \cap B)) = P(B) \cdot P(A|B). \tag{3.3}$$

Z těchto dvou vztahů vychází:

$$P(B) \cdot P(A|B) = P(A) \cdot P(B|A). \tag{3.4}$$

Upravením této rovnice poté dostáváme:

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}, \tag{3.5}$$

což je Bayesův teorém. **Q.E.D.**

Pro klasifikaci textu není vhodné použít přímo rovnici z Bayesova teorému, jelikož by bylo třeba zapamatovat si pro každý příznak tři hodnoty a bylo by nutné provádět větší množství výpočtů, nicméně je možné rovnici upravit do tvaru, v němž je třeba si pro každý příznak pamatovat pouze jednu hodnotu pravděpodobnosti a také není třeba provádět tolik výpočtů. Upravená rovnice má následující tvar:

$$P = \frac{p_1 p_2 p_3 \dots p_N}{p_1 p_2 p_3 \dots p_N + (1-p_1)(1-p_2)(1-p_3)\dots(1-p_N)}, \tag{3.6}$$

kde P je pravděpodobnost určující míru náležitosti klasifikovaného textu do negativní třídy a $p_i, i = 1 \dots N$ udává tuto míru pro jednotlivé příznaky i. Výsledné P je potom porovnáno s určitým prahem, který definuje, zda oklasifikovaný text patří do negativní, nebo pozitivní třídy.

3.1.1 Odvození rovnice pro klasifikaci

Nyní přejděme k odvození rovnice 3.6 z Bayesova teorému 3.1.

Mějme X a Y, představující dva příznaky, S znamenající „patří do negativní třídy“ a $\neg S$ znamenající „patří do pozitivní třídy“ (=nepatří do negativní třídy). Pro zjednodušení budeme používat případ se dvěma příznaky.

$$P(S|X \cap Y) = \frac{P(X \cap Y|S) \cdot P(S)}{P(X \cap Y)}$$

Nyní využijeme toho, že platí:

$$P(B) = P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A),$$

tudíž:

$$P(S|X \cap Y) = \frac{P(X \cap Y|S) \cdot P(S)}{(P(S) * P(X \cap Y|S) + P(\neg S) \cdot P(X \cap Y|\neg S))}. \tag{3.7}$$

Nyní aplikujeme onu naivitu, kterou jsme zmiňovali výše (viz kapitola 3).

To znamená, že budeme předpokládat, že pravděpodobnost X je nezávislá na pravděpodobnosti Y, takže můžeme použít vztah:

$$P(A \cap B) = P(A) \cdot P(B)$$

Z toho vyplývá, že rovnici 3.7 můžeme upravit do tvaru

$$P(S|X \cap Y) = \frac{P(X|S) \cdot P(Y|S) \cdot P(S)}{P(S) \cdot P(X|S) \cdot P(Y|S) + P(\neg S) \cdot P(X|\neg S) \cdot P(Y|\neg S)}. \tag{3.8}$$

Vyjdeme opět z Bayesova teorému, který říká, že platí:

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

Toho využijeme a dosadíme do rovnice 3.8, dostaneme tedy:

$$P(S|X \cap Y) = \frac{\frac{P(S|X)\cdot P(X)}{P(S)} \cdot \frac{P(S|Y)\cdot P(Y)}{P(S)} \cdot P(S)}{P(S) \cdot \frac{P(S|X)\cdot P(X)}{P(S)} \cdot \frac{P(S|Y)\cdot P(Y)}{P(S)} + P(\neg S) \cdot \frac{P(\neg S|X)\cdot P(X)}{P(\neg S)} \cdot \frac{P(\neg S|Y)\cdot P(Y)}{P(\neg S)}}. \tag{3.9}$$

Z rovnice 3.8 můžeme odstranit $P(X)$ a $P(Y)$:

$$P(S|X \cap Y) = \frac{\frac{P(S|X)}{P(S)} \cdot \frac{P(S|Y)}{P(S)} \cdot P(S)}{P(S) \cdot \frac{P(S|X)}{P(S)} \cdot \frac{P(S|Y)}{P(S)} + P(\neg S) \cdot \frac{P(\neg S|X)}{P(\neg S)} \cdot \frac{P(\neg S|Y)}{P(\neg S)}}.$$

Po zjednodušení dostaneme:

$$P(S|X \cap Y) = \frac{\frac{P(S|X)\cdot P(S|Y)}{P(S)}}{\frac{P(S|X)\cdot P(S|Y)}{P(S)} + \frac{P(\neg S|X)\cdot P(\neg S|Y)}{P(\neg S)}}. \tag{3.10}$$

Nyní můžeme přistoupit k závěrečné úpravě, která ovšem předpokládá, že záznamy v trénovací množině jsou rovnoměrně rozloženy, tzn. že zhruba 50% naučených záznamů patří do negativní třídy a zhruba 50% do pozitivní třídy. Potom platí, že $P(S) \approx P(\neg S)$, a tak dostaneme z rovnice 3.10 rovnici následující:

$$P(S|X \cap Y) = \frac{P(S|X) \cdot P(S|Y)}{P(S|X) \cdot P(S|Y) + P(\neg S|X) \cdot P(\neg S|Y)}, \tag{3.11}$$

kde $P(\neg A) = 1 - P(A)$, tudíž $P(\neg A|B)$ můžeme přepsat na $1 - P(A|B)$. Z toho získáme rovnici 3.12 ekvivalentní s rovnicí 3.6:

$$P(S|X \cap Y) = \frac{P(S|X) \cdot P(S|Y)}{P(S|X) \cdot P(S|Y) + (1 - P(S|X)) \cdot (1 - P(S|Y))}. \tag{3.12}$$

Jestliže není trénovací množina vyvážená (množství záznamů náležících do pozitivní třídy a množství záznamů náležící do negativní třídy není rovnoměrné), je vhodné pro klasifikaci použít rovnici 3.10, protože při použití rovnice 3.12 může docházet ke zkreslení klasifikace.

3.2 Princip Bayesovského klasifikátoru

Bayesovský klasifikátor si do databáze při učení ukládá pravděpodobnosti s jakými jednotlivé příznaky získané ze vstupních textů náležejí do negativní třídy, tzn. ukládá si hodnotu $P(S|X)$. Po naučení dostatečně velkého objemu trénovacích dat potom využívá tyto pravděpodobnosti pro výpočet, zda klasifikovaný vstupní text náleží do pozitivní, nebo negativní třídy.

Klasifikace tedy probíhá tak, že vstupní text X se rozdělí na příznaky X_i. Klasifikátor se podívá do databáze a zjistí pravděpodobnost $P(S|X_i)$ se kterou jednotlivé příznaky náležejí do negativní třídy. Jestliže klasifikátor při klasifikaci textu narazí na příznak, který dosud není v jeho databázi, pak je mu přiřazena hodnota 0,5 (tzn. klasifikátor neumí určit s jakou pravděpodobností tento příznak náleží do pozitivní či negativní třídy). Pravděpodobnost $P(X)$ že vstupní text náleží do negativní třídy je potom vypočtena pomocí dosazení $P(S|X_i)$ do rovnice 3.12.

3.3 Příklad klasifikace

Uveďme si nyní jednoduchý příklad klasifikace textu: mějme vstupní větu *'Aaaa, my stomach hurts.'*. Tuto větu si klasifikátor rozdělí podle zadaného pravidla na seznam příznaků (v tomto případě pouze slov).

$$[' Aaaa',' my',' stomach',' hurts']$$

V databázi klasifikátoru máme například:

$$\{' my' : 0.6,' stomach' : 0.2,' hurts' : 0.2\},$$

kde čísla za jednotlivými příznaky určují pravděpodobnost náležitosti odpovídajících příznaků do negativní třídy.

Všimněme si, že v databázi příznaků klasifikátoru se nevyskytuje slovo `'Aaaa'`. To znamená, že klasifikátor se při svém učení s tímto slovem doposud nesetkal, a tak mu přiřadí pravděpodobnost 0,5. Jeho databáze příznaků pro klasifikaci vstupní věty *'Aaaa, my stomach hurts.'* bude tedy vypadat následovně:

$$\{' my' : 0.6,' stomach' : 0.2,' hurts' : 0.2,' Aaaa' : 0.5\}$$

Nyní už má klasifikátor vše potřebné k tomu, aby vypočítal pravděpodobnost, zda vstupní text náleží do relevantní, nebo do nerelevantní třídy. Bude postupovat podle rovnice 3.6.

$$\begin{aligned} P &= \frac{p_1 p_2 p_3 \dots p_N}{p_1 p_2 p_3 \dots p_N + ((1-p_1)(1-p_2)(1-p_3)\dots(1-p_N))} \\ P &= \frac{0.2 \cdot 0.6 \cdot 0.2 \cdot 0.5}{0.2 \cdot 0.6 \cdot 0.2 \cdot 0.5 + ((1-0.2)\cdot(1-0.6)\cdot(1-0.2)\cdot(1-0.5))} \\ P &= 0.0857 \end{aligned}$$

Dle tohoto výpočtu se tedy pravděpodobnost, že věta *'Aaaa, my stomach hurts.'* patří do negativní třídy, rovná 0,0857. Což znamená, že věta rozhodně náleží do pozitivní třídy.

Kapitola 4

SVM klasifikátor

Klasifikační metoda pomocí *SVM (support vector machines)* je společně s výše popsaným Bayesovským klasifikátorem další klasifikační metodou, kterou se v této práci zabýváme. Jedná se o metodu učení s učitelem, která je v současné době nejvíce používanou metodou v NLP. Mimo NLP má však také velkou škálu využití, ať už pro klasifikační účely, nebo kupříkladu pro účely regresní analýzy (např. pro předvídání chování finančních trhů, atd.).

V této práci nebude možné popsat problematiku klasifikátoru support vector machines do úplných detailů, nicméně zde budou popsány základní principy jeho fungování v rozsahu potřebném pro pozdější implementaci této metody. Pro podrobnější popis SVM doporučuji knihu Vladimira N. Vapnika z roku 1998 s názvem „Statistical Learning Theory“ [26] a článek „Support-Vector Networks“ [7].

4.1 Základy SVM klasifikátoru

SVM klasifikátor je algoritmus dosahující velmi dobrých výsledků v mnoha odvětvích. Poprvé byl tento klasifikační algoritmus představen v roce 1995 ve článku Vladimira N. Vapnika a kolektivu s názvem „Support-Vector Networks“ [7]. Metoda SVM byla navrhnuta jako klasifikační algoritmus. Aby bylo možné metodu SVM využít pro klasifikaci, musíme ji nejprve naučit pomocí určité sady dat, která je rozdělena na trénovací a testovací data. Každý jeden záznam v trénovacích a testovacích datech má definovanou *třídu (label)* do které záznam spadá. Každý záznam se skládá z několika příznaků, pomocí kterých se SVM učí a následně klasifikuje. Například klasifikujeme-li pomocí SVM klasifikátoru text, těmito příznaky mohou být mimo jiné jednotlivá slova, n-tice slov, délka textu, e-mailové adresy, Twitter tagy atd.

Nyní přejděme k matematickému popisu problematiky SVM. Předpokládejme určitý klasifikační problém a jistou datovou sadu, ve které je množství záznamů, které patří buď do pozitivní, nebo negativní třídy. Trénovací sada X pro tento problém obsahuje l záznamů. Jeden záznam v této sadě je tedy definován jako vektor $x_i \in \mathbb{R}^n$ kde $1 \leq i \leq l$. Každý tento vektor x_i se skládá z množiny příznaků $[x_1, x_2, \ldots, x_n] \in x_i$, kde $x_1, \ldots, x_n$ jsou jednotlivé atributy daného záznamu. Označení třídy y každého záznamu z dat je definováno jako $y \in \{1, -1\}$ v závislosti na tom, do které třídy daný záznam spadá. Trénovací sadu dat lze tedy zapsat jako:

$$X = \{(x_i, y_i)\}_{i=1}^{l} \tag{4.1}$$

$$x \in \mathbb{R}^n, x_i = [x_1, x_2, \ldots, x_n], i \in 1, \ldots |X| \tag{4.2}$$

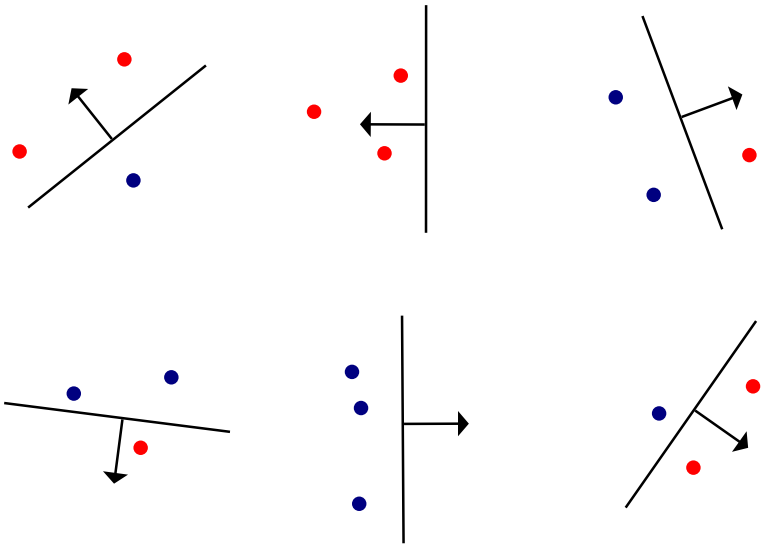
$$y \in \{1, -1\}. \tag{4.3}$$

Cílem SVM klasifikátoru je najít rozhodovací hranici, která rozděluje příznaky v trénovacích datech tak, aby rozdělení odpovídalo jednotlivým třídám a současně se snaží o maximalizaci vzdálenosti všech bodů od rozhodovací hranice. Rozhodovací hranice je definována jako *hyperplocha (nadrovina)*, což je prostor s n dimenzemi, který dělí prostor s $n+1$ dimenzemi do dvou podprostorů. Libovolnou hyperplochu lze definovat jako *diskriminační funkci (discriminant function)*[1] v následující formě:

$$f(x) = w^T x + b, \tag{4.4}$$

kde b je takzvaný *bias* a $w^T x$ definuje skalární součin mezi váhovým vektorem w a vektorem příznaků x. Tento skalární součin je definován jako:

$$w^T x = \sum_{j=1}^{n} w_j x_j. \tag{4.5}$$



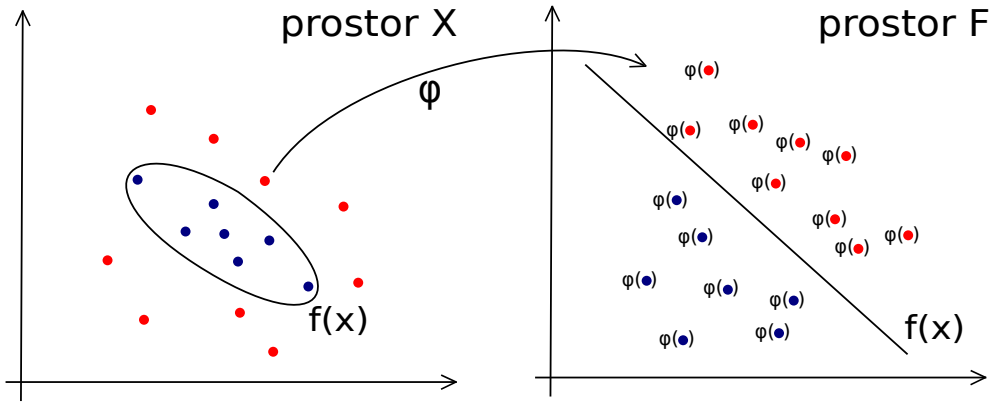
Obrázek 4.1: Několik hyperploch v prostoru R^2 s vektorem $w^T x$

4.2 Nelineární SVM a jádra

Jelikož je u základního klasifikátoru SVM hyperplocha dělící prostor příznaků lineární, považujeme tento klasifikátor za lineární. Protože je lineární dělicí hyperplocha základního SVM klasifikátoru omezující, je možné použít nelineární dělící hyperplochu, jejíž pomocí může klasifikátor dosahovat daleko lepších výsledků. Problémem takto nelineárních klasifikátorů ale je zvyšování komplexicity výpočtů při učení větších počtů příznaků (vysoká dimenzionalita dat). Pro vyřešení výše zmíněného problému byly klasické lineární SVM klasifikátory rozšířeny o podporu takzvaných *jaderných funkcí (kernel functions)* za pomocí nichž jsou lineární klasifikátory schopny vytvořit nelineární dělící hyperplochy. Pro

[1] diskriminační funkce je funkce, která optimálně dělí prostor příznaků. [14]

vysvětlení způsobu, jakým se této nelinearity dosáhne, předpokládejme klasický lineární klasifikátor.



Obrázek 4.2: Transformace vstupního prostoru X do prostoru příznaků F pomocí mapovací funkce $\varphi(x)$. $f(x)$ je znázornění diskriminační funkce

Nyní převedeme trénovací sadu X, také známou jako *vstupní prostor (input space)*, do vysoce dimenzionálního prostoru příznaků F za pomoci nelineární mapovací funkce $\varphi : X \to F$ (viz. obrázek 4.2). Pro prostor příznaků tedy budeme mít následující diskriminační funkci:

$$f(x) = w^T \varphi(x) + b. \tag{4.6}$$

Jelikož je při výpočtu diskriminační funkce $f(x)$ třeba vypočíst mapovací funkci $\varphi(x)$, velmi výrazně roste náročnost jejího výpočtu s počtem dimenzí vstupů. Mějme kupříkladu následující mapovací funkci:

$$\varphi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T. \tag{4.7}$$

Z této mapovací funkce jsme nyní schopni spočítat diskriminační funkci v prostoru příznaků:

$$f(x) = w_1x_1^2 + \sqrt{2}w_2x_1x_2 + w_1x_2^2 + b. \tag{4.8}$$

Je zcela zřejmé, že tento způsob výpočtu je neúnosný. Museli bychom totiž pro výpočet transformovat celý vstupní prostor do prostoru příznaků, což by zvýšilo časovou a paměťovou náročnost klasifikátoru. Tento způsob tedy není vhodný. Existuje však řešení, kterým lze spočítat diskriminační funkci $f(x)$, aniž bychom museli znát, chápat a počítat mapovací funkci do prostoru F. Pro toto řešení je třeba vyjádřit váhový vektor w jako lineární kombinaci jednotlivých záznamů z tréninkové sady.

$$w = \sum_{i=1}^{n} \alpha_i x_i. \tag{4.9}$$

Z toho plyne, že diskriminační funkce vstupního prostoru X je:

$$f(x) = \sum_{i=1}^{n} \alpha_i x_i^T \cdot x + b \tag{4.10}$$

a diskriminační funkce prostoru příznaků F je:

$$f(x) = \sum_{i=1}^{n} \alpha_i \varphi(x_i)^T \varphi(x) + b. \tag{4.11}$$

Existence těchto dvou reprezentací diskriminační funkce pro vstupní prostor X a pro prostor příznaků vzhledem k proměnné α_i bývá nazývána takzvanou *duální reprezentací (dual representation)*.

Kartézský součin $\varphi(x_i)^T \varphi(x)$, kde $x_i, x \in X$ tedy představuje takzvanou *jadernou funkci (kernel function)*. Jaderná funkce je tedy definována jako:

$$k(x, z) = \varphi(x_i)^T \varphi(x). \tag{4.12}$$

Jelikož jadernou funkci lze vypočítat pouze pomocí příznaků ve vstupním prostoru, je možné vypočítat diskriminační funkci, aniž bychom znali potřebnou mapovací funkci. Díky této vlastnosti jaderných funkcí nemusíme transformovat celý vstupní prostor do nového prostoru příznaků F, ale spočítáme pouze jadernou funkci pro jednotlivé příznaky. Tudíž počet dimenzí prostoru F neovlivní komplexnost výpočtu. Výše zmíněná operace, kdy využijeme pouze kartézský součin bodů v daném prostoru, bývá často v literatuře nazývána jako takzvaný *kernel trick*. Pro výpočet diskriminační funkce $f(x) = w^T \varphi(x) + b$ tedy použijeme jadernou funkci $k(x, z) = \varphi(x)^T \varphi(z)$ a dostaneme $f(x) = k(x, z) + b$.

Nyní si na příkladu nejprve znázorníme náročnost převádění všech souřadnic do prostoru F. Předpokládejme prostor $X \in \mathbb{R}^2$ (tzn. $x = (x_1, x_2)$) a dále pak polynomiální mapovací funkci druhého řádu

$$\varphi(x) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2), \tag{4.13}$$

která převede vektor příznaků ze vstupního prostoru X do prostoru F. Abychom nyní pomocí této mapovací funkce vypočítali jadernou funkci, musíme oba body převést do prostoru F a provést skalární součin v prostoru F. Tudíž:

$$K(x, z) = \varphi(x)^T \varphi(z) \tag{4.14}$$

$$K(x, z) = 1 + x_1 z_1 + x_2 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 + x_1 z_1 x_2 z_2. \tag{4.15}$$

Nyní si představme, že nemáme dvoudimenzionální prostor, ale velmi vysoce dimenzionální prostor. Převádění x a z do prostoru F by bylo velmi výpočetně náročné. Proto na následujícím příkladu předvedeme použití výše zmíněné metody *kernel trick*, která umožní vypočíst jadernou funkci, aniž bychom potřebovali spočítat transformace do prostoru příznaků F. Předpokládejme jadernou funkci:

$$K(x, z) = (1 + x^T z)^2, , \tag{4.16}$$

kterou roznásobíme a dostaneme:

$$K(x, z) = 1 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 z_1 + 2x_2 z_2 + 2x_1 z_1 x_2 z_2, \tag{4.17}$$

což je skutečně skalární součin definovaný v prostoru příznaků s mapovací funkcí

$$\varphi(x) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2). \tag{4.18}$$

Tím jsme dokázali, že opravdu není třeba převádět celý prostor příznaků, ale stačí pouze vypočítat jednoduchou funkci $K(x, z) = (1 + x^T z)^2$, a tím dosáhnout daleko jednodušeji kýženého výsledku.

4.2.1 Jádra

Jádra, nebo také jaderné funkce, jsou tedy funkce, pomocí nichž je klasifikátor SVM schopen vypočítat diskriminační rovnici pro optimální rozdělení prostoru příznaků. V závislosti na tom, jaké jádro je při trénování klasifikátoru použito, je klasifikátor lineární nebo nelineární a může pak dosahovat na stejných datech rozdílných výsledků.

V praxi je používáno několik různých jader, nejčastěji *RBF (radial basis function)*, dále *polynomiální jaderná funkce* a někdy je používána i základní *lineární jaderná funkce*.

Lineární jaderná funkce

Lineární jaderná funkce je nejjednodušší ze všech jaderných funkcí. Počítá totiž skalární součin z bodů ve vstupním prostoru, tudíž je schopna pouze lineární klasifikace. Lineární jaderná funkce je definována následovně:

$$K(x, z) = x^T z. \tag{4.19}$$

Polynomiální jaderná funkce

Rovnice 4.16 popisuje tedy polynomiální jadernou funkci druhého řádu, nicméně v praxi jsou často používány polynomiální jaderné funkce vyšších řádů Q, obecně definované jako:

$$K(x, z) = (x^T z \ + c)^Q, \tag{4.20}$$

kde $c \geq 0$ určuje vliv vyšších a nižších řádů polynomu.

Polynomiální jaderné funkce jsou v NLP poměrně hodně používané [13]. Při použití jaderné funkce vyšších řádů ale může nastat takzvané *přeučení (overfitting)*, kdy se SVM klasifikátor naučí na velmi specifické příznaky, které nejsou pro korektní klasifikaci směrodatné, což způsobí markantní zhoršení schopností SVM klasifikátoru. Z tohoto důvodu je nejčastěji používána polynomiální jaderná funkce 2. a 3. řádu, která není schopna tak dokonale kopírovat tvar kolem příznaků.

Radiální jaderná funkce

Radiální jaderná funkce, často označovaná jako *RBF jaderná funkce*, nebo *RBF kernel* je další velmi používanou jadernou funkcí, založenou na radiální bázové funkci *RBF (radial basis function)*. Je definována jako:

$$K(x, z) = exp(-\frac{||x - z||^2}{2\gamma^2}). \tag{4.21}$$

Občas je také používána jiná forma definice pro RBF jaderné funkce, nicméně základní myšlenka, že čím vzdálenější bod v prostoru příznaků, tím menší má vliv na rozhodování, zůstává ve všech těchto definicích zachována. Proměnná γ, pro kterou musí platit $\gamma \geq 0$, definuje jak moc bude klasifikátor brát v úvahu vzdálenější body. Nevhodným výběrem této proměnné může podobně jako v případě polynomiální jaderné funkce dojít k přeučení, proto je důležité zvolit její vhodnou hodnotu.

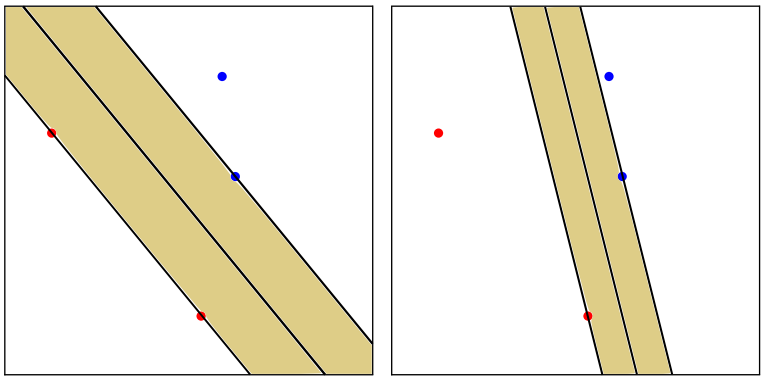
Další jádra

Tato tři výše zmíněná jádra/jaderné funkce samozřejmě nejsou jediná. Kdykoliv je možné vytvořit novou jadernou funkci, která bude popisovat nějaký jiný prostor. Taková jaderná funkce, kterou vytvoříme, však nemusí popisovat žádný reálný prostor. Existují tři přístupy, pomocí kterých se můžeme ujistit, že námi vytvořená jaderná funkce skutečně nějaký reálný prostor popisuje:

- *Konstrukce* – jádro zkonstruujeme z transformační funkce $\varphi(x)$.
- *Matematické podmínky jádra (Mercerovy podmínky)* – pokud jaderná funkce $K(x, z)$ splňuje následující dvě podmínky, pak existuje prostor, který je danou jadernou funkcí popsán:
  - $K(x, x^{'})$ musí být symetrická funkce (tzn. $K(x, x^{'}) = K(x^{'}, x), \forall x, x^{'} \in X$).
  - Pro matici: $\begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_N) \\ K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_N) \\ \cdots & \cdots & \cdots & \cdots \\ K(x_N, x_1) & K(x_N, x_2) & \cdots & K(x_N, x_N) \end{bmatrix}$
    musí platit, že je pozitivně semi-definitní pro libovolné $x_1, \ldots, x_N \in X$.
- *Do we even care?* – tento přístup je velmi zvláštní, nicméně bývá občas také používán. Spočívá ve vytvoření libovolné jaderné funkce a následné aplikaci v klasifikátoru SVM a pokud klasifikace funguje, nehraje roli, že dané jádro nepopisuje žádný reálný prostor. Tento přístup však rozhodně není doporučitelným.

4.3 Vytvoření rozhodovací hranice

Jak již bylo výše zmíněno, SVM klasifikátor se při učení snaží vytvořit optimální rozhodovací hranici, tedy hyperplochu, tak, aby dělila prostor příznaků na dva podprostory, kde každý obsahuje objekty vždy pouze z jedné třídy a vždy se snaží maximalizovat vzdálenost rozhodovací hranice ode všech bodů v prostoru (margin).



Obrázek 4.3: Ukázka možných šířek (marginů), vlevo maximální; vpravo tenká

Nyní definujme maximální šířku volného prostoru mezi rozdělovací hyperplochou f a nejbližšími body v prostoru X (margin) – viz obrázek 4.3. Maximální šířka je definována jako:

$$m_X(f) = \frac{1}{||w||}. \tag{4.22}$$

Abychom tedy dostali optimální rozdělovací hranici s maximální šířkou (marginem), budeme muset maximalizovat hodnotu šířky (marginu). Tato operace je ovšem ekvivalentní s minimalizací $\frac{1}{2}||w||^2$. Nalezení specifické diskriminační funkce s maximální šířkou (marginem) je tedy ekvivalentní s následujícím optimalizačním problémem:

$$\begin{array}{ll} \underset{w,b}{\text{minimalizace}} & \frac{1}{2}||w||^2 \\ \text{kde} & y_n(w^T x_i + b) \geq 1, \quad n = 1, \ldots, l \end{array}. \tag{4.23}$$

Tato optimalizace předpokládá, že trénovací množina X je lineárně separovatelná. Podmínka $y_n(w^T x_n + b) \geq 1$ potom zajišťuje, že diskriminační funkce oklasifikuje všechna data v trénovací množině korektně.

Nicméně může nastat problém, že trénovací množina X nemusí být lineárně separovatelná. Aby byl klasifikátor schopný naučit se i na takové trénovací množině, musíme jej upravit tak, aby mohl nekorektně oklasifikovat nějaké záznamy z trénovací množiny a tím, i za cenu chyb, diskriminační funkci najít. Toto opatření může někdy pomoci nalézt větší maximální šířku (margin), a tím zlepšit výsledky klasifikátoru oproti předchozí, menší maximální šířce (marginu). Tuto úpravu uděláme za pomocí takzvané *chybové proměnné (slack variable)* ξ_i, kterou odečteme od pravé strany optimalizační podmínky – tzn. umožníme chybu. Takto upravený problém nyní vypadá následovně:

$$\begin{array}{ll} \underset{w,b}{\text{minimalizace}} & \frac{1}{2}||w||^2 \\ \text{kde} & y_n(w^T x_i + b) \geq 1 - \xi_i, \quad n = 1, \ldots, l \end{array}. \tag{4.24}$$

Pokud tedy platí, že $y_i(w^T x_i + b) < 1$ respektive $\xi_i > 1$, pak je záznam $x_i \in X$ špatně klasifikován. Pokud ale platí, že $0 \leq \xi_i \leq 1$, pak záznam $x_i \in X$ leží uvnitř maximální šířky. Záznam x_i je tedy klasifikován korektně, jestliže platí, že $\xi_i \leq 0$. Z tohoto vyplývá, že suma všech chybových proměnných reprezentuje míru chybovosti:

$$\xi(X) = \sum_{i=1}^{n} \xi_i \tag{4.25}$$

$$X = \{(x_i, y_i)\}_{i=1}^{l}. \tag{4.26}$$

Abychom byli schopni minimalizovat míru chybovosti (penalizaci za špatnou klasifikaci a šířkové (marginové) chyby – případ kdy záznamy leží uvnitř maximální šířky) vzhledem k maximalizaci šířky (marginu), zavedeme konstantu $C > 0$, kterou budeme míru chybovosti přenásobovat. Tato konstanta se nazývá *konstanta měkké maximální šířky (soft-margin constant)*. Naše optimalizační úloha tedy s touto konstantou měkké maximální šířky vypadá následovně:

$$\begin{aligned} &\underset{w,b}{\text{minimalizace}} && \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{kde} && y_n(w^T x_i + b) \geq 1 - \xi_i, \quad n = 1, \ldots, l \end{aligned} \quad . \tag{4.27}$$

Nyní pro vyřešení tohoto optimalizačního problému použijeme metodu Lagrangeových multiplikátorů a získáme optimalizační problém:

$$\begin{aligned} &\underset{\alpha}{\text{maximalizace}} && \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ &\text{kde} && \sum_{i=0}^{n} y_i \alpha_i = 0, \qquad\qquad 0 \leq \alpha_i \leq C \end{aligned} \quad . \tag{4.28}$$

Skalární součin $x_i^T x_j$ v maximalizační rovnici můžeme nahradit libovolnou jadernou funkcí a dosáhnout tím nelineární transformace, a tedy i velké maximální šířky ve vysoce dimenzionálních prostorech příznaků (viz kapitola 4.2.1).

Díky využití Lagrangeovy funkce má výsledek řešení optimalizačního problému zajímavé vlastnosti. Bylo například dokázáno, že takto získané řešení je vždy globální díky tomu, že formulace problému je konvexní [6].

Zajímavou vlastností klasifikátoru support vector machines je, že ne všechny vektory z tréninkové množiny se podílejí na výsledném řešení. Použijme rovnici pro výpočet váhového vektoru w, získaného derivací při řešení Lagrangeovy funkce, a to:

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i. \tag{4.29}$$

Všechny vektory x_i, pro které tedy platí, že $\alpha_i > 0$, jsou vektory, které leží na pomezí maximální šířky, uvnitř maximální šířky, nebo jsou špatně klasifikovány. Tyto vektory jsou nazývány *support vektory (support vectors)*. Vektory, které však mají hodnotu $\alpha_i \leq 0$, nejsou vůbec do řešení zahrnuty. Tudíž tyto vektory by mohly být zcela odstraněny z tréninkové sady a nemělo by to žádný vliv na výsledné řešení. Díky této vlastnosti jsou SVM méně náchylné k přeučení a také klasifikační model, který tyto vektory tvoří, je díky tomu velmi malý a rychlý.

4.4 Výběr volných parametrů klasifikátoru a jaderných funkcí

Vhodný výběr volných parametrů klasifikátoru a jaderných funkcí je velmi důležitým faktorem při používání klasifikátoru založeného na metodě SVM. Nejsou-li vybrány vhodné parametry, může klasifikátor dosahovat vskutku výrazně horších výsledků, než jsou-li vybrány tyto parametry optimálně. Obecně je třeba optimálně určit parametr C (míru chybovosti vzhledem k maximalizaci šířky (marginu) – viz kapitola 4.27) a parametr použité jaderné funkce (určující vlastnosti použité jaderné funkce – viz kapitola 4.2.1), jenž budeme nadále nazývat γ.

Nastává ovšem problém, jakým způsobem tyto volné parametry volit tak, aby byly výsledky klasifikace co nejpřesnější. Ve své knize „The Nature of Statistical Learning Theory“ [25] Vladimir Vapnik doporučil nastavovat volné parametry v závislosti na znalosti množiny trénovacích dat.

Jestliže takovými znalostmi o datech nedisponujeme, můžeme využít velmi často používanou metodu hledání hodnot optimálních parametrů v mřížce. Tato metoda sice funguje, ale aby byla přesná, je třeba ji počítat pro dostatečnou hustotu bodů v mřížce, což může být velmi výpočetně náročné.

V této práci byla pro hledání optimálních metod použita metoda *simulovaného žíhání (simulated annealing)*. Simulované žíhání je pravděpodobnostní optimalizační metoda pro nalezení globální minimální nebo maximální energie, odpovídající souřadnicím ve stavovém prostoru. Tato metoda je inspirovaná fyzikou procesu žíhání oceli, při kterém se materiál předehřeje na určitou teplotu a postupně se nechá ochlazovat, čímž se odstraní napětí v materiálu, který se takto homogenizuje. U optimalizačního procesu simulovaného žíhání se nastaví počáteční souřadnice ve stavovém prostoru a počáteční teplota, která se postupně po krocích snižuje až do úplného vychladnutí.

Při každém kroku se vygenerují náhodné souřadnice ve stavovém prostoru (stav) a spočítá se k nim odpovídající energie. Jestliže je tato energie nižší, než energie předchozího stavu, pak je tento stav použit pro další iteraci. Pokud však energie nižší nebyla, spočítá se pravděpodobnost skoku do daného stavu s vyšší energií. Tato pravděpodobnost závisí na teplotě – čím vyšší teplota, tím více umožňuje algoritmus skočit do stavu s vyššími hodnotami energie. Tímto se významně eliminuje riziko uváznutí v lokálním minimu. Ekvivalentně lze algoritmus použít pro maximalizaci energie.

4.4.1 Popis algoritmu simulovaného žíhání

Pro co nejjasnější předvedení použitého algoritmu simulovaného žíhání představíme jeho pseudokód:

Algoritmus 1 Pseudokód algoritmu simulovaného žíhání – minimalizace energie

```
1:  state ← STATE0
2:  energy ← calculate_energy(s)
3:  state_best ← state, energy_best ← energy
4:  temp ← INIT_TEMP
5:  while temp > STOP_TEMP do
6:     temp ← temp · COOLING_FACTOR
7:     state_new ← generate_neighbors(state)
8:     energy_new ← calculate_energy(state_new)
9:     if P(energy, energy_new, temp) > random() then
10:       state ← state_new, energy ← energy_new
11:    end if
12:    if energy_new < energy_best then
13:       state_best ← state_new, energy_best ← energy_new
14:    end if
15: end while
16: return state_best
```

Proměnné $STATE0$ – počáteční stav (za stav budeme nadále považovat bod v prostoru souřadnic volných parametrů SVM klasifikátoru), $INIT\_TEMP$ – počáteční teplota, $STOP\_TEMP$ – konečná teplota a $COOLING\_FACTOR$ – faktor chládnutí jsou parametry, které jsou při spuštění algoritmu simulovaného žíhání zadány. Funkce $calculate\_energy(stav)$ potom počítá energii stavu $stav$ a funkce $generate\_neighbor(stav)$ vygeneruje možný sou-

sední stav ke stavu *stav*. V pseudokódu je také použita funkce *random*(), která generuje náhodná čísla v rozmezí $< 0, 1 >$.

Na následujících řádcích se podrobněji podíváme na některé zajímavé součásti tohoto algoritmu.

Výpočet energie stavu

Abychom byli schopni najít minimální energii systému v souřadnicích, musíme být schopni vypočítat energii libovolného vygenerovaného stavu. Pro tuto optimalizaci volných parametrů klasifikátoru SVM jsme vycházeli ze dvou požadavků, které na SVM klasifikátor máme, a sice:

- snažíme se maximalizovat přesnost klasifikátoru
- snažíme se minimalizovat počet support vektorů klasifikátoru.

Výsledná energie potom bude záviset na míře naplnění těchto dvou výše zmíněných požadavků.

Vzhledem k tomu, že implementovaná metoda simulovaného žíhání je minimalizační, musíme převést reprezentaci těchto požadavků tak, aby vytvořily metriku umožňující využití v procesu minimalizace.

V případě prvního požadavku (snaha o maximalizaci přesnosti klasifikátoru), je zřejmé, že maximalizaci je třeba převést na minimalizaci. Místo o maximalizaci přesnosti klasifikátoru tedy usilujeme o minimalizaci chybovosti klasifikátoru. Takto tedy dostaneme vzorec pro výpočet energie přesnosti E_p klasifikátoru:

$$E_p = 1 - (\frac{spravne\_klasifikovane\_vstupy}{vsechny\_vstupy}), \qquad E_p \in < 0, 1 > . \tag{4.30}$$

Pomocí tohoto vzorce vypočteme energii chybovosti klasifikátoru, kterou se budeme snažit minimalizovat.

Pokud jde o druhý požadavek (snaha o minimalizaci počtu support vektorů klasifikátoru), tento požadavek již má minimalizační charakter, a proto jej nemusíme nijak převádět. Pro výpočet energie tohoto požadavku E_{SV} dostaneme následující vzorec:

$$E_{SV} = \frac{pocet\_pouzitych\_SV}{vsechny\_SV}, \qquad E_{SV} \in < 0, 1 > . \tag{4.31}$$

Výpočet celkové energie E stavu se tudíž bude rovnat součtu energie přesnosti a energie support vektorů, tedy:

$$E = E_p + E_{SV}, \qquad E \in < 0, 2 > . \tag{4.32}$$

Generování možného sousedního stavu

Pro generování možného sousedního stavu byla použita metoda navrhnutá v článku „Parameter determination of support vector machine and feature selection using simulated annealing approach“ [16]. Tato metoda je založena na vygenerování směrového vektoru s počátkem v aktuálním stavu. Na tomto vektoru je pak náhodně zvolen možný sousední stav.

Pravděpodobnost přijetí nového stavu

Při vyšších teplotách na počátku běhu algoritmu je možné, že algoritmus simulovaného žíhání použije jako svůj následující stav stav, který má vyšší energii, než stav původní. Tímto postupem se zamezí uváznutí v lokálních minimech a umožní se tak nalezení globálního minima. S postupným snižováním teploty se pravděpodobnost skoku na takovéto stavy s vyšší energií snižuje.

Pro výpočet této pravděpodobnosti je použit následující vzorec:

$$P_{skoku} = e^{\frac{\Delta E}{T}}, \tag{4.33}$$

kde ΔE je rozdíl mezi energií sousedního stavu a původního stavu a T je aktuální teplota v iteraci simulovaného žíhání. Pro určení, zda se má do nového stavu skočit či ne, je vygenerováno náhodné číslo $X \in < 0, 1 >$. Pokud je $P_{skoku} > X$, pak je pro další iteraci použit sousední stav jako stav základní.

Kapitola 5

Výběr vhodných příznaků

Počínaje touto kapitolou se budeme zabývat postupy a problémy, které se vyskytly při implementaci vytvořeného klasifikačního programu. Následující kapitoly jsou tedy na rozdíl od předchozích kapitol zaměřeny praktičtěji.

V této kapitole popíšeme tvorbou příznaků, které klasifikátory používají pro trénovaní a klasifikaci textu.

Pro potřeby implementovaných klasifikátorů rozdělujeme příznaky na dvě skupiny:

- Speciální příznaky – v textu se mohou nacházet skryté znaky, které většinou nejsou klasifikátory během klasifikace schopny zachytit a které mohou výrazným podílem napomoci přesnosti klasifikace. Každý druh klasifikovaného textu ale může mít citlivost k různým příznakům zcela jinou. Proto je nutné dbát na to, jak vybrat optimální příznaky pro daný vstupní text a použitou klasifikační metodu.
- Textové příznaky – mimo výše zmíněných speciálních příznaků se v textu nacházejí příznaky jednotlivých slov, které jsou pro klasifikaci nejdůležitějšími nositeli informace. Proto také velmi záleží na výběru, popřípadě úpravě, vstupních slov a způsobu jejich převodu na textové příznaky.

Nyní přistoupíme k detailnějšímu popisu výše popsaných skupin příznaků tak, jak je k nim přistupováno v praktické části této práce. Následně popíšeme práci Bayesovského klasifikátoru a klasifikátoru SVM s vytvořenými příznaky.

5.1 Speciální příznaky

V této části práce budeme popisovat jednotlivé druhy speciálních příznaků, které jsou z textu extrahovány a které jsou následně předávány klasifikátorům, jež s nimi dále pracují. Tyto speciální příznaky jsou rovněž implementovány v programu vytvořeném jako součást této práce.

Speciální příznaky jsou mnohdy v textu skryté a klasickými přístupy pro získávání textových příznaků z textu (=tokenizací) je často nejsme schopni získat. Tyto speciální příznaky jsou nositeli důležitých informací proto, že mohou z textu například určovat náladu pisatele, zobecňovat informace obsažné v textu atd., a tím umožnit zpřesnění klasifikace. Vzhledem k tomu, že takovýchto speciálních příznaků je jistě možné definovat velké množství, je nereálné je popsat a implementovat všechny. V implementovaném programu proto bylo zvoleno několik typů těchto speciálních příznaků, které byly implementovány a budou zde tedy popsány.

V následujícím popisu příznaky dělíme do skupin podle typu informace, kterou se pomocí nich snažíme z textu získat. Tyto skupiny jsou potom dále děleny na typy příznaků, popisující jednotlivé varianty příznaků v dané skupině .

V dodatku C této práce jsou podrobně popsány regulární výrazy, jejichž pomocí jsou speciální příznaky z textu získávány.

5.1.1 URL

Skupina příznaků s názvem URL je skupinou, hledající v klasifikovaném textu internetové odkazy (`http://adresa.com`, `ftp://adresa.cz`, `adresa.cz/odkaz.html`, atd.) a snažící se z těchto odkazů vytvořit příznaky co nejvhodnější pro klasifikaci.

Skupina URL má v praktické části práce následující možné varianty speciálních příznaků:

- *Celé URL* – celé URL nalezené v textu je bráno jako příznak.
- *Doména URL* – doména daného URL je brána jako příznak (např. `http://healthland.time.com/2013/04/02/.../index.html` ⟶ `time.com`).
- *Existence URL ANO/NE* – existuje-li v textu URL, pak se vytvoří pro daný text příznak definující, že se v textu URL vyskytovalo. Pokud se v textu URL nenachází, vytvoří se namísto příznaku *ANO* příznak *NE*, definující neexistenci URL v textu.
- *Existence URL ANO* – tento typ příznaku URL je variací na předchozí příznak *Existence URL ANO/NE*. Příznak se vytvoří pouze tehdy, pokud se v textu URL nachází. Jestliže se v textu URL nenachází, příznak vytvořen není.

5.1.2 E-mailové adresy

Skupina příznaků e-mailové adresy sdružuje dohromady varianty příznaků, pracující s nalezenými e-mailovými adresami v textu.

Možné varianty příznaků v této skupině jsou:

- *Celý e-mail* – celý e-mail nalezený v textu je brán jako příznak.
- *Existence e-mailu ANO/NE* – příznak definuje, zda se v textu nachází nějaká e-mailová adresa. Pokud ano, je vytvořen příznak *ANO* a pokud ne, je vytvořen příznak *NE*.
- *Existence e-mailu ANO* – nachází-li se v textu e-mailová adresa, vytvoří se příznak definující existenci této e-mailové adresy. Na rozdíl od předchozí varianty *Existence e-mailu ANO/NE* se nevytváří příznak *NE* v případě nenalezení e-mailové adresy v textu.

5.1.3 Emotikony

Další skupinou příznaků, které v této práci rozpoznáváme a předáváme klasifikátorům, jsou příznaky emotikonů (smajlíků).

Tuto skupinu příznaků pro potřeby naší implementace dělíme na následující typy:

- *Existence jednotlivých emotikonů* – pro každý nalezený emotikon je vytvořen příznak.

- *Existence emotikonů obecně ANO/NE* – jestliže se v textu nachází nějaké emotikony, je vytvořen příznak *ANO*, jestliže se nenacházejí, je vytvořen příznak *NE*.
- *Existence emotikonů obecně ANO* – obdobně jako *Existence emotikonů obecně ANO/NE*, pouze s tím rozdílem, že jestliže se v textu nenacházejí žádné emotikony, tento příznak se nevytvoří.
- *Nálada emotikonů* – v textu jsou vyhledávány emotikony a zjišťuje se jejich nálada. Typy emotikonů jsou rozděleny do třech tříd – smutné(„:-(“, „:'(“, ...), veselé(„:-)“, „:-P“, ...) a ostatní („o.O“, „:-O“, ...). Jestliže je v textu nalezen emotikon z výše zmíněných tříd, pak je vytvořen odpovídající příznak *SAD*, *HAPPY*, nebo *OTHER*. Nachází-li se v textu více typů emotikonů, je samozřejmě vytvořeno více odpovídajících příznaků (tzn. například pokud se v textu nachází následující množina emotikonů: „:-)“, „:-P“, „:(“, jsou pro daný text vytvořeny dva příznaky – *HAPPY* a *SAD*).

5.1.4 Tagy

Takzvané tagy jsou textové značky (klíčová slova), které autoři připisují k textu, aby daný text přiřadili k nějakému tématu a umožnili tak ostatním uživatelům snáze nalézt příspěvky k jimi hledanému tématu. Tyto tagy bývají ve tvaru #TAG (např: „#influenza“, „#sick“, ...).

V této práci rozlišujeme v případě této skupiny příznaků následující typy:

- *Celý tag* – celý tag nalezený v textu je použit jako příznak pro klasifikaci.
- *Existence tagů ANO/NE* – jestliže se v textu nacházejí nějaké tagy, je vytvořen příznak *ANO*, jestliže se nenacházejí, je vytvořen příznak *NE*.
- *Existence tagů ANO* – obdobně jako v případě typu *Existence tagů ANO/NE*, ale pokud žádný tag nebyl nalezen, příznak NE není vytvořen.

5.1.5 Tagy uživatelů

Podobně jako v případě tagů, mohou uživatelé ve svých příspěvcích označovat ostatní uživatele. Tagy uživatelů jsou používané ve formátu @JMÉNO_UŽIVATELE (např. @Rick_Astley, atd.).

V této práci rozlišujeme následující typy tagů uživatelů:

- *Celý tag uživatelů* – celý tag nalezený v textu je použit jako příznak pro klasifikaci.
- *Existence tagů uživatelů ANO/NE* – jestliže se v textu nacházejí nějaké tagy uživatelů, je vytvořen příznak *ANO*, jestliže se nenacházejí, je vytvořen příznak *NE*.
- *Existence tagů uživatelů ANO* – obdobně jako v případě typu *Existence tagů uživatelů ANO/NE*, ale pokud žádný tag uživatelů nebyl nalezen, příznak NE není vytvořen.

5.1.6 Věty

V případě této skupiny se jedná pouze o jediný typ příznaku. V celém vstupním textu se spočítá počet napsaných vět, který je následně použit jako příznak pro klasifikaci.

5.1.7 Časy

V textu se velmi často nacházejí různé časové údaje, které by mohly velkou měrou přispět k lepším výsledkům klasifikace. Proto byla vytvořena skupina příznaků nazvaná *Časy*, která vytváří příznaky z nalezených časových informací v textu.

Obsahuje tyto typy:

- *Celý čas* – jako příznaky pro klasifikaci se použijí všechny časové informace nalezené v textu.
- *Čas ve 24h formátu* – časové informace nalezené v textu jsou konvertovány do 24 hodinového formátu a následně použity jako příznaky.
- *Pouze hodiny ve 24h formátu* – časové informace jsou stejně jako v případě typu *Čas ve 24h formátu* převedeny do 24h formátu, ale jako příznaky jsou použity pouze informace o hodinách.

5.1.8 Data

Podobně jako skupina *Časy*, funguje i skupina *Data*, která vytváří příznaky z údajů o datu. V následujícím textu budeme písmenem D označovat dny, písmenem M měsíce a písmenem Y roky (tzn. datum 15.2.1998 je ve formátu DMY).

Obsahuje tyto možné typy příznaků:

- *Celé datum* – jako příznak je v případě tohoto typu příznaků použito datum nalezené v textu.
- *Datum ve formátu DMY* –v případě tohoto typu je datum nalezené v textu převedeno do formátu DMY a v tomto formátu je použito jako příznak (tzn. DMY).
- *Datum ve formátu MY* – podobně jako v předchozím typu je datum převedeno do formátu DMY, ale jako příznak jsou použity pouze informace o měsíci a roku (tzn. MY).
- *Datum ve formátu Y* – opět podobné jako *Datum ve formátu DMY*, ale pro vytvoření příznaku je použit pouze rok z data nalezeného v textu (tzn. Y).

5.2 Textové příznaky

Pro získávání textových příznaků z textu obecně existuje velmi mnoho přístupů a způsobů, zde však popíšeme pouze dva – první budeme nazývat základní přístup k získávání textových příznaků a druhý budeme nazývat alternativní přístup k získávání textových příznaků z textu. Oba tyto přístupy je možné ještě rozšířit o využití stematizátorů (převedení textových příznaků na kořeny). Pro převedení slov do obecnějšího tvaru lze kromě *stematizace* použít také takzvané *lematizace*, která na rozdíl od stematizace nepřevede slova na kořeny, ale do jejich základního tvaru. Pro potřeby této práce však byla zvolena metoda stematizace.

5.2.1 Základní přístup k získávání textových příznaků

Základní přístup pro získávání textových příznaků dělí vstupní větu pouze na jednotlivá slova, která jsou posléze použita jako textové příznaky.

To znamená:

Mějme vstupní text *'Damned headache. I should be sleeping.'*. Takovýto text se rozdělí na slova, tzn. vznikne následující seznam slov:

$['Damned','headache','I','should','be','sleeping']$.

Takto vytvořená slova jsou tedy textové příznaky, s nimiž se klasifikátor učí nebo které klasifikuje.

5.2.2 Základní přístup s využitím stematizace

Tento přístup je velmi podobný předchozímu. Liší se od něj pouze tím, že slova, která jsou vytvořena rozdělením pomocí stematizačního algoritmu převede na kořeny (=stematizace). Tímto krokem se sníží počet slov ve slovníku klasifikátoru. To způsobí, že u Bayesovského klasifikátoru dojde k výraznému zmenšení velikosti slovníku vytvořeného při učení a u klasifikátoru SVM se sníží počet dimenzí příznaků a zjednoduší se tak nalezení optimální nadroviny. To znamená:

Mějme vstupní text „Damned headache. I should be sleeping.“. Takovýto text se rozdělí na slova, tzn. vznikne následující seznam slov:

$['Damned','headache','I','should','be','sleeping']$.

Nyní všechna tato slova převedeme na tvar kořene (stematizace), čímž získáme seznam následujících slov:

$['Damn','headache','I','should','be','sleep']$.

Takto vytvořené kořeny slov se již použijí jako textové příznaky pro učení klasifikátorů, a nebo může být vstupní text rozdělený na tyto textové příznaky klasifikátory klasifikován do daných tříd.

5.2.3 Alternativní přístup k získávání textových příznaků

Při využití základního přístupu k získávání texových příznaků je problémem skutečnost, že vytvořené příznaky jsou zcela samostatné bloky, které se použité klasifikátory (ať už Bayesovský klasifikátor, nebo klasifikátor SVM), učí aniž by byly schopny z těchto textových příznaků určit kontext. Bayesovský klasifikátor nebere v potaz závislosti jednotlivých textových příznaků na sobě (což vychází z odvození rovnice 3.6 používané pro klasifikaci – viz 3.1.1), proto někdy bývá označován jako naivní. Tuto nevýhodu jsme se snažili kompenzovat tím, že jsem použili alternativní přístup k získávání textových příznaků, zachovávající nejbližší kontext textových příznaků. Ten na rozdíl od základního přístupu k získávání textových příznaků nevytváří příznaky pouze z jednotlivých slov, ale také z n za sebou jdoucích slov, čímž bere v potaz jejich kontext ve větě. Nevýhodou tohoto přístupu je, že pro velké učící množiny výrazně roste velikost slovníku u Bayesovského klasifikátoru a u SVM klasifikátoru se zvětšuje počet dimenzí příznaků, což vede k obtížnějšímu nalezení optimální dělící nadroviny a může vést i k přeučení. Tato vlastnost ale opět může být alespoň částečně kompenzována stematizací jednotlivých slov, a tím i snížením celkového počtu různých textových příznaků. Ukažme si alternativní přístup k získávání textových příznaků na příkladu:

Mějme opět text „'Damned headache. I should be sleeping.'“. Prvním krokem při zpracování tohoto textu je jeho rozdělení na věty, protože kontext za sebou jdoucích slov funguje

vždy v rámci jedné věty. Pro rozdělení vstupního textu na jednotlivé věty využíváme knihovnu NLTK (viz. kapitola 6.1.1). Vstupní text je nyní v tomto tvaru

$$['Damned\ headache.','I\ should\ be\ sleeping']$$

Na rozdíl od základního přístupu, který vytváří textové příznaky pouze z jednotlivých slov, vytváříme textové příznaky z libovolných za sebou jdoucích n-tic o maximální délce n vždy tak, že n-tice se skládají vždy jen ze slov dané věty. Hodnota n je nastavena na optimální hodnotu určenou experimentálně za pomocí testů tak, aby stále ještě vylepšovala výslednou přesnost klasifikátoru a přitom aby velikost slovníku nepřesáhla přijatelnou mez. Před tím, než se ze slov vytvoří textové příznaky, převedou se všechna slova na kořeny. Textové příznaky pro vstupní text a $n = 3$ budou tedy pro první větu vypadat následovně:

$$['Damn','headache',('Damn','headache')]$$

a pro druhou větu:

$$['I','should','be','sleep',['I','should'],['should','be'],['be','sleep'],$$
$$['I','should','be'],['should','be','sleep']]$$

S těmito textovými příznaky pak klasifikátor pracuje stejně jako s jakýmikoliv jinými textovými příznaky.

5.3 Příznaky v Bayesovském klasifikátoru

Bayesovský klasifikátor bere postupně všechny textové a speciální příznaky z trénovací množiny a v závislosti na manuálně přiřazených třídách trénovacích dat počítá pravděpodobnosti s jakými dané textové a speciální příznaky náležejí do negativní třídy. Takto vypočítané pravděpodobnosti pro jednotlivé textové a speciální příznaky si ukládá do svého slovníku, pomocí něhož následně klasifikuje nové vstupní texty.

V Bayesovském klasifikátoru je velmi důležité vybrat pro danou trénovací množinu dat vhodné příznaky, které se z textu budou extrahovat a používat pro klasifikaci. Proto je v této práci klasifikátor trénován se všemi možnými kombinacemi použitých příznaků a jsou zvoleny ty příznaky, pro které mají výsledky klasifikátoru na testovacích datech s těmito daty největší korelaci.

5.4 Příznaky v SVM klasifikátoru

Na rozdíl od Bayesovského klasifikátoru, problém volby vhodných textových a speciálních příznaků u SVM klasifikátoru částečně odpadá. Jelikož si klasifikátor při učení vytváří váhový vektor, který určuje míru důležitosti jednotlivých příznaků v prostoru příznaků pro výsledek klasifikace, je jasné, že klasifikátor dokáže irelevantní příznaky ignorovat a řídit se hlavně příznaky, které na výslednou klasifikaci budou mít nejlepší vliv.

Pokud ovšem bude použito příliš velké množství příznaků, klasifikátoru výrazně naroste počet dimenzí, což se může markantně projevit ve zhoršení klasifikačních schopností SVM klasifikátoru. Jelikož speciálních příznaků je jen velmi málo, tento problém by neměl nastat. Naproti tomu ale generování n-tic textových příznaků tento problém způsobit může. Proto je vhodné volit zvolit optimální velikost generovaných n-tic tak, aby se nezhoršovaly klasifikační schopnosti klasifikátoru.

Kapitola 6

Vlastní implementace

V této kapitole se budeme zabývat vlastní implementací obou klasifikačních algoritmů, popsaných v kapitolách 3 a 4. Kromě použitých knihoven zde popíšeme také architekturu programu, implementovaného v praktické části této práce. Také zde budou představeny problémy, ke kterým při implementaci došlo a rovněž bude prezentováno, jakým způsobem byly vyřešeny.

Celý program je implementován v programovacím jazyce Python 2.7. Jazyk Python byl zvolen hlavně z toho důvodu, že pro něj existuje mnoho knihoven, které mohly být v této práci použity a za jejichž pomoci se implementace výrazně zjednodušila. Takto byla využita zejména knihovna pro zpracování přirozeného jazyka `NLTK` (Natural Language ToolKit), knihovna `NumPy` (Numerical Python), která umožňuje v prostředí jazyka Python velmi jednoduše pracovat se složitější matematikou, knihovna `CVXOPT`, využitá k implementaci SVM klasifikátoru pro řešení úloh kvadratického programování a knihovna `pp` (Parallel Python) pro zjednodušení implementace paralelizace.

Tyto knihovny budou mimo jiné také podrobněji popsány dále v této kapitole.

6.1 Použité knihovny

6.1.1 NLTK

`NLTK`, neboli Natural Language ToolKit je balík knihoven pro skriptovací jazyk Python 2.7, určený pro symbolické a statistické zpracování přirozeného jazyka. Nabízí jednoduché rozhraní pro velké množství různých nástrojů pro zpracování textu, ale také velké množství ukázkových dat, která lze použít jako testovací a trénovací data při vyvíjení softwaru, pracujícího s přirozeným jazykem. Díky vynikající dokumentaci je `NLTK` skvělou knihovnou, výrazně zjednodušující implementaci softwaru, pracujícího nějakým způsobem s přirozeným jazykem.

V praktické části této práce je knihovna `NLTK` používána jako součást předzpracování vstupního textu určeného buď k učení klasifikátorů nebo již přímo pro klasifikaci. Je použita k těmto dvěma operacím s textem:

1. *Rozdělení vět* – prvním modulem z této knihovny, který je v této práci použit, je modul `punkt` (`nltk/tokenize/punkt`), jenž rozděluje vstupní text na seznam vět.

2. *Převedení slov na jejich kořeny (stematizace)* – druhým modulem z této knihovny je modul `snowball` (`nltk.stem.snowball`), jehož úkolem je z jednotlivých slov vstupního textu udělat kořen slov (více viz kapitola 5.2).

6.1.2 NumPy

`NumPy` je základním balíkem knihoven pro numerické výpočty v programovacím jazyce Python. Tento balík knihoven umožňuje jednoduchou práci s mnohadimenzionálními poli a dalšími objekty od nich odvozenými (maskovací pole, matice ...). Mimo to v sobě knihovna zahrnuje velké množství rychlých matematických operací nad poli, zahrnujících diskrétní Fourierovy transformace, třídění, základní lineární algebru a mnoho dalších. Využití této knihovny je pro programy řešící větší množství složitějších matematických výpočtů prakticky nutností, neboť výrazně zrychlí výpočty a programátorovi zjednoduší práci s matematickými daty. Mnoho dalších knihoven řešících nějaké rozsáhlejší matematické úkony je založeno právě na knihovně `NumPy`. Mezi tyto další knihovny patří i balík knihoven `CVXOPT` (viz kapitola 6.1.3), použitý pro implementaci SVM klasifikátoru v praktické části této práce.

6.1.3 CVXOPT

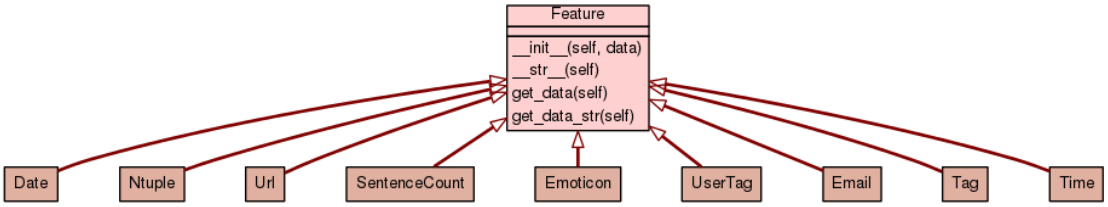
`CVXOPT` je volně dostupný balík knihoven pro řešení konvexních optimalizačních problémů (mimo jiné do této skupiny problémů patří i kvadratické programování, jež je použito pro implementaci SVM klasifikátoru) v programovacím jazyce Python 2.7. Hlavním úkolem tohoto balíku knihoven je zjednodušit vývoj softwaru, jenž potřebuje pro svůj běh řešit konvexní optimalizační úlohy, aniž by bylo potřeba implementovat složité optimalizační algoritmy.

6.2 Architektura programu

Tato kapitola má za cíl přiblížit architekturu a implementaci programu.

Program, implementovaný jako praktická část této práce, je, jak již bylo výše zmíněno, implementován v programovacím jazyce Python, jenž umožňuje využití objektově orientovaného přístupu k vývoji aplikací. Díky této vlastnosti Pythonu bylo možné implementovat program objektově a využít tak výhod tohoto přístupu.

Jak Bayesovský klasifikátor, tak klasifikátor založený na support vector machines, jsou v implementační části této práce vytvořeny tak, aby bylo možné je používat jako modulů programovacího jazyka Python – zcela nezávisle na sobě. To znamená, že lze oba klasifikátory importovat jako moduly do nově implementované aplikace a použít pro klasifikaci libovolného vstupního textu. Této vlastnosti bylo využito v projektu M-Eco, v němž byl Bayesovský klasifikátor použit ve skriptu vkládajícím nová data do databáze, kde filtroval relevantní data od nerelevantních a relevantní data byla následně vkládána do databáze. V této práci jsou tedy implementovány dvě základní třídy `SVM` a `BayesianClassifier`. Obě tyto třídy obsahují metody pro trénování klasifikátoru z trénovacích dat a metody umožňující po předchozím naučení klasifikátoru klasifikovat data do daných tříd.



Obrázek 6.1: Diagram tříd dědících z třídy Features

Jelikož jak Bayesovský klasifikátor, tak klasifikátor založený na support vector machines pracují se vstupním textem prakticky stejně, byla vytvořena třída `Entry` a třídy dědící z rodičovské třídy `Feature` (viz obrázek 6.1), nacházející se v adresáři `common`. Tyto třídy používají oba klasifikátory pro práci s veškerým vstupním textem, tedy jak pro klasifikaci, tak pro učení. Instance třídy `Entry` se vytváří pro každý text vstupující do klasifikátoru. Jejím úkolem je text rozdělit na příznaky tak, aby samotné klasifikátory již s textem nemusely nijak pracovat, ale aby dostaly pouze seznam textových a speciálních příznaků, které byly nalezeny ve vstupním textu. Tato třída tedy zodpovídá za vyhledávání jednotlivých speciálních příznaků (viz kapitola 5.1), které potom jako instance tříd, dědících ze třídy `Features` společně s n-ticemi jednotlivých textových příznaků (viz kapitola 5.2), předává klasifikátorům.

Aby bylo možné oba klasifikátory testovat a posléze i vzájemně porovnávat, byly pro každý klasifikátor vytvořeny třídy, jež mají na starost testování. Tyto třídy se nazývají `SVMTest` pro klasifikátor založený na support vector machines a `BayesianTest` pro Bayesovský klasifikátor. Obě tyto třídy obsahují metody pro spouštění testů klasifikátorů a jejich následné vyhodnocení.

Jelikož oba implementované klasifikátory využívají pro klasifikaci diametrálně odlišných přístupů, nakládají oba klasifikátory s příznaky získanými z instancí třídy `Entry` jinak. Z toho důvodu každý klasifikátor obsluhuje několik tříd, jež při klasifikaci nebo učení používá.

V případě Bayesovského klasifikátoru je takovouto další používanou třídou pouze jediná třída s názvem `WordDictionary`, která se stará o slovník příznaků Bayesovského klasifikátoru. Tento slovník příznaků obsahuje pravděpodobnostní hodnoty jednotlivých příznaků, nalezených během učení a pravděpodobnostní hodnoty jejich náležitosti do určité třídy. Tato třída také zabezpečuje veškeré operace prováděné s tímto slovníkem příznaků, jako například jeho uložení do souboru a opětovné načtení, díky čemuž je možné Bayesovský klasifikátor, stejně jako klasifikátor založený na metodě support vector machines používat, aniž bychom je museli bezprostředně před klasifikací učit, což může být velmi zdlouhavý proces. Mimo operací exportu a importu slovníku také tato třída umožňuje exportovat daný slovník do souboru v jazyce XML.

Podobně jako Bayesovský klasifikátor, tak i klasifikátor založený na metodě support vector machines, potřebuje třídu, jež pracuje se vstupním textem a s příznaky z něj vytvořenými a připraví příznaky do podoby, ve které je klasifikátor schopen s nimi pracovat. Tato třída se u SVM klasifikátoru jmenuje `Data`. Její funkcí je vytvořit prostor příznaků ze vstupních dat a data následně rozdělit na bloky trénovacích a testovacích dat. Další specifickou třídou, již klasifikátor založený na SVM používá, je třída `Kernel` a její podtřídy, které klasifikátor potřebuje pro výpočet jaderné funkce pro klasifikaci. Tuto třídu dědí v implementovaném programu následující tři specifické jaderné funkce:

- Lineární jaderná funkce
- Polynomiální jaderná funkce
- Radiální bázová jaderná funkce (RBF)

Poslední třídou, která je v souvislosti s klasifikátorem založeným na SVM spojena, je třída `Annealing`, jež řeší obtížný úkol vhodného výběru parametrů u klasifikátoru a jím používané jaderné funkce. Více o tomto výběru vhodných parametrů bylo popsáno v kapitole o SVM klasifikátoru (viz kapitola 4.4).

Pro podrobnější popis tříd je vygenerována dokumentace pomocí dokumentačního nástroje `epydoc`. Dokumentace je přiložena na CD, jež je součástí této práce.

6.3 Implementace klasifikačních algoritmů

V této části kapitoly se budeme podrobněji zabývat postupem implementace jednotlivých klasifikačních algoritmů. Pokusíme se zde rozebrat problémy, jež při implementaci nastaly a jejich řešení. Tato sekce bude pro větší přehlednost rozdělena na dvě části: implementaci Bayesovského klasifikátoru a klasifikátoru založeného na SVM.

6.3.1 Implementace Bayesovského klasifikátoru

Při implementaci Bayesovského klasifikátoru nastaly pouze dva větší problémy, a to potřeba ukládání dat slovníku Bayesovského klasifikátoru pro pozdější využití a otázka, jakým způsobem vytvořit klasifikační algoritmus, aby textové i speciální příznaky ovlivňovaly výsledek klasifikace stejnou měrou.

Ukládání slovníku Bayesovského klasifikátoru

Prvním problémem, který bylo třeba vyřešit, bylo ukládání slovníku, obsahujícího pravděpodobnostní hodnoty jednotlivých příznaků, na něž klasifikátor při svém učení narazil, do souboru tak, aby se klasifikátor nemusel před každou klasifikací znovu učit všechna data z trénovací sady, což by bylo velmi zdlouhavé. Pro tuto potřebu jsem zvolil knihovní funkci programovacího jazyku Python s názvem *pickle*, která umožňuje provést takzvanou serializaci libovolné datové struktury programovacího jazyka do souboru a zpět.

Typy příznaků a jejich vliv na klasifikaci

Druhým problémem, který nastal, bylo jakým způsobem by měly být zpracovávány textové a speciální příznaky. Jelikož počet textových příznaků může dalece převyšovat počet speciálních příznaků, nebylo by vhodné, aby se s těmito dvěma typy příznaků zacházelo stejným způsobem. Je totiž vysoce pravděpodobné, že pokud bude textových příznaků mnohem více než speciálních příznaků, budou mít speciální příznaky velmi malý dopad na výslednou klasifikaci. Proto v implementovaném programu Bayesovský klasifikátor pracuje s oběma typy příznaků samostatně a výsledek klasifikace je tvořen aritmetickým průměrem výsledků klasifikace obou typů příznaků.

6.3.2 Implementace SVM klasifikátoru

Při implementaci programu jsme se rozhodli ze studijních důvodů vytvořit vlastní implementaci SVM klasifikátoru namísto použití již hotových implementací tohoto algoritmu. Podobně jako při implementaci Bayesovského klasifikátoru nastaly i při implementaci klasifikátoru založeného na support vector machines určité problémy, z nichž zmíníme následující dva:

- Jaké parametry předat balíčku kvadratického programování.
- Jak optimalizovat výpočet skalárního součinu v prostoru příznaků klasifikátoru SVM.

Parametry kvadratického programování

Jak již bylo zmíněno výše v této kapitole o použitých knihovnách (viz kapitola 6.1), pro řešení optimalizačního problému, vycházejícího z matematického popisu klasifikátoru SVM (viz kapitola 4), byl použit balík knihoven pro programovací jazyk Python s názvem `CVXOPT`, přesněji modul `qp` z tohoto balíku knihoven, umožňující řešit optimalizační problémy pomocí kvadratického programování.

Optimalizační problém, který potřebujeme řešit, je popsán na konci kapitoly 4 a vypadá následovně:

$$\begin{array}{lll} \underset{\alpha}{\text{maximalizace}} & \displaystyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j & \\ \text{kde} & \displaystyle\sum_{i=0}^{n} y_i \alpha_i = 0, & 0 \leq \alpha_i \leq C \end{array} . \tag{6.1}$$

Zde se nám již ukazuje první problém, který je třeba vyřešit, a sice, že optimalizační algoritmus kvadratického programování v balíku `CVXOPT.qp` je implementován tak, aby řešil minimalizační problémy. Musíme tedy převést problém 6.1 na ekvivalentní minimalizační problém:

$$\begin{array}{lll} \underset{\alpha}{\text{minimalizace}} & \displaystyle\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i & \\ \text{kde} & \displaystyle\sum_{i=0}^{n} y_i \alpha_i = 0, & 0 \leq \alpha_i \leq C \end{array} . \tag{6.2}$$

V dokumentaci balíku `CVXOPT` je definován formát kvadratických problémů, které je modul `qp` schopen řešit. Vypadá takto:

$$\begin{array}{ll} \underset{\alpha}{\text{minimalizace}} & \frac{1}{2} \alpha^T P \alpha + q^T \alpha \\ \text{kde} & G\alpha \leq h \\ & A\alpha = b \end{array} . \tag{6.3}$$

Museli jsme tedy určit, jaké hodnoty dosadit za parametry P (kvadratické koeficienty), q (lineární koeficienty), G a h (podmínky nerovnosti), A a b (podmínky rovnosti).

Za proměnnou P jsme dosadili matici

$$P = \begin{bmatrix} y_1 y_1\ x_1^T x_1 & y_1 y_2\ x_1^T x_2 & y_1 y_3\ x_1^T x_3 & \cdots & y_1 y_N\ x_1^T x_N \\ y_2 y_1\ x_2^T x_1 & y_2 y_2\ x_2^T x_2 & y_2 y_3\ x_2^T x_3 & \cdots & y_2 y_N\ x_2^T x_N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_N y_1\ x_N^T x_1 & y_N y_2\ x_N^T x_2 & y_N y_3\ x_N^T x_3 & \cdots & y_N y_N\ x_N^T x_N \end{bmatrix}, \tag{6.4}$$

kde platí, že $x_N \in X$ a $y_N \in Y$. Tato matice však platí pouze v případě, že nepoužíváme jaderných funkcí pro vytvoření nelineární diskriminační funkce. Jestliže však používáme jadernou funkci $K(x_1, x_2)$, počítající skalární součin v prostoru příznaků (viz kapitola 4.2.1), potom musíme nahradit skalární součin $x_m^T x_n$ za tuto jadernou funkci, do níž dosadíme jednotlivé vektory. Matice P při použití jaderných funkcí bude tedy vypadat následovně:

$$P = \begin{bmatrix} y_1 y_1\ K(x_1, x_1) & y_1 y_2\ K(x_1, x_2) & y_1 y_3\ K(x_1, x_3) & \cdots & y_1 y_N\ K(x_1, x_N) \\ y_2 y_1\ K(x_2, x_1) & y_2 y_2\ K(x_2, x_2) & y_2 y_3\ K(x_2, x_3) & \cdots & y_2 y_N\ K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_N y_1\ K(x_N, x_1) & y_N y_2\ K(x_N, x_2) & y_N y_3\ K(x_N, x_3) & \cdots & y_N y_N\ K(x_N, x_N) \end{bmatrix}. \tag{6.5}$$

Dalším parametrem, který musíme kvadratickému programování poskytnout, je parametr lineárních koeficientů q. Tím bude pouze vektor stejné délky, jako je počet vstupních dat, obsahující hodnoty -1, tedy:

$$q = (-1, -1, \ldots, -1), \qquad |q| = |X| \tag{6.6}$$

Nyní se dostáváme k parametrům G, h a A, b. Tyto parametry, jak bylo již výše zmíněno, definují podmínky (*constrains*) optimalizačního procesu. Jak vyplývá z podmínek rovnice 6.2, musíme zadat jednu podmínku rovnosti, a sice:

$$\sum_{i=0}^{n} y_i \alpha_i = 0. \tag{6.7}$$

To je provedeno velmi jednoduchým způsobem tak, že proměnná A bude obsahovat vektor Y (třídy trénovacích dat) a proměnná b bude hodnota 0, tedy:

$$A\ = (y_1, y_2, \ldots, y_N), \qquad y_i \in Y; i \in |Y| \tag{6.8}$$

$$b = 0. \tag{6.9}$$

V programu je možné zvolit, zda chce uživatel používat proměnnou C (viz kapitola 4.3), tedy zda umožní porušovat maximální šířku (margin). Pokud se proměnná nepoužívá, pak má optimalizační problém následující tvar:

$$\begin{aligned} &\underset{\alpha}{\text{minimalizace}} && \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i \\ &\text{kde} && \sum_{i=0}^{n} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq \infty \end{aligned}. \tag{6.10}$$

Podíváme-li se podrobněji na omezení podmínky $\sum_{i=0}^{n} y_i \alpha_i = 0$, vidíme, že α může nabývat libovolné hodnoty od 0 do nekonečna, tudíž parametry podmínky nerovnosti budou následující:

$$G = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & -1 \end{bmatrix} \tag{6.11}$$

$$h = (0, 0, \ldots, 0), \qquad |h| = |X|. \tag{6.12}$$

Proměnné G a h (6.17 a 6.15) po vložení do kvadratického programování popisují právě podmínku $0 \leq \alpha_i \leq \infty$.

Jestliže ale uživatel zadá, že chce používat proměnnou C, situace se mírně komplikuje. Minimalizační úloha, kterou potřebujeme vyřešit, se nepatrně změní do následující podoby:

$$\begin{aligned} &\underset{\alpha}{\text{minimalizace}} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i \\ &\text{kde} \qquad \sum_{i=0}^{n} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq C \end{aligned} \quad . \tag{6.13}$$

Podmínka, jež v předchozím případě (bez C) nebyla shora nijak omezená, je najednou limitována hodnotou proměnné C. Parametr G a h se tedy oproti předchozímu případu bude muset rozšířit tak, aby pojal i pravou část podmínky. Bude tedy vypadat takto:

$$G = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & -1 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{6.14}$$

$$h = (0, 0, \ldots, 0, C, C, \ldots, C), \qquad |h| = 2 \cdot |X|. \tag{6.15}$$

Pokud tyto parametry tedy poskytneme modulu kvadratického programování, získáme výsledné hodnoty Lagrangeových multiplikátorů, které definují support vektory, jež se z trénovací množiny učením vytvořily. S těmito Lagrangeovými multiplikátory pak dále pracujeme a vytvoříme z nich klasifikační model (viz kapitola 4.3).

Optimalizace skalárních součinů v prostoru příznaků

Abychom docílili zrychlení učení SVM klasifikátoru, přistoupili jsme k výpočtu takzvané Gramovy matice. Gramova matice je matice G skalárních součinů vstupního vektoru, tedy $G_{i,j} = x_i^T x_j$ kde $x_i, x_j \in X$. Tím, že vypočteme tuto matici před samotným učením klasifikátoru, zamezíme zbytečnému několikanásobnému výpočtu kartézských součinů mezi stejnými vektory vstupních trénovacích dat, čímž se výrazně zrychlí výpočet.

Gramova matice pro lineární SVM klasifikátor vypadá takto:

$$G_{linear} = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 & \cdots & x_1^T x_N \\ x_2^T x_1 & x_2^T x_2 & x_2^T x_3 & \cdots & x_2^T x_N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^T x_1 & x_N^T x_2 & & x_N^T x_3 \cdots & x_N^T x_N \end{bmatrix}. \tag{6.16}$$

Gramovu matici lze samozřejmě použít i v případě použití jaderných funkcí. V takovém případě je pro výpočet Gramovy matice G použita, namísto standardního kartézského součinu v prostoru příznaků X, jaderná funkce, která ale vlastně představuje kartézský součin v prostoru F, do kterého lze prostor X převést pomocí určité transformační funkce φ (viz kapitola 4).

Gramova matice pro SVM klasifikátor, využívající jaderných metod, vypadá následovně:

$$G_{nonlinear} = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_3) & \cdots & K(x_1, x_N) \\ K(x_2, x_1) & K(x_2, x_2) & K(x_2, x_3) & \cdots & K(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ K(x_N, x_1) & K(x_N, x_2) & K(x_N, x_3) & \cdots & K(x_N, x_N) \end{bmatrix}. \tag{6.17}$$

V praktické části této práce je Gramova matice implementována ve tvaru $G_{nonlinear}$ a v závislosti na použité jaderné funkci (jaderná funkce může být i obyčejným kartézským součinem v prostoru příznaků X) je matice předpočítána pro pozdější použití při trénování klasifikátoru.

Kapitola 7

Testy a vyhodnocení implementovaných klasifikátorů

V této kapitole bude vyhodnocen námi implementovaný program. Nejprve popíšeme datové sady, které byly použity pro testování a porovnávání implementovaných klasifikátorů a poté se zaměříme na testování různých typů příznaků (viz kapitola 5).

Závěrem této kapitoly porovnáme oba implementované klasifikátory a shrneme výsledky, ke kterým jsme při vyhodnocování testů dospěli.

7.1 Data

Jak již bylo zmíněno v úvodu (viz kapitola 1.1), tato práce vznikla s cílem vytvořit klasifikátor textu pro projekt M-Eco [1], ve kterém participovala výzkumná skupina NLP, pracující při Fakultě informačních technologii VUT v Brně. Hlavním cílem projektu M-Eco bylo pomocí analýzy dat ze sociálních sítí, zpravodajských serverů, blogů a dalších zdrojů na internetu hledat možná obecná zdravotní ohrožení (epidemie, pandemie, atd.). Úkolem klasifikátoru v projektu M-Eco byla klasifikace vstupního textu do tříd a následné určení, zda bude vstupní text zapsán do databáze, nebo nebude. K tomuto účelu byla vytvořena datová sada pro trénování a testování Bayesovského klasifikátoru a SVM klasifikátoru.

Pro potřeby této práce byly však vytvořeny datové sady dvě, aby bylo možné implementované klasifikátory testovat na různých typech textů. Jedna datová sada byla vytvořena pro anglické tweety (textové příspěvky ze sociální sítě Twitter) a druhá pro anglické články (z internetových periodik). Zdrojem dat, ze kterých byly vytvořeny obě datové sady, byla databáze projektu M-Eco.

Datová sada obsahující anglické tweety je rozdělena do dvou tříd – **Obsahující zmínku o nemoci (buď nemoci pisatele, nebo nemoci někoho z pisatelova okolí)**, nebo **neobsahující takovou zmínku**. Tedy například tweet *'I have a huge headache'* nebo *'My dad feels sick'* je relevantní k tématu, naproti tomu *'Canadians should expect to see more severe cases of swine flu.'* je nerelevantní.

Pro datovou sadu obsahující anglické články je potom datová sada, podobně jako v případě tweetů, ručně anotována tak, že jsou vstupní články rozděleny do následujících dvou tříd – **zabývající se skutečnou nákazou ve světě**, nebo **nezabývající se skutečnou nákazou ve světě**. Tudíž například článek popisující vypuknutí meningitidy v USA je

[1] http://www.meco-project.eu/

relevantní k tématu, naproti tomu článek popisující průběh a léčbu meningitidy je nerelevantní.

Ke zvolení těchto tříd u obou datových sad vedl předpoklad, že i z nespecializovaných veřejně přístupných zdrojů lze získat informace o výskytu nemocí ve světě a postupu a rozšíření epidemií apod.

Jak vychází z odvození klasifikační rovnice, použité v Bayesovském klasifikátoru (viz vzorec 3.1.1), mělo by rozložení relevantních a nerelevantních dat v trénovací a testovací množině být zhruba v poměru 1:1. Celkově bylo ručně anotováno přibližně 4500 anglických tweetů a přibližně 500 anglických článků. V databázi anotovaných anglických tweetů se nachází 502 relevantních a zbytek nerelevantních záznamů. Pro databázi ručně anotovaných anglických článků je tento poměr 194 relevantních k 269 nerelevantním záznamům. Vzhledem k tomu, že v testu chceme porovnávat dva klasifikátory, z nichž jeden má vlastnost limitující jeho učící sadu, budeme pro oba klasifikátory používat identickou trénovací sadu – tzn. pro tweety bude datová sada pro testy obsahovat 502 relevantních a 502 nerelevantních záznamů a pro články bude datová sada obsahovat 194 relevantních a 194 nerelevantních záznamů.

Pro testování byla použita metoda *n-násobná křížová validace (n-fold cross-validation)*. Tato metoda rozděluje vstupní datovou sadu na n stejně velkých dílů. Jeden z těchto dílů je pak použit jako testovací datová sada a zbylých $n-1$ dílů je použito jako trénovací datová sada. Metoda sestává z n iterací, přičemž v každé iteraci se jako testovací datová sada použije jiná z n částí rozdělených dat. Jako celkový výsledek *n-násobné křížové validace* se použijí průměrné výsledky testů jednotlivých iterací.

Při testování v implementační části této práce byla použita metoda *5-ti násobné křížové validace*, což znamená, že data byla rozdělena na 5 stejných dílů a nad těmito díly se prováděly testy. 5-ti násobná křížová validace byla zvolena kvůli počítači, na kterém testy probíhaly. Validační proces byl totiž paralelizován – každá iterace v implementovaném programu využívá právě jedno jádro a výsledky všech jader jsou po doběhnutí všech iterací zprůměrovány do jednoho výsledku.

7.2 Testovací metriky

Aby bylo možné nějakým způsobem objektivně porovnat různé výsledky klasifikátorů a také posléze porovnat klasifikátory mezi sebou, je třeba zavést vhodné metriky, popisující výsledky klasifikátoru. V následující části práce budou tyto použité metriky vysvětleny.

7.2.1 Korelace

Korelace je statistická metoda, která definuje vzájemný vztah mezi veličinami X a Y. Míru korelace určíme výpočtem Pearsonova korelačního koeficientu, nabývajícího hodnoty $< -1, 1 >$. Jestliže vypočtený Pearsonův korelační koeficient nabývá hodnoty -1, pak to značí, že veličiny X a Y jsou na sobě zcela nezávislé. Naopak nabývá-li Pearsonův korelační koeficient hodnoty 1, pak jsou na sobě veličiny přímo závislé.

Pearsonův korelační koeficient se vypočítá jako:

$$K(X,Y) = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2 - E(X)^2)}\sqrt{(E(Y^2) - E(Y)^2)}}, \tag{7.1}$$

kde X jsou klasifikátorem automaticky spočítané pravděpodobnosti a Y jsou pravděpodobnosti zadané uživatelem při anotaci (0.01 – nerelevantní vstupní text a 0.99 – relevantní

vstupní text).

Při testování klasifikátoru se počítá korelace mezi uživatelem zadanou hodnotou vstupního textu (při anotaci testovací množiny) a výsledkem klasifikátoru. Čím více se tedy korelační koeficient blíží hodnotě 1, tím lepší výsledek klasifikátoru představuje.

Korelaci je tedy možné počítat pouze u Bayesovského klasifikátoru, který přiřazuje každému klasifikovanému vstupu určitou pravděpodobnost (soft-klasifikace). U SVM klasifikátoru hodnota, která by přiřazovala pravděpodobnost náležitosti do určité třídy neexistuje (hard-klasifikace), a proto není možné korelaci spočítat.

7.2.2 Výsledek klasifikace

Abychom byli schopni přesně zjistit jak klasifikátor testovací množinu oklasifikoval, rozdělíme oklasifikované vstupní texty do čtyř skupin:

- *Pravdivě pozitivní (true positive)* – počet vstupních textů, které byly klasifikátorem správně zařazeny do relevantní třídy.
- *Pravdivě negativní (true negative)* – počet vstupních textů, které byly klasifikátorem správně zařazeny do nerelevantní třídy.
- *Falešně pozitivní (false positive)* – počet vstupních textů, které byly klasifikátorem špatně zařazeny do relevantní třídy.
- *Falešně negativní (false negative)* – počet vstupních textů, které byly klasifikátorem špatně zařazeny do nerelevantní třídy.

Toto rozdělení je výsledkem porovnání předpokládaného výstupu klasifikátoru s reálným výstupem implementovaného klasifikátoru. Je zřejmé, že čím méně záznamů je falešně negativních a falešně pozitivních, tím lépe klasifikátor funguje.

7.2.3 Celková přesnost, přesnost, pokrytí, F1-measure

Dalšími metrikami, pomocí nichž budeme moci porovnávat výsledky klasifikátorů, jsou funkce přesnost (precision), pokrytí (recall), celková přesnost (accuracy) a F1-measure. Pro všechny výše zmíněné metody porovnání klasifikátorů se předpokládá, že jsme schopni vypočítat hodnoty *pravdivě pozitivní*, *pravdivě negativní*, *falešně pozitivní* a *falešně negativní* (viz kapitola 7.2.2).

Celková přesnost

Nejjednodušší metrikou pro porovnávání přesnosti klasifikátorů je metrika celková přesnost. Tato metrika definuje poměr, v jakém jsou ve výsledcích zastoupeny správně klasifikované vstupy.

Lze ji jednoduše vypočítat následovně:

$$CP = \frac{pravdive\_pozit. + pravdive\_negat.}{pravdive\_pozit. + pravdive\_negat. + falesne\_pozit. + falesne\_negat.}.$$

Nevýhodou této metriky však je, že nebere v potaz počty záznamů v jednotlivých třídách.

Přesnost, pokrytí

$$Presnost = \frac{pravdive\_pozitivni}{pravdive\_pozitivni + falesne\_pozitivni}$$

$$Pokryti = \frac{pravdive\_pozitivni}{pravdive\_pozitivni + falesne\_negativni}.$$

Přesnost tedy můžeme definovat jako poměr správně oklasifikovaných relevantních záznamů vůči všem oklasifikovaným relevantním záznamům. Pokrytí je potom poměr správně oklasifikovaných relevantních záznamů vůči skutečně relevantním záznamům.

F1-measure

$$F1measure = 2 \cdot \frac{Presnost \cdot Pokryti}{Presnost + Pokryti}.$$

F1-measure je jedna z metrik nejčastěji používaných pro hodnocení klasifikátorů. De facto jde o geometrický průměr přesnosti a pokrytí. F1-measure nabývá hodnot $< 0, 1 >$, kde 0 je nejhorší skóre popisující výsledek klasifikátoru a 1 je nejlepší.

7.3 Testy Bayesovského klasifikátoru

V následující části textu budou představeny výsledky testů Bayesovského klasifikátoru. Pomocí testů budeme hledat optimální nastavení klasifikátoru pro klasifikaci jednotlivých typů textu (anglické tweety a anglické články). V testech Bayesovského klasifikátoru se budeme zaměřovat především na porovnávání způsobů vytváření textových příznaků a výběr speciálních příznaků.

Pro testování klasifikace anglických tweetů byla použita datová sada manuálně anotovaných tweetů a pro testování klasifikace anglických článků byla použita datová sada manuálně anotovaných článků (viz kapitola 7.1).

7.3.1 Testy textových příznaků

Pro zjištění ideální délky n-tic textových příznaků jsme aplikovali testy pro postupně se zvyšující n, a to, dokud se zlepšovala hodnota Pearsonova korelačního koeficientu a F1-measure (viz kapitola 7.2).

Tweety

Jak již bylo výše v této kapitole zmíněno, pro trénování anglických tweetů je používána datová sada obsahující 502 relevantních a 502 nerelevantních tweetů.

Následující tabulka 7.1 prezentuje, jak se klasifikátor choval při změně nastavení maximální délky za sebou jdoucích n-tic textových příznaků:

n	1	2	3	4
Pravdivě pozitivní	83,2 (41,6%)	90,0 (45,0%)	90,0 (45,0%)	90,0 (45,0%)
Pravdivě negativní	93,8 (46,9%)	91,2 (45,6%)	91,2 (45,6%)	91,2 (45,6%)
Falešně pozitivní	6,2 (3,1%)	8,8 (4,4%)	8,8 (4,4%)	8,8 (4,4%)
Falešně negativní	16,8 (8,4%)	10,0 (5,0%)	10,0 (5,0%)	10,0 (5,0%)
Korelace	0,8188	0,8363	0,8363	0,8363
Přesnost	0,9306	0,9109	0,9109	0,9109
Pokrytí	0,832	0,9	0,9	0,9
Celková přesnost	0,885	0,906	0,906	0,906
F1-measure	0,8786	0,9054	0,9054	0,9054

Tabulka 7.1: Vliv různých délek n-tic textových příznaků na Bayesovský klasifikátor při klasifikaci tweetů.

Z výše uvedené tabulky jasně vyplývá, že nejlepším řešením pro klasifikaci tweetů je využít alternativní metodu tvorby textových příznaků s maximální délkou n-tic rovnou dvěma. Takovéto nastavení klasifikátoru výrazně zlepšilo klasifikační schopnosti Bayesovského klasifikátoru oproti ostatním variantám. Je zřejmé, že je-li délka maximálních n-tic větší než dva, Bayesovský klasifikátor klasifikuje stejně přesně jako při používání maximálních n-tic o délce rovnající se dvěma, avšak velikost klasifikačního slovníku Bayesovského klasifikátoru při tomto postupu zcela zbytečně roste.

Články

Obdobně jako v předchozím případě tweetů jsme testovali anglické články. V databázi ručně anotovaných anglických článků se nacházelo 194 relevantních a 269 nerelevantních záznamů. Jak již bylo v úvodu této kapitoly zmíněno, pro trénování a testování byla použita vyvážená sada 194 relevantních a 194 nerelevantních článků.

Byl použit stejný test jako v případě testu tvorby textových příznaků na datové sadě tweetů. Snažili jsme se určit optimální délku n-tic textových příznaků. Výsledky jednotlivých testů jsou zaznamenány v následující tabulce 7.2:

n	1	2	3	4
Pravdivě pozitivní	21,0 (28,4%)	21,4 (28,9%)	21,4 (28,9%)	21,4 (28,9%)
Pravdivě negativní	25,6 (34,6%)	29,4 (39,7%)	29,4 (39,7%)	29,4 (39,7%)
Falešně pozitivní	11,4 (15,4%)	7,6 (10,3%)	7,6 (10,3%)	7,6 (10,3%)
Falešně negativní	16,0 (21,6%)	15,6 (21,1%)	15,6 (21,1%)	15,6 (21,1%)
Korelace	0,3198	0,4286	0,4286	0,4286
Přesnost	0,6481	0,7379	0,7379	0,7379
Pokrytí	0,5676	0,5784	0,5784	0,5784
Celková přesnost	0,6297	0,6865	0,6865	0,6865
F1-measure	0,6052	0,6485	0,6485	0,6485

Tabulka 7.2: Vliv různých délek n-tic textových příznaků na Bayesovský klasifikátor při klasifikaci článků.

Je zřejmé, že výsledky testování maximální délky n-tic textových příznaků nad datovou sadou anglických článků dopadly prakticky stejně jako testy nad datovou sadou anglických tweetů. Délka n-tic rovna dvěma klasifikační schopnosti Bayesovského klasifikátoru zlepšila, avšak větší délka n-tic již na klasifikaci neměla výrazný vliv a zbytečně způsobila zvětšení velikosti slovníku Bayesovského klasifikátoru používaného pro klasifikaci.

Důvodem pro to, že využití delších n-tic již nezlepšuje klasifikační schopnosti Bayesovského klasifikátoru je zřejmě to, že tři za sebou jdoucí slova jsou již dosti specifickým příznakem, který se v textu nevyskytuje příliš často, a tudíž využití n-tic delších než dva již nepřispívá ke zlepšení klasifikačních schopností Bayesovského klasifikátoru. Kdyby se výrazně zvětšila trénovací a testovací datová sada, je možné, že by i delší n-tice zlepšovaly klasifikační schopnosti Bayesovského klasifikátoru.

7.3.2 Testy speciálních příznaků

Abychom mohli otestovat vliv jednotlivých typů speciálních příznaků na klasifikaci Bayesovského klasifikátoru, spustili jsme nejprve test klasifikátoru využívajícího pro klasifikaci pouze textových příznaků. Takto jsme získali kontrolní výsledek, který byl následně porovnán s výsledky klasifikace s využitím jednotlivých speciálních příznaků. V implementovaném programu byla vytvořena metoda, která vliv všech možných speciálních příznaků na klasifikaci porovná a najde jejich optimální kombinaci.

Nyní představíme, jaký vliv měl výběr optimálních speciálních příznaků na klasifikační schopnosti Bayesovského klasifikátoru. Pro každou datovou sadu bude prezentován výsledek bez speciálních příznaků a poté výsledek s vhodně vybranými speciálními příznaky.

Tweety

Pro datovou sadu anglických tweetů byl otestován vliv jednotlivých speciálních příznaků (viz kapitola 5.1) na výsledek klasifikace Bayesovského klasifikátoru. Následující tabulka 7.3 popisuje, jakou měrou jednotlivé příznaky zlepšovaly nebo zhoršovaly výsledek klasifikace Bayesovského klasifikátoru oproti kontrolnímu testu. Porovnáváme hodnoty korelace (kladná hodnota znamená zlepšení a záporná hodnota znamená zhoršení klasifikačních schopností testovaného Bayesovského klasifikátoru).

spec příznaky (viz kapitola 5.1)	Změna korelace
URL – celé	-0,0014
URL – doména	-0,0203
URL – existence A/N	-0,0148
URL – existence A	-0,0236
E-mail – celý	0
E-mail – existence A/N	0
E-mail – existence A	0
Emotikony – nalezené	-0,0029
Emotikony – existence A/N	-0,0025
Emotikony – existence A	-0,0028
Emotikony – nálada	0,0001
Tagy – celé	-0,0046
Tagy – existence A/N	-0,0002
Tagy – existence A	-0,0002
Tagy uživatelů – celé	-0,0021
Tagy uživatelů – existence A/N	0,0045
Tagy uživatelů – existence A	-0,0001
Věty – počet vět	-0,0049
Časy – celý čas	0
Časy – ve 24h formátu	0
Časy – hodiny ve 24h formátu	0
Data – celé datum	0
Data – ve formátu DMY	0
Data – ve formátu MY	0
Data – ve formátu Y	0

Tabulka 7.3: Vliv jednotlivých speciálních příznaků na změnu korelace výsledků Bayesovského klasifikátoru při klasifikaci tweetů.

Z tabulky 7.3 vyplývá, že optimální speciální příznaky pro datovou sadu anglických tweetů jsou následující:

- *Emotikony* – jako speciální příznak je použita nálada pisatele odvozená z nálady emotikonů nalezených v textu.
- *Tagy uživatelů* – je vytvořen speciální příznak *ANO*, jestliže je ve vstupním textu nalezen tag uživatele.

Ostatní skupiny speciálních příznaků (URL, e-mailové adresy, tagy, věty, časy a data) výsledky klasifikace Bayesovského klasifikátoru nad datovou sadou tweetů nezlepšily, a proto nebyly použity.

Následující tabulka 7.4 popisuje, jak se zlepší klasifikační schopnosti Bayesovského klasifikátoru při použití výše zmíněných optimálních speciálních příznaků při trénování a testování na datové sadě tweetů.

	Bez spec. příznaků	S optim. spec. příznaky
Pravdivě pozitivní	90,2 (45,1%)	90,2 (45,1%)
Pravdivě negativní	90,6 (45,3%)	91,2 (45,3%)
Falešně pozitivní	9,4 (4,7%)	8,8 (4,4%)
Falešně negativní	9,8 (4,9%)	9,8 (4,9%)
Korelace	0,8269	0,8341
Přesnost	0,9056	0,911
Pokrytí	0,902	0,902
Celková přesnost	0,904	0,907
F1-measure	0,9038	0,9065

Tabulka 7.4: Porovnání výsledků klasifikace tweetů pomocí Bayesovského klasifikátoru bez použití speciálních příznaků a s použitím speciálních příznaků.

Z tabulky 7.4 vyplývá, že optimálně zvolené speciální příznaky mohou mírně vylepšit klasifikační schopnosti Bayesovského klasifikátoru při klasifikaci anglických tweetů.

Články

Obdobně jako při klasifikaci tweetů i díky vhodnému výběru speciálních příznaků docházelo ke zlepšení klasifikačních schopností klasifikátoru, což popisuje následující tabulka. V tomto případě ale testy klasifikátoru proběhly nad datovou sadou anglických článků. V následující tabulce 7.5 je prezentováno do jaké míry jednotlivé speciální příznaky zlepšovaly, nebo zhoršovaly výsledek klasifikace oproti kontrolnímu výsledku. Opět porovnáváme hodnoty korelace (kladná hodnota znamená zlepšení a záporná hodnota znamená zhoršení klasifikačních schopností testovaného Bayesovského klasifikátoru).

spec příznaky (viz kapitola 5.1)	Změna korelace
URL – celé	0
URL – doména	-0,0142
URL – existence A/N	-0,002
URL – existence A	-0,0022
E-mail – celý	0,0026
E-mail – existence A/N	-0,0042
E-mail – existence A	-0,0044
Emotikony – nalezené	-0,0026
Emotikony – existence A/N	-0,0015
Emotikony – existence A	-0,0016
Emotikony – nálada	-0,0021
Tagy – celé	-0,0039
Tagy – existence A/N	-0,0051
Tagy – existence A	-0,0053
Tagy uživatelů – celé	0,0026
Tagy uživatelů – existence A/N	0,0008
Tagy uživatelů – existence A	0,0008
Věty – počet vět	-0,0364
Časy – celý čas	0
Časy – ve 24h formátu	-0,0103
Časy – hodiny ve 24h formátu	-0,0150
Data – celé datum	0,0180
Data – ve formátu DMY	0,024
Data – ve formátu MY	-0,0063
Data – ve formátu Y	0,0116

Tabulka 7.5: Vliv jednotlivých speciálních příznaků na změnu korelace výsledků Bayesovského klasifikátoru při klasifikaci tweetů.

Z tabulky 7.5 vyplývá, že byly zvoleny následující optimální speciální příznaky, které nejlépe napomáhají zlepšení klasifikačních schopností Bayesovského klasifikátoru při klasifikaci datové sady anglických článků:

- *E-mailové adresy* – celé e-mailové adresy nalezené ve vstupním textu byly brány jako speciální příznaky.
- *Tagy uživatelů* – celé tagy uživatelů nalezené ve vstupním textu byly brány jako speciální příznaky.
- *Data* – data nalezená ve vstupním textu byla převedena do formátu DMY a použita jako speciální příznaky.

Ostatní skupiny speciálních příznaků, které zde nebyly zmíněny (URL, emotikony, tagy, věty a časy), nezlepšily výsledky klasifikace Bayesovského klasifikátoru nad datovou sadou anglických článků, a proto nebyly použity.

	Bez spec. příznaků	S optim. spec. příznaky
Pravdivě pozitivní	21,4 (28,9%)	21,4 (28,9%)
Pravdivě negativní	29,4 (39,7%)	29,4 (39,7%)
Falešně pozitivní	7,6 (10,3%)	7,6 (10,3%)
Falešně negativní	15,6 (21,1%)	15,6 (21,1%)
Korelace	0,4024	0,4286
Přesnost	0,7379	0,7379
Pokrytí	0,5784	0,5784
Celková přesnost	0,6865	0,6865
F1-measure	0,6485	0,6485

Tabulka 7.6: Porovnání výsledků klasifikace článků pomocí Bayesovského klasifikátoru bez použití speciálních příznaků a s použitím speciálních příznaků.

Z tabulky 7.6 je zřejmé, že pro datovou sadu článků speciální příznaky sice o něco zlepšily korelaci klasifikátoru, nicméně na výsledcích klasifikace Bayesovského klasifikátoru se to nikterak neprojevilo. Takto malý vliv na zlepšení klasifikace byl při klasifikaci anglických článků způsoben zřejmě tím, že se ve článcích na rozdíl od tweetů vyskytuje daleko méně implementovaných speciálních příznaků, a to především proto, že články bývají na rozdíl od tweetů psány formálněji, a proto se tam například nevyskytují emotikony a podobně. Je možné, že při zvětšení datové sady pro učení by vliv speciálních příznaků na zlepšení klasifikačních schopností Bayesovského klasifikátoru mohl narůst.

7.4 Testy SVM klasifikátoru

Stejně jako při testování Bayesovského klasifikátoru, je SVM klasifikátor testován na dvou testovacích sadách anglických tweetů a anglických článků. Na rozdíl od Bayesovského klasifikátoru ale nemusíme volit jaké speciální příznaky budou pro klasifikaci využity (viz kapitola 5.4). Klasifikátoru umožníme nalézt si optimální řešení ze všech speciálních příznaků, které implementovaný program umožňuje vytvořit. Na druhou stranu stejně jako v Bayesovském klasifikátoru budeme hledat optimální délku n-tic textových příznaků, která by měla být taková, aby zlepšovala klasifikační schopnosti SVM klasifikátoru. Jestliže totiž bude zvolena příliš velká délka těchto maximálních n-tic, zbytečně se zvětší dimenzionalita příznaků a klasifikační schopnosti SVM klasifikátoru se sníží.

Veškeré testy SVM klasifikátoru byly prováděny za použití RBF jaderné funkce a optimální parametry SVM a jaderné funkce byly nalezeny pomocí metody simulovaného žíhání.

7.4.1 Tweety

Následující tabulka 7.7 popisuje klasifikační schopnosti SVM klasifikátoru pro jednotlivé maximální délky n-tic textových příznaků.

n	1	2	3
Pravdivě pozitivní	93,0 (46,5%)	94,0 (47,0%)	93,0 (46,5%)
Pravdivě negativní	92,8 (46,4%)	91,4 (45,7%)	91,8 (45,9%)
Falešně pozitivní	7,2 (3,6%)	8,6 (4,3%)	8,2 (4,1%)
Falešně negativní	7,0 (3,5%)	6,0 (3,0%)	7,0 (3,5%)
Přesnost	0,9281	0,9162	0,919
Pokrytí	0,93	0,94	0,93
Celková přesnost	0,9281	0,9151	0,9185
F1-measure	0,9291	0,9279	0,9245

Tabulka 7.7: Vliv různých délek n-tic textových příznaků na SVM klasifikátor při klasifikaci tweetů.

Z výše uvedené tabulky 7.7 jasně vyplývá, že klasifikační schopnosti SVM klasifikátoru se se zvyšováním maximální délky n-tic textových příznaků zhoršovaly, tudíž není vhodné tyto n-tice pro klasifikaci tweetů pomocí SVM klasifikátoru používat. Tento výsledek přisuzujeme tomu, že čím více roste velikost maximálních n-tic textových příznaků, tím se rozšiřuje počet dimenzí prostoru příznaků. To má potom za následek přeučení (=overfitting, viz kapitola 4.2.1).

7.4.2 Články

Stejně jako pro datovou sadu anglických tweetů jsme pro datovou sadu anglických článků provedli testy optimální délky n-tic textových příznaků. Následující tabulka 7.8 popisuje výsledky klasifikace SVM klasifikátoru pro jednotlivé testované délky n-tic textových příznaků.

n	1	2
Pravdivě pozitivní	24,4 (33,0%)	19,8 (26,8%)
Pravdivě negativní	32,0 (43,2%)	34,2 (46,2%)
Falešně pozitivní	5,0 (6,8%)	2,8 (3,8%)
Falešně negativní	12,6 (17,0%)	17,2 (23,2%)
Přesnost	0,8299	0,8761
Pokrytí	0,6594	0,5351
Celková přesnost	0,7621	0,7297
F1-measure	0,7349	0,6644

Tabulka 7.8: Vliv různých délek n-tic textových příznaků na SVM klasifikátor při klasifikaci tweetů.

Pro maximální délku n-tic rovnou třem již test klasifikátoru nedoběhl, jelikož test byl příliš náročný na RAM paměť počítače, na kterém testy běžely (potřeboval asi 25GB paměti). Nicméně vzhledem k výsledkům testů na datové sadě tweetů můžeme předpokládat, že vliv maximální délky n-tic textových příznaků na klasifikační schopnosti SVM klasifikátoru bude v případě článků stejný, jako v případě datové sady obsahující tweety. Z tabulky 7.8 je tedy stejně jako při klasifikaci tweetů jasné, že nejlepších klasifikačních výsledků dosahuje SVM klasifikátor když jsou používány pouze základní jednoslovné textové příznaky.

7.5 Porovnání Bayesovského klasifikátoru a SVM klasifikátoru

V této části kapitoly přistoupíme k porovnání obou implementovaných klasifikátorů. Pro oba klasifikátory vybereme optimální nastavení příznaků, které jsme získali pomocí testů popsaných výše v této kapitole a porovnáme výsledky klasifikace prováděné oběma klasifikátory na obou vytvořených datových sadách, tedy jak na datové sadě anglických tweetů, tak na datové sadě anglických článků.

Následující tabulka 7.9 popisuje nejlepší výsledky Bayesovského klasifikátoru a SVM klasifikátoru na datové sadě tweetů:

Klasifikátor	Bayesovský klasifikátor	SVM klasifikátor
Pravdivě pozitivní	90,2 (45,1%)	93,0 (46,5%)
Pravdivě negativní	91,2 (45,3%)	92,8 (46,4%)
Falešně pozitivní	8,8 (4,4%)	7,2 (3,6%)
Falešně negativní	9,8 (4,9%)	7,0 (3,5%)
Přesnost	0,9111	0,9281
Pokrytí	0,902	0,93
Celková přesnost	0,907	0,9281
F1-measure	0,9065	0,9291

Tabulka 7.9: Vliv různých délek n-tic textových příznaků na SVM klasifikátor při klasifikaci tweetů.

Bayesovský klasifikátor používal n-tice textových příznaků o maximální délce rovné dvěma a optimální speciální příznaky popsané výše v této kapitole v testech speciálních příznaků Bayesovského klasifikátoru. SVM klasifikátor používal všechny možné speciální příznaky a n-tice textových příznaků o maximální délce rovné jedné – tzn. pouze jednotlivá slova.

Jak z tabulky 7.9 jasně vyplývá, klasifikační schopnosti SVM klasifikátoru na datové sadě anglických tweetů jasně předčí Bayesovský klasifikátor naučený na stejné datové sadě, využívající optimální kombinace speciálních příznaků.

Nyní opět porovnáme Bayesovský klasifikátor s SVM klasifikátorem, tentokrát však ale na datové sadě anglických článků. Následující tabulka 7.10 popisuje toto porovnání klasifikátorů:

Klasifikátor	Bayesovský klasifikátor	SVM klasifikátor
Pravdivě pozitivní	21,4 (28,9%)	24,4 (33,0%)
Pravdivě negativní	29,4 (39,7%)	32,0 (43,2%)
Falešně pozitivní	7,6 (10,3%)	5,0 (6,8%)
Falešně negativní	15,6 (21,1%)	12,6 (17,0%)
Přesnost	0,7379	0,8299
Pokrytí	0,5784	0,6594
Celková přesnost	0,6865	0,7621
F1-measure	0,6485	0,7349

Tabulka 7.10: Vliv různých délek n-tic textových příznaků na SVM klasifikátor při klasifikaci tweetů.

Stejně jako při testech prováděných na datové sadě tweetů, používal Bayesovský klasifikátor n-tice textových příznaků o maximální délce rovné dvěma a vhodně zvolené speciální příznaky. SVM klasifikátor používal všechny možné speciální příznaky a n-tice textových příznaků o maximální délce rovné jedné – tzn. pouze jednotlivá slova z textu.

Stejně jako v případě klasifikace datové sady tweetů jsou při klasifikaci anglických článků klasifikační schopnosti SVM klasifikátoru lepší než klasifikační schopnosti Bayesovského klasifikátoru. V případě anglických článků je ale rozdíl mezi výsledky klasifikace Bayesovského klasifikátoru a SVM klasifikátoru velký – Bayesovský klasifikátor dosáhl ohodnocení pomocí metriky F1-measure rovné 0,6485 a SVM klasifikátor dosáhl ohodnocení 0,7349, což je poměrně velký rozdíl ve prospěch SVM klasifikátoru.

7.6 Shrnutí výsledků testů

Pro Bayesovský klasifikátor bylo naším cílem v případě textových příznaků experimentálně nalézt optimální délku n-tic textových příznaků a v případě speciálních příznaků zvolit takové speciální příznaky, které co nejvíce napomáhají zlepšení klasifikačních schopností implementovaného Bayesovského klasifikátoru při klasifikaci datových sad vytvořených pro testování. U obou datových sad jsme zjistili, že optimální délka n-tic textových příznaků je rovna dvěma a že textové příznaky mají při vhodné volbě délky n-tic textových příznaků poměrně velký vliv na zlepšení klasifikačních schopností Bayesovského klasifikátoru. Speciální příznaky už takový vliv na zlepšení klasifikačních schopností Bayesovského klasifikátoru neměly, nicméně v případě klasifikace tweetů klasifikační schopnosti klasifikátoru mírně vylepšily. Při klasifikaci anglických článků neměly speciální příznaky na klasifikaci prakticky žádný vliv.

V testech prováděných s klasifikátorem SVM bylo naším cílem, podobně jako v případě Bayesovského klasifikátoru, optimálně zvolit maximální délku n-tic textových příznaků, která by napomáhala zlepšit klasifikační schopnosti SVM klasifikátoru. Z výsledků testů vyplývá, že s rostoucí délkou n-tic textových příznaků klasifikační schopnosti SVM klasifikátoru klesají. Z toho důvodu je optimální pro SVM klasifikátor používat jako textové příznaky pouze jednotlivá slova, nikoliv však n-tice za sebou jdoucích slov.

V poslední, třetí části této kapitoly, bylo představeno porovnání klasifikačních schopností obou implementovaných klasifikátorů při klasifikaci obou vytvořených testovacích datových sad. Pro porovnávání klasifikátorů bylo použito optimálních parametrů pro příslušné datové

sady k nastavení obou klasifikátorů, vycházející z předchozích testů v této kapitole. Z provedených porovnání SVM a Bayesovského klasifikátoru je jasně patrné, že SVM klasifikátor je lepší metodou pro klasifikaci, která při testování na dvou velmi odlišných testovacích sadách dává v porovnání s Bayesovským klasifikátorem lepší výsledky v případě klasifikace anglických tweetů a výrazně lepší výsledky v případě klasifikace anglických článků.

Kapitola 8

Závěr

V této diplomové práci jsme se zaměřili na dvě klasifikační metody pro klasifikaci textu – SVM klasifikátor a Bayesovský klasifikátor. Obě tyto metody jsme podrobně analyzovali, popsali jejich matematický model, implementovali a poté porovnali jejich klasifikační schopnosti na různých datových sadách.

V celé diplomové práci jsme kladli velký důraz na praktickou využitelnost implementovaných klasifikátorů. Bayesovský klasifikátor byl dlouhou dobu používán v projektu M-Eco, ve kterém participovala výzkumná skupina NLP, pracující při Fakultě informačních technologii VUT v Brně. Hlavním cílem projektu M-Eco bylo pomocí analýzy dat ze sociálních sítí, zpravodajských serverů, blogů a dalších zdrojů na internetu hledat možná obecná zdravotní ohrožení (epidemie, pandemie, atd.). Úkolem klasifikátoru v projektu M-Eco byla klasifikace vstupního textu do tříd a následné určení, zda bude vstupní text zapsán do databáze, nebo nebude. SVM klasifikátor implementovaný v této diplomové práci však již bohužel při práci na tomto projektu použit nemohl být, a to z důvodu ukončení spolupráce na projektu M-Eco v srpnu roku 2012.

Abychom co největší měrou pozitivně ovlivnili klasifikační schopnosti obou implementovaných klasifikátorů, bylo dbáno na výběr vhodných příznaků, extrahovaných z klasifikovaného textu. Tyto příznaky jsme rozdělili do dvou skupin – na textové a speciální příznaky. Textové příznaky byly vytvářeny ze slov, nacházejících se v klasifikovaném textu. Do speciálních příznaků potom patřily ostatní netextové příznaky, které byly z textu získány – např. emotikony, počet slov ve větě, e-mailové adresy atd. (viz kapitola 5). Příznaky jsme volili tak, aby jejich následné zpracování klasifikátory mělo co nejlepší vliv na výsledné klasifikační schopnosti implementovaných klasifikátorů.

Pro SVM klasifikátor bylo třeba mimo výběru optimálních příznaků určit optimální nastavení volných parametrů klasifikátoru a použité jaderné funkce (viz kapitola 4, 4.2.1). Pro zjištění těchto parametrů jsme použili algoritmus simulovaného žíhání.

Pro testování obou implementovaných klasifikátorů jsme anotovali dvě testovací datové sady (datová sada obsahující tweety – zprávy ze sociální sítě Twitter a datová sada anglických článků). Pro oba klasifikátory jsme se snažili zvolit optimální textové a speciální příznaky, s cílem co nejvíce vylepšit klasifikační schopnosti implementovaného SVM a Bayesovského klasifikátoru. V kapitole o testech jsme pro obě námi vytvořené datové sady a pro oba námi vytvořené klasifikátory optimální příznaky nalezli. V závěru jsme využili nalezené optimální nastavení obou klasifikátorů pro klasifikaci obou námi vytvořených testovacích datových sad a porovnali jsme jejich klasifikační schopnosti. Z výsledků tohoto porovnání plyne, že SVM klasifikátor dosahuje při klasifikaci obou datových sad lepší přesnosti klasifikace než Bayesovský klasifikátor (viz kapitola 7.5).

Možnou cestou, jak dále pokračovat v práci na tomto tématu a navázat na poznatky získané v této práci, by bylo vytvoření větší sady speciálních příznaků a následné otestování jejich vlivu na klasifikační schopnosti obou námi implementovaných klasifikátorů. S velmi dobrými výsledky by zřejmě také mohla být pro oba implementované klasifikátory použita metoda zvaná boosting, popsaná v článku Michaela Kearnse s názvem „Thoughts on hypothesis boosting“ [15] (viz také kapitola 2.4.3). Dalším námětem pro případné rozšíření této práce by mohlo být využití nové implementace SVM klasifikátoru nazavané v-SVM popsané v článku „New Support Vector Algorithms“ [21], které by mohlo přinést vylepšení klasifikačních schopností SVM klasifikátoru.

Literatura

[1] Support Vector Machines (SVMs).
URL http://svms.org/

[2] Abu-Mostafa, Y. S.: Machine Learning online course (MOOC). 2012.
URL http://work.caltech.edu/telecourse.html

[3] Abu-Mostafa, Y. S.; Magdon-Ismail, M.; Lin, H.-T.: *Learning From Data*. AMLBook, 2012, ISBN 1600490069, 9781600490064.

[4] del Amo, A.; Montero, J.; Cutello, V.: On the principles of fuzzy classification. In *Fuzzy Information Processing Society, 1999. NAFIPS. 18th International Conference of the North American*, 1999, s. 675–679, doi:10.1109/NAFIPS.1999.781779.

[5] Breiman, L.; aj.: *Classification and Regression Trees*. New York: Chapman & Hall, 1984, ISBN 0-412-04841-8, 358 s., new edition of [**?**]?
URL http://www.crcpress.com/catalog/C4841.htm

[6] Burges, C. J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.*, ročník 2, č. 2, Červen 1998: s. 121–167, ISSN 1384-5810, doi:10.1023/A:1009715923555.
URL http://dx.doi.org/10.1023/A:1009715923555

[7] Cortes, C.; Vapnik, V.: Support-Vector Networks. In *Machine Learning*, 1995, s. 273–297.

[8] Drozdov, V. V.: Automatic generation of rules for a system that uses a grammatical approach to syntactic analysis. *Autom. Doc. Math. Linguist.*, ročník 44, June 2010: s. 121–126, ISSN 0005-1055, doi:http://dx.doi.org/10.3103/S0005105510030039.
URL http://dx.doi.org/10.3103/S0005105510030039

[9] Elkan, C.: Boosting and naive Bayesian learning. *proceeding of KDD97 New Port beach CA*, 1997.

[10] Freund, Y.; Schapire, R. E.: A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. 1995.

[11] Freund, Y.; Schapire, R. E.: Large Margin Classification Using the Perceptron Algorithm. *Mach. Learn.*, ročník 37, č. 3, Prosinec 1999: s. 277–296, ISSN 0885-6125, doi:10.1023/A:1007662407062.
URL http://dx.doi.org/10.1023/A:1007662407062

[12] Gabriel, R. P.: Deliberate Writing. In *Natural Language Generation Systems*, editace D. D. McDonald; L. Bolc, New York: Springer, 1988, s. 1–46.

[13] Goldberg, Y.; Elhadad, M.: splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio: Association for Computational Linguistics, June 2008, s. 237–240.
URL http://www.aclweb.org/anthology/P/P08/P08-2060

[14] Joo, S.-W.: Linear Discriminant Functions and SVM. 1995.

[15] Kearns, M.: Thoughts on hypothesis boosting. *Unpublished manuscript*, 1988.

[16] Lin, S.-W.; Lee, Z.-J.; Chen, S.-C.; aj.: Parameter determination of support vector machine and feature selection using simulated annealing approach. *Appl. Soft Comput.*, ročník 8, č. 4, 2008: s. 1505–1512.
URL http://dblp.uni-trier.de/db/journals/asc/asc8.html#LinLCT08

[17] MALIK, F.: *Extrakce informací z hypertextu*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, 2007.
URL http://is.muni.cz/th/60743/fi_m/

[18] Manning, C. D.; Schütze, H.: *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press, 1999, ISBN 0-262-13360-1.

[19] Marek, T.: klasifikace dokumentů podle tématu, semestrální projekt. Brno: VUT v Brně, 2011.

[20] Medlock, B.: An Introduction to NLP-based Textual Anonymisation. 2006.

[21] Schölkopf, B.; Smola, A. J.; Williamson, R.; aj.: New Support Vector Algorithms. 1998.

[22] Sewell, M.: Structural Risk Minimization. 2008.
URL http://www.svms.org/srm/

[23] Taskar, B.; Segal, E.; Koller, D.: Probabilistic Classification and Clustering in Relational Data. In *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, editace B. Nebel, Seattle, US, 2001, s. 870–878.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.9319

[24] Turing, A. M.: Computing Machinery and Intelligence. 1950, one of the most influential papers in the history of the cognitive sciences:
http://cogsci.umn.edu/millennium/final.html.
URL http://cogprints.org/499/

[25] Vapnik, V.: *The Nature of Statistical Learning Theory*. Information Science and Statistics, Springer, 2000, ISBN 9780387987804.
URL http://books.google.cz/books?id=sna9BaxVbj8C

[26] Vapnik, V. N.: *Statistical Learning Theory*. Wiley-Interscience, 1998.

[27] Vijayabhanu, R.; Radha, V.: Statistical Normalization techniques for the prediction of COD level for an anaerobic wastewater treatment plant. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, New York, NY, USA: ACM, 2012, ISBN

978-1-4503-1310-0, s. 232–236, doi:10.1145/2393216.2393256.
URL http://doi.acm.org/10.1145/2393216.2393256

Dodatek A

Obsah CD

K diplomové práci je přiloženo CD. Toto CD obsahuje:

- Zdrojové kódy programu s implementovaným SVM a Bayesovským klasifikátorem. (Adresář `/program/`)
- Ručně anotované testovací a trénovací datové sady (anglické tweety a anglické články). (Adresář `/data/`)
- Dokumentaci k programu vygenerovanou pomocí nástroje Epydoc. (Adresář `/doc/`)
- Zdrojové kódy technické zprávy a její verzi ve formátu PDF. (Adresář `/thesis/`)

Dodatek B

Manuál

V této příloze bude popsán způsob práce s implementovaným programem.

Oba implementované klasifikátory (Bayesovský klasifikátor a SVM klasifikátor) jsou ovládány pomocí spouštěcího skriptu, nazvaného `control.py`. Tento skript se nachází v adresáři `/program/` na přiloženém CD a umožní uživateli s implementovanými klasifikátory jak provádět testy, tak vytvořit klasifikační modely a přímo s těmito modely provádět klasifikaci.

Ovládací skript obou klasifikátorů je schopen pracovat buď s jednotlivými klasifikátory samostatně (`./control.py bayes [parametry]`, `./control.py svm [parametry]`), nebo oba klasifikátory porovnat (`./control.py common [parametry]`). V případě práce s jednotlivými klasifikátory je možné s každým klasifikátorem provést několik možných operací, které nyní popíšeme. Následně popíšeme volbu `common` pro porovnání obou implementovaných klasifikátorů.

B.1 Bayesovský klasifikátor

Seznam techto operaci lze vypsat spuštěním následujícího příkazu: `./control.py bayes -h`. Detailnější popis jednotlivých operací s Bayesovským klasifikátorem lze vypsat spuštěním příkazu: `./control.py bayes [operace] -h`, kde namísto slova `[operace]` napíšeme název požadované operace. Možné operace proveditelné s Bayesovským klasifikátorem jsou:

- `test` – spustí test Bayesovského klasifikátoru s danými parametry.
- `features` –spustí proces vyhledávání optimálních speciálních příznaků.
- `model` – vytvoří klasifikační model pro Bayesovský klasifikátor.
- `classify` – klasifikuje daný vstupní text za použití klasifikačního modelu pro Bayesovský klasifikátor.

B.2 SVM klasifikátor

Stejně jako v případě Bayesovského klasifikátoru lze i pro SVM klasifikátor vypsat seznam operací, které lze s SVM klasifikátorem provádět. K tomuto výpisu slouží následující příkaz: `./control.py svm -h`. Pro detailnější popis jednotlivých operací SVM klasifikátoru je

třeba spustit příkaz (`./control.py svm [operace] -h`), kde namísto slova `[operace]` napíšeme název požadované operace. Implementovaný program umožňuje s implementovaným SVM klasifikátorem provádět následující operace:

- `data` – tato operace vygeneruje vnitřní reprezentaci datové sady, se kterou klasifikátor SVM následně pracuje. Tato operace je prováděna samostatně kvůli její časové náročnosti. Před prací s SVM klasifikátorem je třeba tento příkaz vždy spustit.
- `test` – spustí test SVM klasifikátoru s danými parametry.
- `annealing` – spustí proces simulovaného žíhání, který má za úkol hledat optimální nastavení volných parametrů SVM klasifikátoru a použité jaderné funkce. Před spuštěním tohoto procesu by měla být vygenerována data.
- `model` – vytvoří klasifikační model pro SVM klasifikátor.
- `classify` – klasifikuje daný vstupní text za použití klasifikačního modelu pro SVM klasifikátor.

B.3 Porovnání klasifikátorů

Tato volba slouží pro porovnání obou implementovaných klasifikátorů při testech se stejnými podmínkami – ideální volba příznaků, stejná délka používaných n-tic textových příznaků a stejná datová sada. Tento proces může být v závislosti na zvolených parametrech velmi časově i paměťově náročný.

Podrobnější popis parametrů se kterými se program spouští se nachází v nápovědě programu, kterou lze zobrazit spuštěním příkazu `./control.py common -h`.

Dodatek C

Regulární výrazy speciálních příznaků

V této příloze budou vypsány regulární výrazy, které jsou používány pro získávání speciálních příznaků z textu. Regulární výrazy budou pro přehlednost popsány ve formátu, v jakém je přijímá programovací jazyk Python.

C.1 URL

Regulární výraz pro vyhledávání URL v textu:

```
urls_re = re.compile(
    r'(https?|s?ftp)(://)(www\.)?([\w\.-]+\.[a-zA-Z]{2,4})(:\d*)?' +
    r'(/[-_~$.+!*\'()\[\],;:@&=\?/~#%\w#]*)[^\.\,\)\(\s]' +
    r'|' +
    r'(www\.)([\w\d\.-]+\.[a-zA-Z]{2,4})(:\d*)?' +
    r'(/[-_~$.+!*\'()\[\],;:@&=\?/~#%\w#]*)[^\.\,\)\(\s]'
)
```

C.2 E-mailové adresy

Regulární výraz pro vyhledávání e-mailových adres v textu:

```
emails_re = re.compile(r'\s([\w_\.-]+)(@)([\w\.-]+)(\.)([a-zA-Z]{2,6})')
```

C.3 Emotikony

V případě emotikonů není regulární výraz tak jednoduchý jako v ostatních případech, a to z toho důvodu, že je třeba rozdělovat emotikony do tříd podle nálady. Proto jsou definovány základní stavební kameny, ze kterých se emotikony skládají:

- Horní část emotikonu –
  ```
  top = r'[<>O}\]3P]?'
  ```
- Oči emotikonu –
  ```
  eyes = r'[:=Xx;]'
  ```

- Slzy emotikonu –
  ```
  tear = r'\'?'
  ```
- Nos emotikonu –
  ```
  nose = r'[oO-]?'
  ```
- Ústa emotikonu –
  ```
  mouths = r'[dD\)\]/S\(\[\\\|pPoOcC@{}3&]'
  ```
- Smutná ústa emotikonu –
  ```
  sad_mouths = r'[/S\(\[\\\|cC@{]'
  ```
- Veselá ústa emotikonu –
  ```
  happy_mouths = r'[dD\)\]pP}3]'
  ```
- Ostatní ústa emotikonu –
  ```
  other_mouths = r'[oO&]'
  ```

Z těchto stavebních kamenů jsou potom skládány dohromady jednotlivé typy emotikonů:

- Všechny emotikony –
  ```
  faces = ''.join([top, eyes, tear, nose, mouths])
  ```
- Smutné emotikony –
  ```
  sad_faces = ''.join([top, eyes, tear, nose, sad_mouths])
  ```
- Veselé emotikony –
  ```
  happy_faces = ''.join([top, eyes, tear, nose, happy_mouths])
  ```
- Ostatní emotikony –
  ```
  other_faces = ''.join([top, eyes, tear, nose, other_mouths])
  ```

Mimo tyto klasické emotikony jsou také jako speciální příznaky brány méně standardní (ostatní) emotikony:

- Všechny ostatní emotikony –
  ```
  other_emoticons = [
      r'\^_+\^', r'o\.O', r'@_+@',
      r'-_+-', r'\.\.+', r',,+', r'<3'
  ]
  ```
- Ostatní smutné emotikony –
  ```
  other_sad_emoticons = [r'-_+-']
  ```
- Ostatní veselé emotikony –
  ```
  other_happy_emoticons = [r'\^_+\^', r'<3']
  ```
- Ostatní ostatní emotikony –
  ```
  other_other_emoticons = [r'o\.O', r'@_+@', r'\.\.+', r',,+']
  ```

Z těchto výše zmíněných složek jsou potom vytvořeny regulární výrazy, popisující jednotlivé typy emotikonů, které lze nalézt v textu. Regulární výrazy jsou složeny následovně:

- Všechny emotikony –
```
emoticons_re = re.compile(
    '\s(' + '|'.join(other_emoticons + [faces]) + ')\s'
)
```

- Smutné emotikony –
```
sad_emoticons_re = re.compile(
    '\s(' + '|'.join(other_sad_emoticons + [sad_faces]) + ')\s'
)
```

- Veselé emotikony –
```
happy_emoticons_re = re.compile(
    '\s(' + '|'.join(other_happy_emoticons + [happy_faces]) + ')\s'
)
```

- Ostatní emotikony –
```
other_emoticons_re = re.compile(
    '\s(' + '|'.join(other_other_emoticons + [other_faces]) + ')\s'
)
```

C.4 Tagy

Regulární výraz pro vyhledávání Twitter tagů v textu:
`tags_re = re.compile(r'#\w+')`

C.5 Tagy uživatelů

Regulární výraz pro vyhledávání tagů uživatelů v textu:
`tags_re = re.compile(r'@\w+')`

C.6 Věty

Speciální příznak specifikující počet vět daného vstupního textu není získáván za pomocí regulárních výrazů, ale je vytvářen při rozdělování textu na věty pomocí knihovny `nltk.punkt`.

C.7 Časy

Regulární výraz pro vyhledávání časových údajů v textu:
```
time_re = re.compile(
    r'([0-2][0-9])(:)([0-9]{2})(\s*)(am|pm)?',
    flags=re.IGNORECASE)
)
```

C.8 Data

Regulární výraz pro vyhledávání dat v textu:

```
date_re = re.compile(
    r'(\d{1,2})([./-])(\d{1,2})([./-])(\d{2,4})' +
    r'|' +
    r'(\d{1,2})([./\s-])' +
    r'(jan|feb|mar|apr|jun|jul|aug|sep|oct|nov|dec)' +
    r'(uary|ruary|ch|il|e|ly|ust|tember|ober|ember)?'
    r'(\s)(\d{1,4})', flags=re.IGNORECASE
)
```