Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

DIPLOMOVÁ PRÁCE

Plzeň 2013 Michal Patočka

Západočeská univerzita

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Metody strojového učení pro analýzu sentimentu

Plzeň, 2013 Michal Patočka

Poděkování

Michalu Konkolovi za knihovnu s algoritmy Machine Learning.

Ivanu Habernalovi za implementaci Confusion Matrix a za skvělé vedení práce.

The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations"(LM2010005) is highly appreciated.

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

Plzeň, 16 Květen 2013 Michal Patočka, ...

Abstrakt

Analýza sentimentu
Tato práce prozkoumává možnosti použití algoritmů strojového učení pro analýzu sentimentu v českém jazyce. Prozkoumávané techniky jsou testovány na datech získaných ze serveru heureka.cz. Kromě použití tří základních algoritmů strojového učení (Naive Bayes, Maximum Entropy a SVM), je představeno a implementováno pět metod selekce příznaků (Mutual information, Information gain, Chi square, Odds ratio a Relevancy score). V rámci několika navržených experimentů je prozkoumán vliv těchto algoritmů na úspěšnost klasifikace algoritmů strojového učení.

Klíčová slova: sentiment analysis, machine learning, feature selection, mutual information, information gain, chi square, odds ratio, relevancy score, naive bayes, maximum entropy, svm, confusion matrix, imbalanced dataset

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Analýza sentimentu je jedním z úkolů, který spadá pod širší problematiku zpracování přirozeného jazyka (NLP- natural language processing). Hlavní myšlenkou této analýzy je určit tzv. sentiment poskytnutého textu. Pod tímto pojmem si lze představit postoj tvůrce textu k určité otázce. Druhů sentimentu může být obecně celá řada (Tsytsarau and Palpanas, 2012) a závisí na původu zpracovávaného textu. Nejjednodušší kategorizaci však lze provést pouze do tří tříd. Libovolný text tak můžeme klasifikovat podle jeho sentimentu takto na pozitivní negativní a neutrální.

Někdy je ještě zvlášť uváděna kategorie bipolární (mající pozitivní i negativní sentiment), tu lze ale špatně vyjádřit na jakékoliv stupnici a proto bývá často zanedbávána. V této práci je z tohoto důvodu dále předpokládán princip rozdělování textu pouze do tří výše uvedených tříd, ty ovšem nejsou jediným hlediskem, podle kterého lze sentiment textu členit. Nejprve je nutno určit, jestli zadaný text vůbec nějaký sentiment obsahuje. Rozlišujeme proto další dvě třídy - objektivní a subjektivní.

Do objektivní kategorie spadá obecné vyjádření, prosté jakýchkoliv emocí a postojů pisatele. Nás ovšem zajímá vyjádření objektivní, které lze klasifikovat do výše uvedených tří kategorií. Tento přístup lze nicméně dále zjednodušit sloučením kategorie neutrálních a objektivních vyjádření. Toto ulehčení nám

ušetří jinak potřebnou dvouúrovňovou klasifikaci dokumentu a umožní jednokrokový přístup bez přílišné ztráty na obecnosti. Ve výsledku nás totiž zajímají především kategorie pozitivní a negativní.

Sentiment lze dále členit podle velikosti zpracovávaného textu. Můžeme určovat sentiment jednotlivých slov, vět, ale i celých odstavců nebo příspěvků. Příznaky mohou být rozdílných druhů a může být velice těžké je správně vybrat. Této problematice se proto věnuje samostatná kapitola této práce (viz kapitola 2.1).

K tomu abychom mohli vytvořit automatický klasifikátor, který by sám rozděloval text do požadovaných kategorií, je nejprve nutné získat data, která již obsahují informace o jejich sentimentu (Ptaszynski et al., 2012). To je poměrně významné omezení, protože takováto data lze získat velice špatně. My však takovýto dataset nutně potřebujeme k tomu, abychom mohli „natrénovat" algoritmus strojového učení (viz kap. 2.2). Tato zvláštní skupina algoritmů si vytváří klasifikační model, podle poskytnutých trénovacích dat. Pokud pak takto natrénovanému algoritmu dáme na vstup neoznačkovaný dokument, je schopen s určitou dávkou pravděpodobnosti určit, do jaké kategorie sentimentu tento dokument spadá. Čím lépe jsou označkována trénovací data a čím lépe jsou vybrány příznaky daného textu, tím lepší budou výsledky klasifikace.

Jak lze nahlédnout, analýza sentimentu není triviální problém. Všechny implementace tak pracují s určitou chybou, kterou dané algoritmy produkují.

Tato práce se zaměřuje na hlavní problémy tohoto přístupu určování sentimentu textu. Ukazuje, jak získat použitelná trénovací data z běžně dostupných internetových stránek. Na těchto datech demonstruje nejběžnější postupy pro získání relevantních příznaků, které jsou pak použity k trénování několika vybraných algoritmů strojového učení.

Hlavní náplní této práce je nicméně představení a otestování účinnosti algoritmů selekce příznaků. Tyto algoritmy slouží jako doplněk k algoritmům strojového učení a s jejich pomocí lze vybrat takové příznaky, které jsou pro klasifikátor důležité a přispívají tak co nejvíce ke správnému výsledku.

Tím můžeme snadno redukovat velikost vektoru příznaků a urychlit tak dobu výpočtu při použití velkého množství dat (Singh et al., 2010).

1.1 Motivace

Analýza sentimentu je používána především pro datamining názorů široké veřejnosti. Výhodou tohoto přístupu je, že tak může být učiněno i bez vědomí lidí, kteří se k dané věci vyjadřují. V době sociálních sítí není problém získat velké množství příspěvků obsahující požadovaná klíčová slova (Saif et al., 2012; Abdul-Mageed et al., 2012). Analýza sentimentu takovýchto dat tak představuje překvapivě levnou alternativu marketingových průzkumů a uživatelských anket. Tato data mají velkou finanční hodnotu především pro marketingová oddělení firem a umožňují například přípravu lépe zaměřených reklamních kampaní. Jako konkrétní příklad použití analýzy sentimentu je často uváděno předvídání volebních výsledků. Tento postup již byl v minulosti úspěšně použit (Wang et al., 2012). Takto získaná data mohou zpřesnit výsledky předvolebních průzkumů.

1.2 Související práce

Práce nejblíže naší problematice používají algoritmy strojového učení k určení sentimentu označkovaných dat získaných z různých zdrojů. Je použito více postupů pro extrahování příznaků z příspěvků a jsou uvedeny úspěšnosti jednotlivých přístupů. Práce (Pang and Lee, 2008) navíc srovnává úspěšnost klasifikace při použití rozdílných algoritmů strojového učení. V práci (Turney and Littman, 2002) je krom toho použito jednoduché selekce příznaků na základě Mutual information (viz 2.3.1). Práce (Hatzivassiloglou and McKeown, 1997) je rovněž blízká našemu záměru, jelikož se zabývá analýzou textu za účelem zjištění, jestli byl použit subjektivní jazyk . Určení obsahu sentimentu je jedním z našich cílů. Nicméně tato práce řeší pouze polovinu námi definovaného problému. Pomocí postupů v ní uvedených můžeme určit,

jestli článek sentiment obsahuje, ale již nedokážeme určit jaký.

Většina předchozích výzkumů týkajících se analýzy sentimentu byla založena na znalostních systémech nebo modelech inspirovaných kognitivní lingvistikou (Hearst, 1992; Sack, 1994). Rovněž byly použity postupy manuální, nebo poloautomatické tvorby slovníků (Huettner and Subasic, 2000; Tong, 2001). Tyto přístupy trpí stejným problémem - trénovací dataset je vytvářen člověkem. Relevantní výzkumy ukazují, že člověk nemusí mít vždy nejlepší intuici k určování klíčových slov použitých v trénovacích datech (Pang and Lee, 2008).

Mnoho námi použitých postupů se rovněž vyskytuje v pracích zaobírajících se klasifikací dokumentů. Dokumenty mohou být klasifikovány podle zdroje, jako například v (Biber, 1988), kde je použit přístup tzv. statisticky rozpoznávané stylistické variace. Práce (Kessler et al., 1997) se zase snaží určit původ poskytnutého dokumentu - vydavatelství, autora, popřípadě novin ve kterých byl článek otištěn. Další práce se zabývají určením žánru textu. Algoritmy strojového učení zde rovněž byly použity k určení typu článku, který byl vyňat z novin. Určované kategorie článků byly například sloupec, úvodník nebo sportovní zpravodajství (Karlgren and Cutting, 1994).

Všechny tyto práce do jisté míry používají algoritmy strojového učení. O těchto algoritmech nejlépe pojednává kniha (Manning et al., 2008), která shrnuje všechny důležité aspekty týkající se této problematiky. Podrobnější informace o algoritmu SVM lze získat z (Meyer and Wien, 2001) nebo (Joachims, 1997). Algoritmus Maximum Entropy dobře shrnuje článek (Manning, 2013).

Samostatnou kapitolou jsou články zabývající se selekcí příznaků. (Mejova and Srinivasan, 2011) zkoumá účinnost algoritmů jako frequency-based selection, mutual information a POS selection na příznacích založených na stematech a bigramech. (Singh et al., 2010) uvádí vlastní způsob selekce příznaků nazvaný WCP (within class popularity) a porovnává jeho účinnost s běžně používanými postupy. Obsáhlejší studií na toto téma je (Forman et al., 2003), porovnávající větší množství algoritmů pro selekci příznaků a

parametry vhodné pro jejich výběr.

1.3 Čeština v analýze sentimentu

Ačkoliv některé studie byly prováděny s ohledem na jejich multijazyčnost (Abbasi et al., 2008; Tromp, 2012), práce zabývající se syntaktickou analýzou v českém jazyce doposud nebyla publikována. Pokud tedy nebereme v potaz studie využívající strojového překladu, nebo technik nezávislých na jazyce, je tato práce pilotním průzkumem. Čeština do projektu přináší následující problémy:

Chybějící označkovaná data Pro češtinu chybí označkované datasety jako např. Reuters-21578 pro angličtinu (Debole and Sebastiani, 2005), které jsou běžně používány pro trénink klasifikátorů v jiných jazycích. Je proto třeba použít k tréninku ne zcela ideálně označkovaná data ze sítí poskytující uživatelské recenze, nebo manuálně označkovat vlastní dataset.

Flektivita češtiny Oproti anglickým 26 rozlišuje čeština 42 písmen. V důsledku ohebnosti českých slov (skloňování, časování...) jich rovněž rozlišuje větší počet. Díky těmto dvěma rozdílům je klasifikace v češtině výpočetně náročnější. Pro češtinu navíc do nedávné doby neexistovaly nástroje schopné provádět POS tagging a lematizaci (viz kapitola 2.1.3 a 2.1.4).

Krom těchto několika bodů se postup klasifikace textu v češtině nijak zásadně neliší od klasifikace v jiných jazycích a lze tak použít ověřené postupy z výše uvedených prací.

Kapitola 2

Teoretická část

2.1 Příznaky pro analýzu sentimentu využitím strojového učení

Definice 1 *Příznaky jsou základní části důkazů, které spojují dokument d s třídou c, kterou se snažíme určit. (Vapnik, 1982)*

Použití příznaků (anglicky features) je základní technika používaná pro klasifikaci dokumentů do tříd kategorií. V definici je sice použit termín dokument, ale klasifikovat můžeme velké množství rozdílných věcí. Jako příznak pak můžeme použít libovolnou informaci o tomto objektu, která nám pomůže zařadit klasifikovaný objekt do správné kategorie.

Základní myšlenkou automatické klasifikace je vzít označkovanou množinu dat (trénovací množina), z jejíchž prvků vybereme příznaky. Tuto množinu poskytneme algoritmu strojového učení (viz kapitola 3.2), který si podle této předlohy vytvoří interní klasifikační model. Takto natrénovaný klasifikátor je pak používán k určení kategorie neznámého prvku. Algoritmus se snaží prvek zařadit do té kategorie, která obsahuje příznaky co nejpodobnější příznakům vyextrahovaným z rozpoznávaného prvku.

Jak správně určit množinu příznaků je jeden z největších problémů algoritmů strojového učení. Lze jen velice špatně předpovědět, jaký vliv bude mít zavedení nového příznaku na úspěšnost klasifikace. Často se stává, že zavedení nového příznaku výsledek klasifikace dokonce zhorší. Některé příznaky je možné určit na první pohled, určení jiných může být poměrně komplikované(Mayfield and Penstein-Rosé, 2010).

Následující příklad ukazuje identifikaci příznaků v jednoduché klasifikační úloze.

Úkol: Uvažujme dvě množiny bodů v ohraničené části roviny rozlišené černou a bílou barvou. Body tvoří shluky podle svých barev. Pokud přidáme na náhodné místo v rovině nový bod, neměl by pokud možno porušit pravidlo shlukování bodů se stejnou barvou. Definujte vhodnou množinu příznaků bodu, která pomůže klasifikátoru určit barvu nového bodu.

Řešení: První myšlenka, která napadne většinu lidí, je použít souřadnice bodů v prostoru. Tento přístup je poměrně efektivní a lze ho i dále rozvádět. Můžeme tak použít například kvadrant, ve kterém se bod nachází, vzdálenost od středu definované plochy nebo extrémní souřadnice (pro body blízko hranic definované plochy).

Zásadní chybou tohoto přístupu je ovšem přílišné založení na vzdálenosti od ostatních bodů a ignorování počtu bodů v dané oblasti. Pokud bychom měli několik málo bodů bílé barvy uvnitř velkého shluku černých bodů a nový bod by se vyskytl blízko bílé množiny, byl by tento bod pravděpodobně označen jako bílý. Řešením tohoto problému je zavést příznak udávající „hustotu“ barvy v dané oblasti. Konstrukce tohoto příznaku však již není triviální (Guyon and Elisseeff, 2006).

Na tomto příkladu je demonstrováno, že i u jednoduché klasifikační úlohy může být obtížné správně určit množinu příznaků, abychom dosáhli pokud možno optimálních výsledků klasifikace. U složitějšího klasifikačního problému, jakým je právě například analýza sentimentu, je správné určení příznaků ještě důležitější. Jako výhodné se proto jeví použít několik prověřených postupů, které mají za úkol tento problém ulehčit. Tyto postupy jsou popsány

v následujících odstavcích.

2.1.1 Základní příznaky příspěvků

Pro následující úvahy budeme předpokládat, že náš dataset je tvořen uživatelskými recenzemi produktů extrahovanými z internetového serveru (blíže viz kapitola 3.1). Jednotlivé recenze obsahují procentuální hodnocení produktu. Taková množina přímo vybízí k tomu, abychom jednotlivá slova recenzí použili pro trénink klasifikátoru. Musíme mít ovšem na paměti několik omezení:

- Ačkoliv má celý příspěvek jasně daný sentiment, neznamená to, že všechna slova v tomto příspěvku mají sentiment shodný.
- Příspěvek pravděpodobně obsahuje řadu nevýznamných slov (viz kapitola 2.1.2).
- Řada příspěvků může být ohodnocena špatně (uživatel není neomylný).
- V příspěvku se může vyskytovat ironie a sarkasmus. Hodnocení tak může vyznít zcela opačně, než recenzent zamýšlel.
- Příspěvek může obsahovat pravopisné chyby.

Pokud tedy pro klasifikaci použijeme pouze všechna nikterak nefiltrovaná slova příspěvku, zavedeme do klasifikátoru velké množství šumu, které negativně ovlivní výsledky klasifikace. Dobrým nápadem je tento šum alespoň trochu odfiltrovat zavedením minimální hranice počtu výskytů slova ve všech příspěvcích. Tato hranice (tzv. threshold) nám výrazně omezí počet příznaků a odstíní ty méně významné.

I tento přístup je nicméně problematický. Pokud odfiltrujeme málo používané slovo (například odborný termín), který se používá pouze v určité souvislosti, ztratíme tím velice dobrý příznak. Jak ukazuje experiment 3.2, není v tomto případě filtrování pouze na základě počtu výskytů ideálním řešením. Lepším přístupem je použití algoritmu selekce příznaků (viz sekce 2.3).

2.1.2 Stopwords

Stopwords je označení pro slova, která nemají z hlediska sentimentu žádný význam.Využívají se především v oblasti zaobírající se Information Retrieval (Manning et al., 2008). Pokud se podíváme na jejich slovní druhy, zjistíme, že se jedná především o spojky (a, k, aby), zájmena (my, vy, ji) nebo způsobová slovesa. Pokud bychom provedli frekvenční analýzu rozsáhlejšího textu, zjistili bychom, že jsou to právě tato slova, která se v textu vyskytují nejčastěji. Jejich odfiltrováním dosáhneme výrazného snížení šumu v příznacích příspěvků a tím i lepších výsledků klasifikace. K tomu ovšem potřebujme získat seznam stopwords.

Pro český jazyk je jako nejkomplexnější sbírka stopwords uváděn seznam použitý v projektu Apache Lucene [1]. Tento seznam ovšem neobsahuje znaménka jako "...", ".", ",", ":", ";", "(", ")", která do jisté míry mohou být rovněž považována za stopwords, protože neobsahují žádný použitelný sentiment. Jejich výskyt v běžném textu je rovněž poměrně vysoký.

Je ovšem třeba rozlišovat tato znaménka a emotikony, které naopak mohou nést velice výrazný sentiment. Tento problém je dobře vidět například u znaku "-", který může představovat pomlčku nebo znaménko mínus. V prvním případě nese nulový a v druhém naopak velice výrazný (negativní) sentiment.

Pro účely tohoto projektu byl tedy seznam stopwords doplněn o nejčastěji se vyskytující samostatná znaménka, která neobsahují sentiment v žádném z případů, ve kterém jsou použita. Kompletní seznam naleznete v příloze A.

2.1.3 Lemmatizace

Lemmatizace je technika používaná ke snižování velikosti slovníku, která využívá redukci počtu slovních tvarů. Například u sloves je za slovníkový tvar považován infinitiv a u podstatných jmen první pád jednotného čísla. Lemmatizace převede všechna slova do těchto tvarů. Pro lepší představu tabulka

[1] lucene.apache.org

slova	v	hrnci	jsem	už	vaří	a	nedám	na	něj	dopustit
lemata	v	hrnec	být	už	vařit	a	dát	na	on	dopustit

Tabulka 2.1: Slova běžné věty a jim odpovídající lemata.

slovo	**tag**
v	`RR-6-----`
hrnci	`NNIS6---A--`
jsem	`VB-S--1P-AA--`
už	`Db-------`
vařil	`VpYS--XR-AA--`
a	`J------`
nedám	`VB-S--1P-NA--`
na	`RR-4-----`
něj	`P5ZS4-3----`
dopustit	`Vf----A--`

Tabulka 2.2: POS tagy jednoduché věty.

2.1 uvádí slova jednoduché věty a jim odpovídající lemmata.

Je třeba mít na paměti, že lemmatizací přijdeme o informaci vypovídající o tvaru slova, což může negativně ovlivnit výsledky klasifikace.

2.1.4 POS tagging

POS je zkratka z anglického part of speech, což se do češtiny překládá jako slovní druh. Technika POS tagging tedy není nic jiného než strojové určování slovního druhu zadaného slova a získání informací o jeho tvaru (jako například pád, rod, druh...). Výstupem algoritmu provádějícího POS tagging bývají krátké řetězce, které kódují jednotlivé slovní druhy a jejich parametry. Pro větu uvedenou v tabulce 2.1 jsou odpovídající POS tagy uvedeny v tabulce 2.2.

Uvedené zkratky parametrů vycházejí z anglických názvů pro slovní druhy a jsou seřazeny sestupně zleva doprava, podle stupně významnosti parametru. Cílem POS taggingu je získat další informaci o tvaru slov, která byla ztracena lemmatizací. Bližší informace lze najít v dokumentaci námi použitého POStaggeru[2].

Některé slovní druhy jsou obecně vyšším nositelem sentimentu než ostatní. Jedná se především o přídavné jména (dobrý, slušné, vybavená), slovesa (milovat, doporučit,vyhodit) a příslovce (strašně, špatně, ohromně). Pokud se algoritmus strojového učení zaměří na tyto příznaky, lze očekávat lepší výsledky klasifikace.

2.1.5 N-Gramy

Pod pojmem n-gram rozumíme uspořádanou n-tici slov příspěvku. Hlavní myšlenkou použití n-gramů jako příznaků je fakt, že dvojice (trojice, čtveřice...) slov může být nositelem sentimentu, který se nevyskytuje v jeho jednotlivých částech. Je ovšem třeba rozlišovat n-gramy a n-tice slov. Prvky n-gramu na sebe vždy navazují (jsou tvořeny n mínus k-tým až n-tým slovem). N-tice mohou být složeny z libovolných slov věty (ale i celého příspěvku) v libovolném pořadí. Jako příznaky lze použít oba dva přístupy.

Ačkoliv první přístup dává zdánlivě větší smysl, výhodou druhého přístupu je vytvoření nesrovnatelně většího vektoru příznaků. Obzvláště pro větší hodnoty n vektor příznaků roste velice rychle. Často se proto tato technika využívá ke konstrukci obrovského vektoru příznaků, ze kterého se pak extrahují ty nejdůležitější (viz algoritmy selekce příznaků - sekce 2.3).

Jelikož můžeme určit i slovní druhy (viz 2.1.4), můžeme konstruovat n-gramy přesně zadaných struktur. Oblíbený je například bigram přídavné jméno - podstatné jméno (např. dobrá cena) nebo obecnější podstatné jméno - sloveso - přídavné jméno (např. produkt být špatný).

[2]http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/cz/html/index.html

Problémem takovéto konstrukce příznaků je přílišná specifičnost. Nicméně pokud je tento příznak nalezen, jedná se o velice silný indikátor náležení k dané třídě. Lze však očekávat, že se bude vyskytovat jenom ve velice omezené množině dat (Mejova and Srinivasan, 2011).

2.1.6 Redukce počtu příznaků

Ačkoliv algoritmy strojového učení obvykle dávají lepší výsledky úměrně s větším množstvím poskytnutých příznaků, problémem při enormním množství příznaků začíná být výpočetní doba těchto algoritmů. Vyvstává proto otázka, jak zmenšit vektor příznaků tak, aby se pokud možno co nejméně zhoršila úspěšnost klasifikace.

Redukce počtu příznaků dosáhneme jejich filtrováním podle námi specifikovaných kritérií. Obvyklý požadavek je vyfiltrovat ty příznaky, které mají nejnižší váhu. Tato váha je udávána veličinami založenými na různých statistikách. Jedno z možných kritérií, podle kterého lze filtrovat již bylo uvedeno výše - počet výskytů slov. Jiný způsob filtrování představují například algoritmy selekce příznaků (viz kap 2.3).

2.2 Algoritmy strojového učení

Algoritmy strojového učení (dále také jako ML – machine learning) spadají pod problematiku umělé inteligence. Učením v tomto případě rozumíme vytváření vnitřního klasifikačního modelu, který umožní určit, do které kategorie spadá rozpoznávaný prvek. Použití algoritmů strojového učení pro analýzu sentimentu je tak sofistikované řešení problému klasifikace dokumentů. Konkrétně se zde budeme bavit o tzv. učení s učitelem (supervised learning). To znamená, že pro vstupní data je předem určen správný výsledek.

Postup použití algoritmu strojového učení má dvě části:

1. Naučení modelu z množiny označkovaných trénovacích dat

2. Klasifikace neoznačkovaných dat pomocí natrénovaného modelu

Tento proces je formálně popsán na následujících řádkách. Nejprve definujeme množinu trénovacích dat:

$$\{(D_i \in \mathbf{D}, S_i \in \mathbf{S})\} \tag{2.1}$$

Prvním prvkem trénovací množiny je jeden z dokumentů D_i z množiny všech dokumentů $\mathbf{D}$. Dimenzemi těchto dokumentů jsou jejich příznaky. Druhým prvkem je S_i jeden ze sentimentů z množiny $\mathbf{S}$. Dále definujeme úkol ML jako:

$$\text{nalézt g} : \mathbf{D} \rightarrow \mathbf{S}, g(D_i) = \arg\max f(D_i, S_i) \tag{2.2}$$

(Tsytsarau and Palpanas, 2012)

Slovně lze tento předpis vyjádřit tak, že chceme najít funkci g, která mapuje dokumenty na jednotlivé sentimenty. Na vstup je algoritmu poskytnuta množina trénovacích párů {dokument(s vektorem příznaků),sentiment}. Algoritmus ke své práci používá blíže nespecifikovanou hodnotící funkci f. Bez ztráty na obecnosti lze celý učící proces algoritmu považovat za postupné zpřesňování hodnotící funkce. Tato definice je společná pro všechny algoritmy ML, nicméně jejich implementace, použití a efektivita je vzájemně velice odlišná. Pro naše potřeby budeme používat trojici klasifikátorů – Naive-Bayes (2.2.1), Maximum Entropy (2.2.2) a Support vector machines (2.2.3).

2.2.1 Naive Bayes

Hlavními myšlenkami algoritmu Naive-Bayes je použití pravděpodobnostního modelu přístupu k textu a bayesovského teorému, který uplatňuje tzv. naivní přístup. Ten předpokládá, že výskyt všech slov v dokumentu je statisticky nezávislý na výskytu všech ostatních slov. To je poměrně významné

zjednodušení skutečnosti, které sice pozitivně ovlivňuje výpočetní složitost algoritmu, nicméně poskytuje relativně horší výsledky.

Uvažujeme-li binární verzi klasifikátoru (rozdělení do 2 tříd +/−), je cílem tohoto algoritmu určit pravděpodobnost $p(+|d)$, tedy pravděpodobnost toho, že dokument d spadá do třídy +. V ideálním případě by pak dokument měl být zařazen do kategorie +, pokud je hodnota $p(+|d) \geq 0,5$.

$$p(+|d) = \frac{p(+) \cdot \prod_i p(w_i|+)}{p(+) \cdot \prod_i p(w_i|+) + p(-) \cdot \prod_i p(w_i|-)} \tag{2.3}$$

$p(+)$ a $p(-)$ vyjadřují pravděpodobnost výskytu dané třídy. Tuto pravděpodobnost snadno odvodíme z trénovacích dat. $p(w_i|c)$ pak označuje pravděpodobnost výskytu příznaku w ve třídě c. Pokud bychom chtěli tento model generalizovat pro n tříd modifikovali bychom pouze podobu čitatele.

$$p(c|d) = \frac{p(c) \cdot \prod_i p(w_i|c)}{\sum_{j=0}^{n} p(c_i) \cdot \prod_i p(w_i|c_j)} \tag{2.4}$$

Hodnota jmenovatele zůstává konstantní pro všechny třídy.

Rovnici lze dále zjednodušit vyjádřením v exponenciální podobě:

$$p(c|d) = \frac{\exp[\log p(c) + \sum_i p(w_i|c)]}{\sum_{j=0}^{n} \exp[\log p(c_i) + \sum_i \log p(w_i|c_j)]} \tag{2.5}$$

(Manning et al., 2008)

2.2.2 Maximum entropy

Algoritmus Maximum entropy (dále také jako MaxEnt) používá komplexnější model, který již není zjednodušen předpokladem statistické nezávislosti slov v dokumentu. Jinými slovy poskytuje méně zkreslený odhad na základě uvedených informací (Manning, 2013). Základní předpis pro výpočet je následující:

$$p(c|d) = \frac{1}{Z(d)} \exp(\sum_i \lambda_i w_i(d, c)) \tag{2.6}$$

$\lambda_{i,c}$ udává váhu příznaků. Velké $\lambda_{i,c}$ znamená, že w_i je silným příznakem třídy c. Pro určení vah příznaků se používají iterační metody jako například:

- Improved Iterative Scaling (IIS)
- Generalized Iterative Scaling (GIS)
- Limited-Memory Variable Metric (L-BFGS)

Pro naše potřeby budeme dále používat metodu L-BFGS, jelikož studie na toto téma prokázaly, že je pro tyto účel nejvhodnější (Pang and Lee, 2008).

2.2.3 Support vector machines

Klasifikátor Support vector machines (dále jako SVM) je založen na principu strukturální minimalizace rizik, který se snaží najít hypotézu h, pro kterou může garantovat nejnižší skutečnou chybu (true error). Skutečná chyba hypotézy h je pravděpodobnost, že h bude chybná v náhodně generovaném případě vybraném z testovací množiny. Následující nerovnice pak udává vztah mezi chybou hypotézy h a chybou na trénovací množině.

$$p(error(h)) \leq trainerror(h) + 2\sqrt{\frac{d(\ln \frac{2n}{d} + 1) - \ln \frac{\eta}{4}}{n}} \tag{2.7}$$

Tato nerovnice platí s pravděpodobností alespoň $1 - \eta$. Neznámá n označuje počet trénovacích příkladů a d je VC-dimenze (VCdim), což je vlastnost prostoru hypotézy a indikuje jeho působivost (Shao et al., 2000). Tato nerovnice odráží kompromis mezi složitostí hypotézy a trénovací chybou. Jednoduchý prostor hypotézy (malé VCdim) pravděpodobně nebude obsahovat dobré

aproximační funkce a povede tedy k vysoké trénovací a tím pádem i skutečné chybě. Na druhé straně příliš velký prostor hypotézy (velké VCdim) vede sice k malé trénovací chybě, nicméně druhý term na pravé straně rovnice bude velký. Tato situace se obvykle nazývá „overfitting“ (Dietterich, 1995). Pokud nastane tato situace, může model příliš prudce reagovat i na malé fluktuace ve vstupních datech. Z těchto poznatků můžeme vyvodit závěr, že je velmi důležité vybrat prostor hypotézy se správnou složitostí.

V principu strukturální minimalizace rizik je tohoto výběru dosaženo tím, že se definuje struktura prostoru hypotézy h_i, přičemž se zvyšuje jejich VC-dimenze d_i.

$$h_1 \subset h_2 \subset h_3 \subset ... \subset h_i \subset ... \forall_i : d_i \leq d_{i+1} \tag{2.8}$$

Cílem je najít index i, pro který je nerovnice (2.7) minimální. Strukturu inkrementace VCdim lze vybudovat pomocí lineárních prahových funkcí typu:

$$h(\vec{d}) = sign\{\vec{w} \cdot \vec{d} + b\} = \begin{cases} +1 & \text{pokud } \vec{w} \cdot \vec{d} + b > 0, \\ -1 & \text{pokud } \vec{w} \cdot \vec{d} + b \leq 0 \end{cases} \tag{2.9}$$

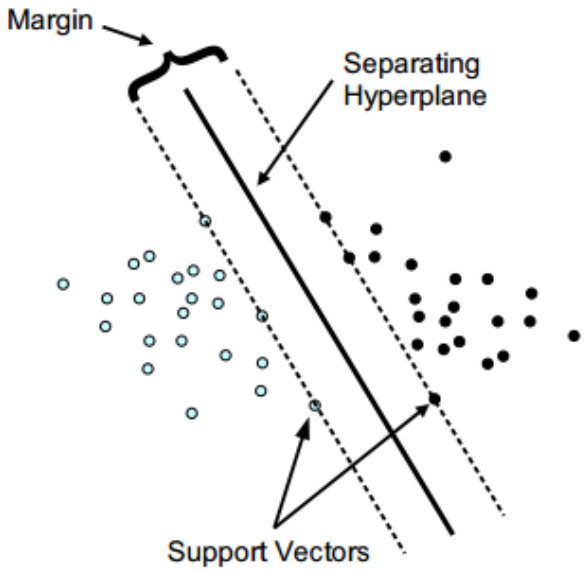
Místo budování struktury založené na počtu příznaků a strategie selekce příznaků (jako u MaxEnt), SVM používá struktury, které uznávají skutečnost, že většina příznaků v kategorizaci textu je relevantní.

Definice 2 *Vezměme v úvahu nadroviny* $h(\vec{d}) = sign\{\vec{w} \cdot \vec{d} + b\}$ *jako hypotézy. Pokud všechny trénovací vektory* $\vec{d_i}$ *jsou obsaženy ve sféře o průměru* R *a pro všechny* $\vec{d_i}$ *platí*

$$|\vec{w} \cdot \vec{d} + b| \geq 1 \text{ } \textit{přičemž} \text{ } ||\vec{w}|| = A \tag{2.10}$$

Pak mají tyto nadroviny VCdim ohraničené

$$d \leq \min([R^2 A^2], n) + 1 \tag{2.11}$$



Obrázek 2.1: Dvojice support vectors je tvořena příklady, které jsou nejblíže dělicí nadrovině (separating hyperplane). Nadrovina pak rozděluje prostory tříd tak, aby dosáhla maximálních okrajů (margin).

(Vapnik, 1982)

Povšimněte si, že VCdim těchto nadrovin nutně nezávisí na počtu příznaků. Namísto toho VCdim závisí na Euklidovské délce $||\vec{w}||$ váhového vektoru $\vec{w}$. To znamená, že můžeme snadno generalizovat ve vysoko dimenzionálních prostorech, pokud má naše hypotéza malý váhový vektor (Joachims, 1997).

V základní podobě SVM najde nadrovinu, která odděluje trénovací data a zároveň má nejmenší váhový vektor. Tato nadrovina rozděluje dvě trénovací množiny s maximálním okrajem (viz obr. 2.1). Nalezení této nadroviny lze vyjádřit jako následující optimalizační problém.

$$\begin{aligned} &\text{minimalizujte } ||\vec{w}|| \\ &\text{tak aby } \forall_i y_i[\vec{w} \cdot \vec{d} + b] \geq 1 \end{aligned} \tag{2.12}$$

Přičemž y_i se rovná ± 1 podle toho, zda dokument spadá do třídy + nebo − (uvažujeme-li binární klasifikaci). Omezení tohoto přístupu je, že je nutné, aby všechny trénovací příklady byly označeny korektně.

Jelikož je tento optimalizační problém obtížní pro numerický výpočet, používá se se metoda Lagrange multipliers (Joshi et al., 2012) k převedení tohoto problému na ekvivalentní kvadratický optimalizační problém.

$$\begin{aligned} \text{minimalizujte } &\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{d_i} \cdot \vec{d_j} \\ &\text{tak aby } \forall_i : yi[\vec{w} \cdot \vec{d} + b] \geq 1 \end{aligned} \tag{2.13}$$

Pro tento typ optimalizační úlohy existuje algoritmus, který dokáže najít globální optimum. Výsledkem tohoto algoritmu je soubor koeficientů. Tyto koeficienty mohou být použity ke konstrukci nadroviny splňující podmínky minimalizace uvedené v (2.12).

$$\vec{w} \cdot \vec{d} = (\sum_{i=1}^{n} \alpha_i^* y_i \vec{d_i}) \cdot \vec{d} = \sum_{i=1}^{n} \alpha_i^* y_i (\vec{d_i} \cdot \vec{d}) \\ \text{přičemž b} = \frac{1}{2}(\vec{w} \cdot \vec{d_+} + \vec{w} \cdot \vec{d_-}) \tag{2.14}$$

Tato rovnice ukazuje, jak je konstruován výsledný váhový vektor jako lineární kombinace trénovacích příkladů. K výsledku přispívají pouze ty příklady, jejichž koeficient α_i je větší než nula. Tyto vektory jsou nazývány $SupportVectors$. Na obrázku 2.1 lze vidět dvojici support vectors. Jsou tvořeny trénovacími příklady, které mají minimální vzdálenost od oddělující nadroviny.

SVM ve své základní podobě může být použito pouze pro řešení binárních klasifikačních problémů (rozdělení do dvou tříd). Aby bylo možné klasifikovat do více tříd, používá se tzv. $one-against-one$ přístup. Tento přístup používá více instancí binárního klasifikátoru, který spouští pro všechny možné kombinace tříd. Rekonstrukce výsledku se provede pomocí volícího mechanismu (Arun Kumar and Gopal, 2010).

2.3 Selekce příznaků

Pod pojmem selekce příznaků rozumíme výběr takových příznaků, které mají pokud možno co největší váhu pro algoritmus strojového učení - to znamená, že přispívají největší měrou k úspěšné klasifikaci. Základní myšlenkou všech těchto algoritmů udává definice 3.

Definice 3 *Příznak má tím větší váhu, čím méně rovnoměrně je zastoupen v jednotlivých kategoriích. (Forman et al., 2003)*

Pokud je tedy příznak zastoupen například z 90% v jedné třídě, znamená to, že je velice silným indikátorem této třídy a algoritmus selekce příznaků by mu měl přiřadit velkou váhu. Kromě této základní myšlenky využívají některé

algoritmy ještě procentuální zastoupení třídy, do které příznak spadá. Tento podíl je důležitý především pro nevyvážené datasety (viz kapitola 2.4.2).

Potřebujeme-li zredukovat velikost vektoru příznaků například protože jich je příliš mnoho, chceme, abychom odstranili ty, které mají pokud možno nejnižší vliv na úspěšnost klasifikace. Algoritmy selekce nám takové příznaky pomohou identifikovat. V důsledku odfiltrování těchto příznaků se pak může stát, že se úspěšnost klasifikace zvýší, protože jsou odfiltrovány především špatné příznaky. Vše ovšem závisí na vhodnosti použitého algoritmu, přísnosti nastaveného filtrování a použitých datech.

Algoritmy selekce příznaků pracují tím způsobem, že pro každý příznak spočtou jeho váhu. V námi vytvořené implementaci je tato váha normalizována na interval <0,1>, kde 0 znamená minimální váhu příznaku a 1 maximální. Vektor příznaků tak můžeme snadno filtrovat podle toho, jak velkou redukci vektoru příznaků právě potřebujeme.

Algoritmy selekce příznaku pracují s pravděpodobnostmi výskytu příznaků v jednotlivých kategoriích. Následující rovnice ukazují metodu výpočtu jednotlivých pravděpodobností pro 2 kategorie (c_1 a c_2). t_k a $\bar{t_k}$ označuje přítomnost, respektive nepřítomnost příznaku.

	$\mathbf{c_1}$	$\mathbf{c_2}$
$\mathbf{t_k}$	a	b
$\mathbf{\bar{t_k}}$	c	d

Nejprve si vyjádříme N jako součet počtu příznaků ve všech kategoriích.

$$N = a + b + c + d \tag{2.15}$$

Pravděpodobnost výskytu příznaku v kategorii c_1 pak spočteme jako podíl počtu výskytů termů v dané kategorii s celkovým počtem příznaků.

$$p(t_k, c_1) = \frac{a}{N} \tag{2.16}$$

Některé algoritmy selekce příznaků k výpočtu rovněž potřebují vyjádření pravděpodobnosti výskytu příznaku ve všech kategoriích (test existence). Tuto pravděpodobnost opět vyjádříme podílem.

$$p(t_k) = \frac{a+b}{N} \tag{2.17}$$

Obdobně lze vyjádřit pravděpodobnost, že příznak bude dané kategorie (test příslušnosti).

$$p(c_1) = \frac{a+c}{N} \tag{2.18}$$

Rovněž je občas zapotřebí vypočítat podmíněnou pravděpodobnost výskytu příznaku v kategorii. Tento vztah udává pravděpodobnost toho, že příznak existuje, za předpokladu, že se nachází v dané kategorii.

$$p(t_k|c_1) = \frac{a}{a+c} \tag{2.19}$$

Pokud uvažujeme 3 různé kategorie, do kterých může příznak spadat (pozitivní/neutrální/negativní), dostaneme celkem 6 různých pravděpodobností, které nám udávají pravděpodobnost výskytu/absence příznaku. Tím známe všechny vztahy potřebné k výpočtu následujících algoritmů.

Rovnice jsou uváděny v součtových tvarech pro diskrétní náhodné veličiny, přičemž $m = \{t_k, \bar{t_k}\}$ a n je rovno počtu kategorií. Pokud není uvedeno jinak, je jako základ logaritmů použito 2.

2.3.1 Mutual information

Mutual Information (dále jako MI) je veličina dříve označována také jako transinformace. Měří vzájemnou závislost dvou náhodných proměnných. Tento základní předpis, od kterého se odvozují některé další, má následující tvar:

$$MI = \sum_{i=0}^{n} \sum_{k=0}^{m} \log \frac{P(t_k, c_i)}{P(c_i)P(t_k)} \tag{2.20}$$

Tento vzorec vyjadřuje množství informace, kterou dvě náhodné veličiny sdílejí. Jinými slovy - měří nakolik známost jedné náhodné proměnné ovlivní redukci neznámosti druhé proměnné. Pokud jsou tedy tyto veličiny vzájemně nezávislé, znalost jedné veličiny nijak neovlivní znalost druhé a $MI = 0$. Naopak – pokud jsou tyto veličiny identické, je MI rovno neurčitosti obsažené v první z náhodných veličin (tzv. entropie náhodné veličiny). Lze tedy tvrdit, že pokud jsou tyto dvě veličiny identické, pak mají stejnou entropii.

Pokud jsou veličiny X a Y nezávislé, pak platí:

$$p(x, y) = p(x)p(y) \tag{2.21}$$

tudíž

$$\log \left(\frac{p(x, y)}{p(x)p(y)} \right) = \log 1 = 0 \tag{2.22}$$

Dále platí, že MI je vždy nezáporná a symetrická ($MI(X, Y) = MI(Y, X)$). (Zheng et al., 2004)

2.3.2 Information gain

Infromation gain (dále IG) je známá také pod názvy Kullback-Leiblerova divergence, relativní entropie nebo informační divergence. Udává nesymetrickou míru rozdílu dvou pravděpodobnostních rozdělení. Přesněji IG náhodné veličiny X z Y je velikost ztráty informace, pokud je X použito k aproximaci Y. Narozdíl od MI není IG symetrická.

$$IG = \sum_{i=0}^{n} \sum_{k=0}^{m} P(t_k, c_i) \log \frac{P(t_k, c_i)}{P(c_i)P(t_k)} + P(\bar{t_k}, c_i) \log \frac{P(\bar{t_k}, c_i)}{P(c_i)P(\bar{t_k})} \tag{2.23}$$

(Uchyigit, 2012)

Obdobně jako MI je i IG vždy nezáporná. Navíc lze použít hodnotu MI k snadnějšímu výpočtu IG, což se hodí, pokud používáme oba algoritmy najednou. Výsledný předpis udává rovnice 2.25.

$$IG = \sum_{i=0}^{n} \sum_{k=0}^{m} P(t_k, c_i) MI(t_k, c_i) + P(\bar{t_k}, c_i) MI(\bar{t_k}, c_i) \tag{2.24}$$

2.3.3 Chi square

Tento algoritmus označovaný jako χ^2 je velice dobře známý ze světa statistiky, kde se často používá k určování, zda množina dat vyhovuje dané distribuční funkci (chi-square test). Pro účely selekce příznaků je jeho použití obdobné. K výpočtu χ^2 se využívá dvou dalších koeficientů, které lze rovněž použít k selekci příznaků.

Nejprve vyjádříme koeficient GSS

$$GSS(t_k, c_i) = P(t_k, c_i) P(\bar{t_k}, \bar{c_i}) - P(t_k, \bar{c_i}) P(t_k, \bar{c_i}) \tag{2.25}$$

Dále vyjádříme koeficient NGL

$$NGL(t_k, c_i) = \frac{\sqrt{N} GSS(t_k, c_i)}{\sqrt{P(t_k) P(\bar{t_k}) P(c_i) P(\bar{c_i})}} \tag{2.26}$$

Výsledná rovnice je druhou mocninou koeficientu NGL.

$$\chi^2 = \sum_{i=0}^{n} \sum_{k=0}^{m} NGL(t_k, c_i)^2 \tag{2.27}$$

(Uchyigit, 2012)

2.3.4 Odds ratio

Odds Ratio (OR) je míra velikosti účinku, která popisuje sílu asociací, nebo nezávislost mezi dvěma binárními hodnotami. Používá se jako deskriptivní

statistika a hraje důležitou roli v logistické regresi. Slovně vyjádřeno je OR poměr šance výskytu události v jedné skupině oproti šanci výskytu události ve všech ostatních skupinách. Tento vztah lze formálně zapsat jako:

$$OR = \sum_{i=0}^{n} \sum_{k=0}^{m} \log \frac{P(t_k|c_i)P(\bar{t_k}|\bar{c_i})}{P(\bar{t_k}|c_i)P(t_k|\bar{c_i})} \tag{2.28}$$

(Zheng et al., 2004)

Za předpokladu výskytu pouze dvou skupin znamená $OR = 1$, že událost se se stejnou pravděpodobností vyskytuje v obou skupinách. Hodnota $OR > 1$ indikuje vyšší výskyt v první skupině, a $OR < 1$ vyšší výskyt ve druhé. OR je vždy nezáporné a definované pouze za předpokladu, že obě podmíněné pravděpodobnosti ve jmenovateli jsou nenulové.

2.3.5 Relevancy score

Relevancy score (RS) udává „nevyváženost“ výskytu příznaku napříč všemi uvažovanými dokumenty dané třídy. Jeho definice je velice podobná OR, jelikož rovněž využívá podmíněných pravděpodobností.

$$RS = \sum_{i=0}^{n} \sum_{k=0}^{m} \log \frac{P(t_k|c_i)}{P(\bar{t_k}|\bar{c_i})} \tag{2.29}$$

(Uchyigit, 2012)

2.4 Vyhodnocování úspěšnosti klasifikace

2.4.1 Křížová validace

Nejčastějším způsobem testování úspěšnosti klasifikačních algoritmů je metoda křížové validace (crossvalidation). Za předpokladu, že máme označkovanou celou datovou množinu, můžeme koncipovat experiment tak, že pro

trénink klasifikátoru použijeme pouze část této množiny (ať už náhodně vybranou, nebo pevně daný úsek). Zbytek dat necháme vyhodnotit klasifikátorem. Jelikož jsou tato data již označkovaná předem, můžeme určit, jestli byla klasifikace jednotlivých prvků úspěšná. Tento postup se obvykle opakuje několikrát a výsledné hodnocení se zprůměrovává.

Obvykle se používá větší část dat k tréninku a pouze malá podmnožina k testování úspěšnosti klasifikace. V experimentech uvedených v této práci je použito rozdělení 1:9, 90% datasetu je použito pro trénink a pouze 10% k testování úspěšnosti klasifikace. Celý experiment je pak prováděn 10x, přičemž hranice testovací množiny se posunuje vždy s fixním krokem. To zaručuje, že data v testovacím setu jsou pokaždé rozdílná.

Jako měřítko úspěšnosti lze použít Accuracy. Tento ukazatel udává podíl úspěšně klasifikovaných prvků - je nejblíže tomu co laicky označujeme jako procentuální úspěšnost. Takto lze ovšem vyjádřit úspěšnost klasifikace pouze tehdy, pokud všechny kategorie mají stejný počet prvků (viz dále).

2.4.2 Problém nevyváženého datasetu

Častý problém při konstruování datasetu je tzv. nevyrovnaný dataset (imbalanced dataset). Tento problém znamená, že data používaná pro trénink algoritmů strojového učení nejsou rovnoměrně rozdělena do jednotlivých kategorií. Pokud v tomto případě použijeme Accuracy k vyjádření úspěšnosti klasifikace, zjistíme, že výsledek neodpovídá skutečnosti.

Pokud by měla totiž jedna kategorie výrazně vyšší zastoupení v datech, stačilo by, aby algoritmus strojového učení zařadil všechny prvky z testovacího setu do této kategorie a výsledek by zůstal dobrý. Pokud bychom ovšem brali úspěšnost klasifikace po jednotlivých kategoriích, zjistili bychom, že úspěšnost klasifikace ve všech kategoriích kromě jedné je 0%. Potřebujeme proto jinou metriku, kterou bychom dokázali vyjádřit úspěšnost klasifikace i pro nevyvážené datasety.

actual / predicted	**negative**	**positive**
negative	a	b
positive	c	d

Tabulka 2.3: Binární confusion matrix.

2.4.3 Confusion matrix

V této práci je pro řešení problému nevyváženého datasetu použito postupu uvedeném v (Kohavi and Provost, 1998) založeném na confusion matrix. Tato matice se v základní binární podobě skládá ze čtyř prvků (viz tabulka 2.3).

a počet prvků správně identifikovaných jako negativní

b počet prvků špatně identifikovaných jako pozitivní

c počet prvků špatně identifikovaných jako negativní

d počet prvků správně identifikovaných jako pozitivní

Z této matice se počítá množství ukazatelů. Tou základní je již výše uvedená Accuracy (AC), která udává celkové množství prvků, které byly určeny správně. Jak již bylo řečeno, je tento ukazatel problematický při použití v nevyrovnaném datasetu.

$$AC = \frac{a+d}{a+b+c+d} \tag{2.30}$$

Recall, nebo také true positive rate (TP), udává podíl pozitivních prvků, které byly určeny správně.

$$TP = \frac{d}{c+d} \tag{2.31}$$

Precision (P) udává podíl pozitivních prvků, které byly skutečně pozitivní.

$$P = \frac{d}{b+d} \tag{2.32}$$

My však budeme dále používat ukazatel známý pod názvem F1 measure, který zohledňuje jak precision tak recall.

$$F_1 = 2 \cdot \frac{P \cdot TP}{P + TP} \tag{2.33}$$

Tento výraz je slovně interpretován jako vážený průměr precision a recall. Nejlepší hodnota F1 measure je 1 a nejhorší 0. F1 measure lze bez potíží používat na nevyvážených datasetech.

Pro jednoduchost zde bylo předvedeno odvození ukazatelů z confusion matrix o dvou kategoriích. Postup pro odvození matice, kterou budeme používat (3 kategorie), je obdobný. Velikost matice je obecně $n * n$, kde n je počet kategorií.

Kapitola 3

Experimentální část

Tato část je zaměřená na praktické pokusy s analýzou sentimentu. Některé experimenty využívají informací o nejlepších dosažených výsledcích z předchozích experimentů. V několika případech je omezena velikost zpracovávaných dat z výpočetních důvodů.

Navržené experimenty:

Získání vhodných dat Získat dostatečně velký soubor dat, který je navíc alespoň z části označkovaný (již obsahuje informace o sentimentu obsahu).

Předzpracování dat Provést předzpracování dat a zjistit, které algoritmy předzpracování přispívají k lepšímu výsledku.

Výběr příznaků Definovat jednotlivé typy příznaků a porovnat, nakolik přispívají k úspěšné klasifikaci.

Srovnání algoritmů strojového učení Vyzkoušet, který algoritmus strojového učení generuje nejlepší výsledky.

Srovnání algoritmů selekce příznaků Porovnat účinnost algoritmů selekce příznaků.

Obrázek 3.1: Ukázka uživatelské recenze na serveru heureka.cz.

Použití nejlepších výsledků Použít nastavení s nejlepšími výsledky ze všech ostatních experimentů s cílem získat co nejlepší výsledek klasifikace.

3.1 Získání datasetu

Pro záměry této práce se ukázalo jako nejsnazší řešení vygenerovat kompletně nový dataset z dat volně dostupných na internetu. K tomuto účelu bylo využito serveru heureka.cz [1]. Tento portál obsahuje informace o zboží v českých eshopech a umožňuje porovnání cen. Co je důležité pro tuto práci je fakt, že server obsahuje recenze jednotlivých produktů a jejich hodnocení. Navíc je v každé recenzi uveden seznam pozitiv a negativ (viz obr. 3.1).

Data byla získána ze serveru jednoduchou technikou dataminingu a zpracována do relační databáze s pomocí knihovny JSoup[2], která umožňuje parsovat HTML soubory pomocí metody DOM. Ze získaných dat nás zajímá především procentuální hodnocení produktu, které by mělo udávat sentiment celé recenze. Seznam pozitiv a negativ je třeba zpracovávat zvlášť, protože i pozitivní recenze může obsahovat seznam negativ. Tabulka 3.1 udává počet prvků, které byly zpracovány.

[1] www.heureka.cz
[2] www.jsoup.org

typ prvku	**počet**
produkt	75325
recenze	141691
pozitivní nebo negativní vyjádření v recenzi	307234

Tabulka 3.1: Počty prvků extrahovaných ze serveru heureka.cz.

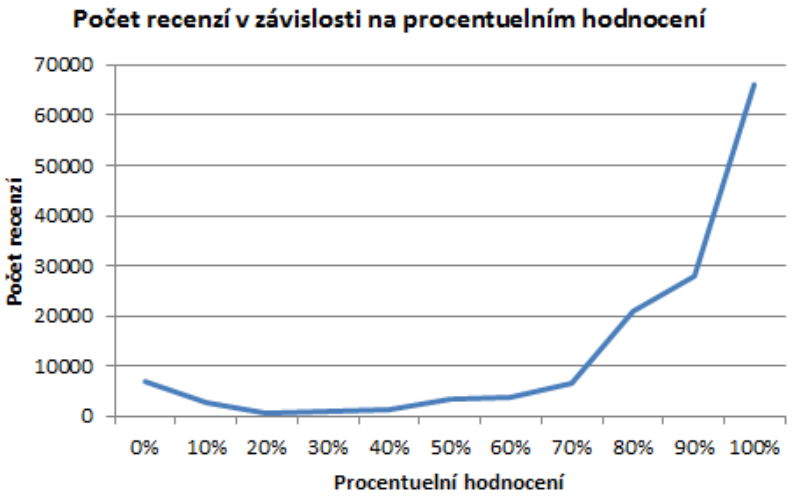
hodnocení [%]	**počet hodnocení**
0	6831
10	2626
20	760
30	1083
40	1289
50	3372
60	3887
70	6676
80	21012
90	27966
100	66188

Tabulka 3.2: Počet recenzí na serveru heureka.cz, mající dané hodnocení.

Na heuréce je možné v recenzi ohodnotit produkt 0-5 hvězdičkami, přičemž je možno dávat i půlky hvězdiček. Vzniká tak 11 kategorií ohodnocení odstupňovaných po 10% od 0% až po 100%. Tabulka 3.2 udává zastoupení počtu recenzí v jednotlivých procentuálních kategoriích.

Toto rozdělení odpovídá definici nevyváženého datasetu (viz 2.4.2), tento problém je dobře rozpoznatelný na grafu 3.2. Navíc máme 11 kategorií a potřebujeme recenze klasifikovat pouze do tří (pozitivní/neutrální/negativní). Je proto třeba určit hranici, která odděluje tyto tři kategorie.

Jelikož předem nemáme o příspěvcích žádné informace, nezbývá nám, než provést manuální průzkum několika produktů a rozdělit příspěvky podle



Obrázek 3.2: Zastoupení recenzí v procentuálních kategoriích.

kategorie	**procentuální hnodnocení**	**počet hodnocení**
pozitivní	90-100%	94155
neutrální	50-80%	13935
negativní	0-40%	12589

Tabulka 3.3: Navržené rozdělení do kategorií.

hodnocení recenzí, které tyto produkty obsahují. K tomuto účelu byl vybrán produkt IPhone 5, protože obsahuje větší množství recenzí rozdílných kategorií a navíc je znám široké veřejnosti. Na základě těchto recenzí bylo navrženo rozdělení do tří kategorií popsané v tabulce 3.3.

3.2 Předzpracování dat

Základní metody předzpracování dat již byly uvedeny vkapitole 2.1. V tomto experimentu porovnáme 6 metod předzpracování a jejich vliv na klasifikaci. Porovnávané metody jsou:

čistý text jsou použita všechna slova v textu (= žádné předzpracování)

threshold slovo se musí v textu vyskytnou minálně tolikrát, kolik je hodnota thresholdu, jinak bude vyřazeno

stopwords z čistého textu jsou odfiltrována stopwords

threshold + stopword kombinace filtrování stopwords a minimálního počtu výskytů slova

lemma místo samotných slov jsou použita lemmata. Z lemmat jsou již předem automaticky odfiltrovány stopwords.

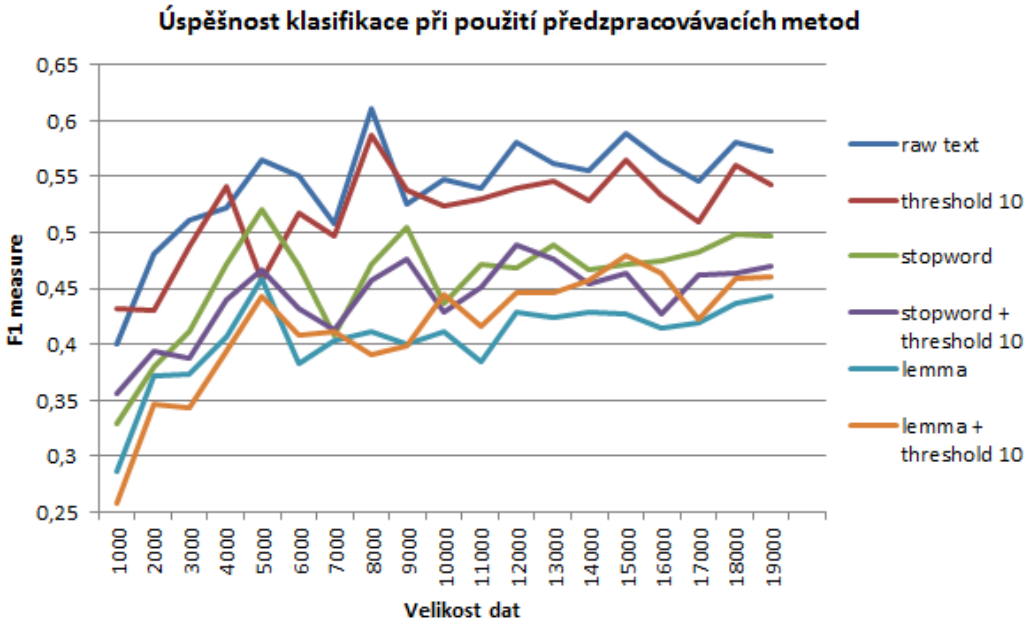
lemma + threshold lemata jsou navíc filtrována podle minimálního počtu výskytů

Z důvodu výkonnostních omezení byl tento experiment proveden na limitovaném množství dat. Bylo použito vzorku od 1000 po 19000 recenzí (90% dat použito pro trénování a 10% pro testování) a algoritmu maximum entropy. Měřené parametry byly úspěšnost klasifikace (udaná F1 measure), velikost vektoru vyextrahovaných příznaků a doba výpočtu [3]. Graf 3.3 udává úspěšnost klasifikace při použití jednotlivých metod předzpracování.

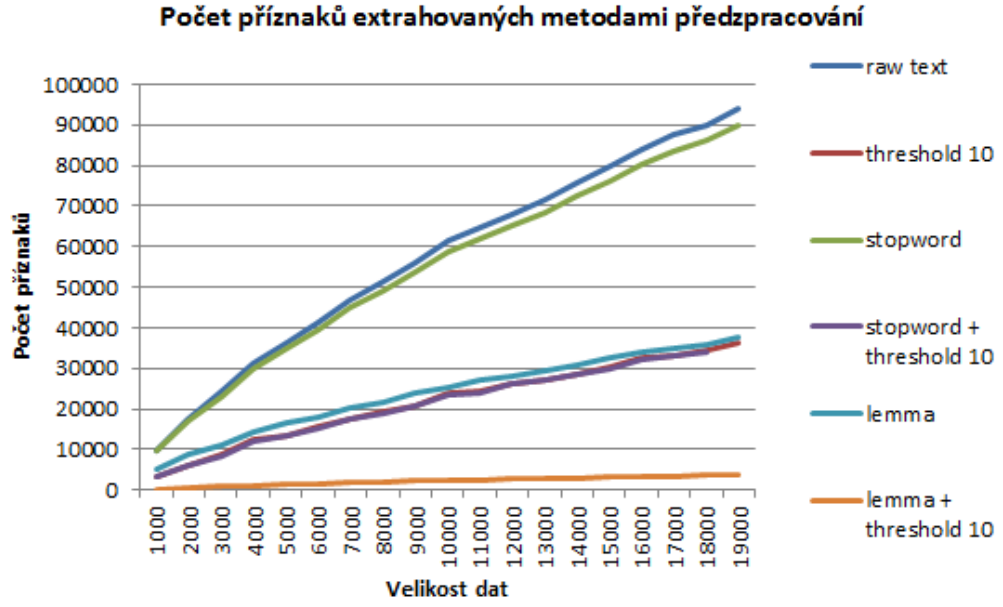
Tyto výsledky jsou nicméně poněkud zavádějící, jelikož z nich nejlépe vychází metody, které využívají minimálního nebo žádného předzpracování. Tohoto efektu je dosaženo tím, že jednodušší metody generují z recenzí daleko větší počet příznaků (viz graf 3.4). Ačkoliv mají tyto příznaky daleko menší váhu než vyfiltrované příznaky po přísnějším předzpracování, jejich daleko vyšší počet přispívá k úspěšnější klasifikaci.

Porovnáme- li například dvě podobné hodnoty - čistý text (2000) s hodnotou lemma + threshold 10 (15000) a porovnáme počet příznaků vyextrahovaných těmito metodami, zjistíme, že zatímco z čistého textu bylo v tomto případě vygenerováno 17677 příznaků, k dosažení obdobné úspěšnosti klasifikace bylo

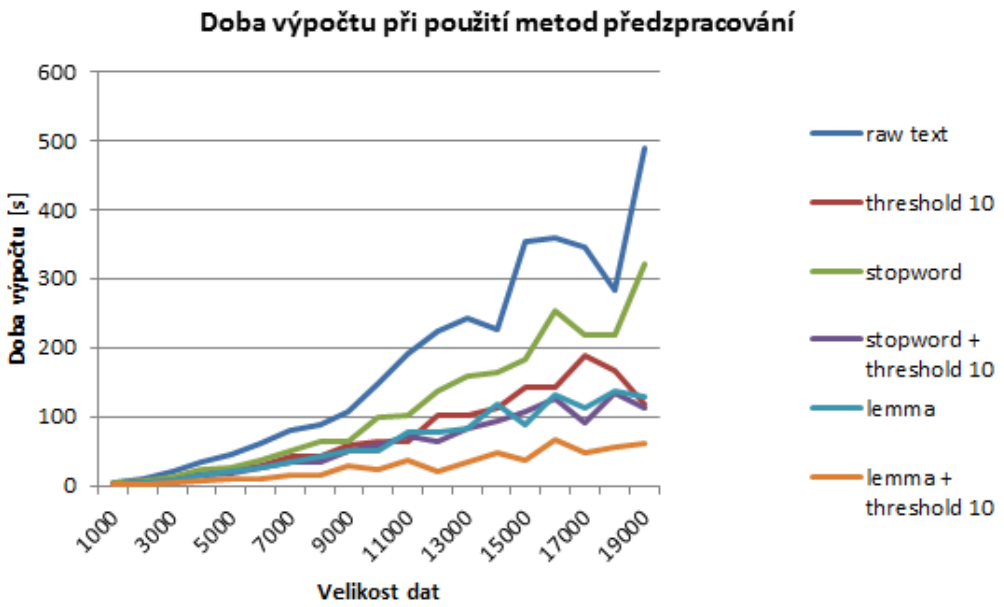
[3] K výpočtu byl použit počítač s konfigurací: Intel Core I7 (3612QM) 2,10GHz, 8GB RAM, WIN7



Obrázek 3.3: Úspěšnost klasifikace při použití metod předzpracování.



Obrázek 3.4: Počet příznaků extrahovaných metodami předzpracování.



Obrázek 3.5: Doba výpočtu (v sekundách) potřebná k určení úspěšnosti klasifikace při použití jednotlivých metod předzpracování.

zapotřebí pouze 3639 lemat. Z toho lze usoudit, že lemata mají v tomto případě přibližně 5x větší váhu než nikterak nefiltrovaná slova.

Problém při použití přísného předzpracování je, že počet příznaků roste jenom velice pomalu s velikostí dat (viz graf 3.4). Extrahované příznaky tak sice mají daleko větší váhu, ale jejich nízký počet nestačí k tomu, aby bylo dosaženo úspěšnosti klasifikace srovnatelné s metodami hloupějšího předzpracování.

Pokud nám ovšem záleží na výpočetním čase , lze tohoto efektu využít. Přísnější předzpracování a tím menší počet příznaků s vyšší váhou samozřejmě znamená rychlejší výpočet (viz graf 3.5).

Další výhodou použití přísnějšího předzpracování je „přehlednost" vektoru příznaků. Hloupé metody předzpracování zanášejí do vektoru příznaků velké množství naprosto nerelevantních výrazů, které mají jenom minimální vliv na úspěšnost klasifikace. Pokud nepoužijeme vyfiltrování stopwords, nejčas-

typ předzpracování	10 nejčastějších příznaků
raw text	a, na, se, je, v, i, to, pro, s, cena
stopword	-, cena, velmi, ovládání, dobře, kvalita, design, moc, +, kvalitní, dobrý
lemma	cena, mít, dobrý, velmi, moci, spokojený, kvalita, doporučovat, velký, kvalitní, dobře

Tabulka 3.4: Nejčastější příznaky získané jednotlivými metodami předzpracování. Omezení minimálního počtu výskytů slova toto pořadí logicky neovlivní.

tějšími příznaky budou slova s největší frekvencí výskytu v českém jazyce (viz tabulka 3.4).

Nestanovení nejnižšího počtu výskytu slova zanáší vektor příznaků výrazy, které se objevují pouze v několika málo recenzích (dost často se navíc jedná o pravopisné chyby). Problém je nicméně s odbornými termíny, které se vyskytují relativně málo, ale mohou nést vysokou míru informace. Lépe než threshold je proto vhodné pro zmenšování vektoru příznaků použít některý z algoritmů selekce příznaků (viz sekce 2.3).

Lemata velice výrazně redukují velikost vektoru příznaků, protože sdružují sobě podobné výrazy. Použitím lemmat se nicméně připravíme o informaci, kterou v sobě obsahují tvary jednotlivých slov.

Metody předzpracování se tedy vyplatí používat, pouze pokud chceme menší množství příznaků, které mají vysokou váhu a jsou přehledné. Pokud chceme použít pouze hrubou sílu a nezáleží nám na době výpočtu, vyplatí se metody předzpracování zcela ignorovat.

Toto ovšem platí pouze pro námi vytvořený dataset. Pro rozdílná data se metody předzpracování mohou ukázat daleko výhodnější.

3.3 Výběr příznaků

Tento experiment je rozdělen na dvě části. V první části jsou představeny základní příznaky, které lze z recenze extrahovat. V druhé části jsou tyto příznaky kombinovány. V obou případech je měřena úspěšnost klasifikace udaná Accuracy a F1 measure (včetně konfidenčního intervalu), velikost vyextrahovaného vektoru příznaků a doba výpočtu.

V tomto experimentu bylo z důvodu výpočetní složitosti použito podmnožiny dat o velikost 20 000 recenzí. Jako klasifikátor byl zvolen algoritmus maximum entropy. V křížové validaci bylo vždy použito 90% recenzí k trénování a 10% k testování.

3.3.1 Základní příznaky

Následující seznam udává množinu příznaků, které byly použity v tomto experimentu.

Počet pozitivních/negativních připomínek Ukázka příznaku, který není konstruován za pomoci slov recenze. Udává pouze počet pozitivních nebo negativních vyjádření, které se v recenzi vyskytují (ze seznamu plusů a mínusů).

Slova Jako příznaky jsou použita všechna slova příspěvku.

Lemmata Lemmata všech slov, která příspěvek obsahuje.

POS tagy Jako příznaky jsou použity POS tagy slov (první 4 nejdůležitější znaky).

Přídavná jména Všechna přídavná jména, která se v příspěvku vyskytují.

Slovesa Všechna slovesa, která se v příspěvku vyskytují.

Bigramy Všechny po sobě jdoucí dvojice slov, které příspěvek obsahuje.

Dvojice slov Všechny možné kombinace dvou slov z příspěvku.

Specifické bigramy Ukázka komplexnější typ příznaku. Obsahuje pouze bigramy typu přídavné jméno + podstatné jméno, podstatné jméno + podstatné jméno, nebo příslovce + podstatné jméno.

Tabulka 3.5 udává naměřené hodnoty. Nejlepších výsledků klasifikátor dosáhl při použití celých slov jako příznaků. Oproti slovům výrazně zaostávají příznaky konstruované z lemmat. K výpočtu na základě lemmat byl nicméně potřebný pouze třetinový čas.

Překvapivě vysoká úspěšnost klasifikace byla dosažena při použití příznaku založeného na počtu pozitivních a negativních připomínek u recenze. Ačkoliv se jedná pouze o dva příznaky, je výsledek srovnatelný s příznaky na základě POS tagů, jichž je několik set. Z toho lze usuzovat, že tyto příznaky jsou pro určení kategorie recenze velice významné.

Složitější příznaky konstruované ze specifických slovních druhů v tomto srovnání propadly. Relativně nejlepších výsledků bylo dosaženo při použití přídavných jmen. Úspěšnosti při použití těchto příznaků se nicméně nepřibližují úspěšnosti při použití všech slov.

Porovnání úspěšnosti klasifikace dvojic slov a bigramů skončilo remízou. Accuracy sice jasně hovoří pro dvojice slov, F1 measure je však v tomto případě porovnatelná (nepřekračuje odchylku konfidenčního intervalu). Výhodou bigramů je ovšem nesrovnatelně nižší výpočetní čas.

Příznak	**Accuracy**	**F1 measure**	**Confidence 95%**	**Počet příznaků**	**Doba výpočtu[s]**
Počet pozitivních / negativních připomínek	0,57	0,32	0,02	2	4
Slova	**0,73**	**0,57**	0,02	97652	307
Lemmata	0,62	0,43	0,02	39487	93
POS tagy	0,66	0,33	0,02	669	22
Přídavná jména	0,50	0,35	0,02	5961	31
Slovesa	0,38	0,27	0,02	2812	14
Bigramy	0,39	0,30	0,02	51224	91
Dvojice slov	0,46	0,32	0,02	32984	1249
Specifické bigramy	0,32	0,25	0,02	21282	45

Tabulka 3.5: Úspěšnost klasifikace při použití jednoduchých příznaků (pro 20 000 recenzí).

3.3.2 Kombinované příznaky

V této části experimentu byly využity příznaky zkonstruované v předchozím kroku. Tyto příznaky byly rozličně nakombinovány a bylo na nich provedeno stejné měření za stejných podmínek. Použité kombinace byly zvoleny tak, aby pokud možno dávaly smysl a demonstrovaly určitý záměr.

Na tomto základě byly vybrány následující kombinace příznaků:

Lemmata + POS tagy Tento typ kombinace příznaků by měl zajistit vrácení informace o tvarech slov, která byla ztracena lemmatizací.

Lemmata + POS tagy + počet pozitivních/negativních připomínek V tomto případě vyzkoušíme, nakolik jednoduchý příznak nezaložený na slovech příspěvku dokáže vylepšit výsledky klasifikace.

Slova + počet pozitivních/negativních připomínek Kombinace příznaku s nejlepším výsledkem z předchozího kroku s příznakem, který je nezávislý na slovech.

Slova + POS tagy Pro porovnání s lemmaty kombinovanými s POS tagy. Jelikož slova již informace o svém tvaru obsahují, jedná se vlastně o duplicitní informaci.

Slova + POS tagy + počet pozitivních/negativních připomínek Opět pro porovnání s obdobným příznakem založeným na lemmatech a pro otestování, ve kterém případě jednoduchý příznak přispěje více k úspěšnosti klasifikace.

Slova + bigramy Pro otestování, kolik informace navíc obsahují bigramy oproti samostatným slovům.

Slova + bigramy + počet pozitivních/negativních připomínek

Slova + dvojice slov Pro porovnání s příznakem slova + bigramy.

Přídavné jména + slovesa Kombinace dvou specifických slovních druhů, teoreticky obsahující vyšší množství informace o sentimentu než ostatní slova příspěvku.

Přídavné jména + slovesa + počet pozitivních/negativních připomínek

Specifické bigramy + slovesa + přídavná jména + počet pozitivních / negativních připomínek Nejkomplikovanější testovaný příznak. Kombinuje bigramy typu přídavné jméno + podstatné jméno, podstatné jméno + podstatné jméno, nebo příslovce + podstatné jméno, slovesa, přídavná jména a počet plusů a mínusů v příspěvku.

Vypočtené výsledky naleznete v tabulce 3.6.

Úspěšnost klasifikace při použití lemmat v kombinaci v POS tagy se prakticky nezměnila oproti úspěšnosti při použití samotných lemmat. Z toho plyne, že POS tagy v tomto případě nedokáží nahradit informaci ztracenou lemmatizací.

Slova v kombinaci s POS tagy mají dokonce nižší úspěšnost než samotná slova. Duplicitní informace obsažená v POS tazích tak klasifikaci ve výsledku uškodila.

Porovnání kombinace bigramů se slovy s kombinací dvojic slov s bigramy dopadlo opět nerozhodně. V obou případech je navíc výsledná úspěšnost klasifikace nižší než při použití samostatných slov.

Složitější kombinace příznaků na základě specifických slovních druhů dopadly znovu jako nejhorší. V tomto případě se model přeučil na trénovacích datech a ztratil schopnost generalizovat.

Samostatnou kapitolou je příznak udávající počet pozitivních a negativních připomínek. Ve všech případech kdy byl zkombinován s jiným příznakem přispěl k zlepšení výsledku klasifikace o 1 až 4%.

Absolutně nejlepšího výsledku (F1 measure 0,58) bylo v tomto experimentu dosaženo při použití celých slov v kombinaci s počet pozitivních a negativní připomínek.

3.4 Srovnání ML algoritmů

V tomto experimentu jsou porovnány 3 algoritmy strojového učení uvedené v této práci (SVM, MaxEnt a Naive Bayes). K experimentu byly jako příznaky použity lemata slov. Velikost dat byla omezena na vzorek 1000 až 20000 recenzí. Při křížové validaci z nich bylo použito 90% trénování a 10% k testování. U jednotlivých iterací byla měřena úspěšnost klasifikace (dána F1 measure) a doba výpočtu.

Graf 3.6 ukazuje úspěšnost klasifikace jednotlivých algoritmů strojového učení v závislosti na velikosti dat. Jak lze z tohoto grafu a v něm uvedených lineárních trendů vyčíst, úspěšnost klasifikace roste jen velmi pomalu s velikostí dat. Algoritmus SVM z tohoto porovnání vychází nejlépe, navíc jako jediný dává dobré výsledky i při použití malého vzorku dat.

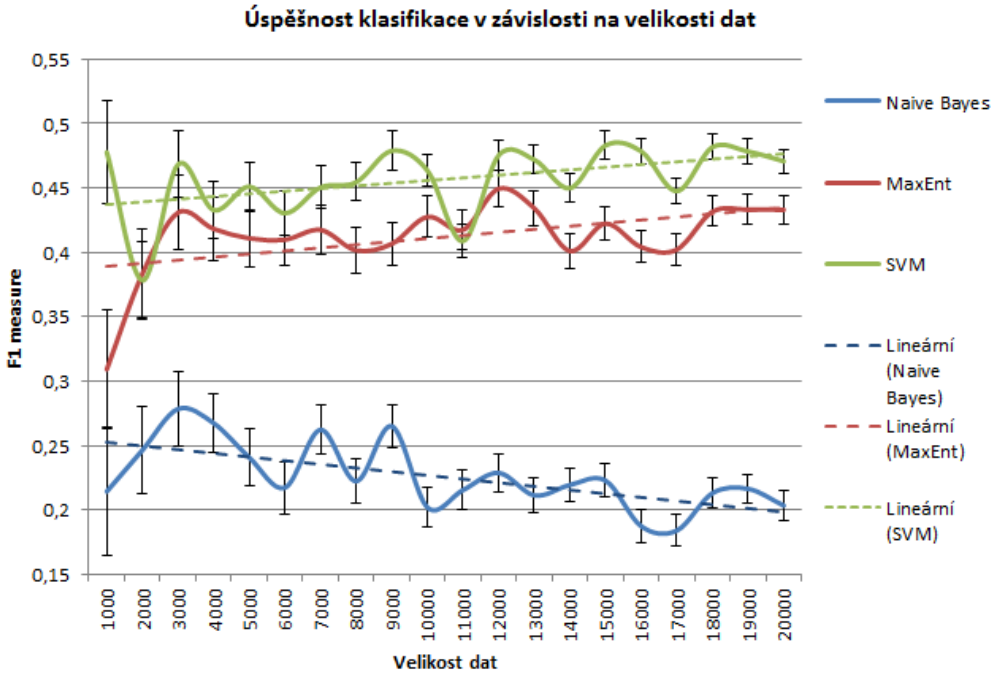
Algoritmus Naive Bayes se v tomto ohledu ukázal jako nepoužitelný. Jeho úspěšnost je o více než 10% nižší než u ostatních dvou algoritmů, navíc vykazuje lehce klesající tendenci úspěšnosti v závislosti na velikosti dat. Jeho jedinou výhodou je rychlost výpočtu (viz graf 3.7) [4].

Algoritmus maximum entropy zaostává za algoritmem SVM pouze o pár procentních bodů. Jeho výhodou je ovšem oproti algoritmu SVM jeho podstatně nižší výpočetní čas. Jak lze vyčíst z grafu 3.7 tento čas navíc s velikostí dat roste daleko pomaleji než u SVM.

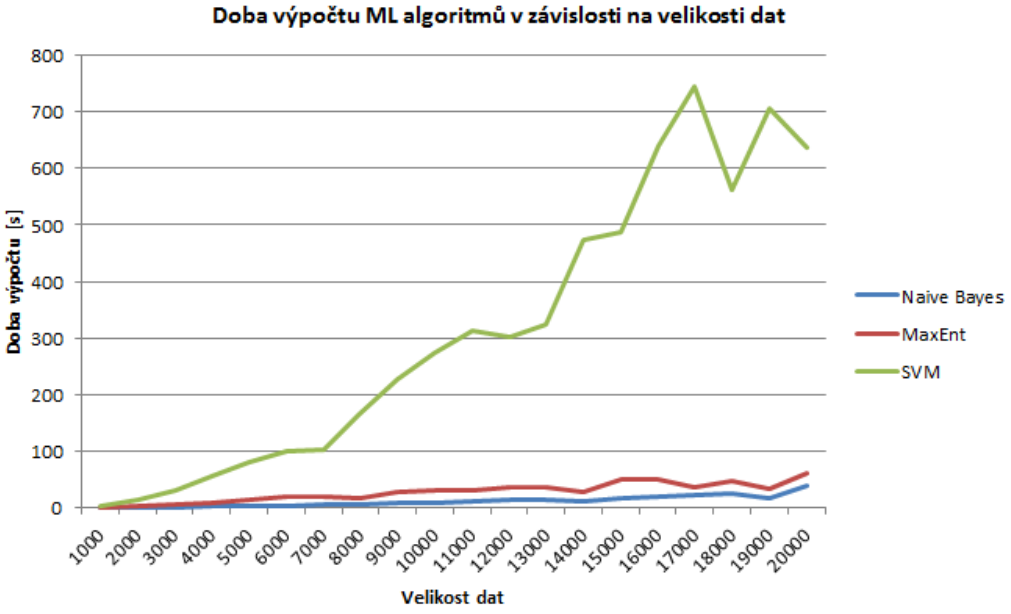
[4] K výpočtu byl použit počítač s konfigurací: Intel Core I7 (3612QM) 2,10GHz, 8GB RAM, WIN7

Příznak	**Accuracy**	**F1 measure**	**Confidence 95%**	**Počet příznaků**	**Doba výpočtu[s]**
Lemmata + slova	0,66	0,43	0,02	40285	186
Lemmata + POS tagy + počet pozitivních / negativních připomínek	0,66	0,45	0,02	40287	271
Slova + počet pozitivních / negativních připomínek	**0,74**	**0,58**	0,02	98997	355
Slova + POS tagy	0,74	0,52	0,02	99579	629
Slova + POS tagy + počet pozitivních / negativních připomínek	0,74	0,55	0,02	99581	749
Slova + bigramy	0,73	0,52	0,02	149348	467
Slova + bigramy + počet pozitivních / negativních připomínek	0,74	0,56	0,02	149350	499
Slova + dvojice slov	0,73	0,53	0,02	128170	2750
Přídavná jména + slovesa	0,56	0,37	0,02	8821	44
Přídavná jména + slovesa + počet pozitivních / negativních připomínek	0,62	0,41	0,02	8823	69
Specifické bigramy + přídavná jména + slovesa + počet pozitivních / negativních připomínek	0,61	0,39	0,02	30424	159

Tabulka 3.6: Úspěšnost klasifikace při použití kombinovaných příznaků (pro 20 000 recenzí).



Obrázek 3.6: Úspěšnost klasifikace v jednotlivých algoritmů strojového učení v závislosti na velikosti dat. Přerušované čáry znázorňují lineární trendy.



Obrázek 3.7: Rychlost výpočtu algoritmů strojového učení(v sekundách) v závislosti na velikost použitých dat.

Algoritmus SVM je proto pro větší data a více příznaků prakticky nepoužitelný, pro malá data však poskytuje nejlepší výsledky.

Z porovnání algoritmů strojového učení proto vychází nejlépe algoritmus maximum entropy, který je dobře použitelný i pro velká data a úspěšnost jeho klasifikace je dostatečná.

3.5 Srovnání algoritmů selekce příznaků

Tento experiment má za úkol porovnat účinnost algoritmů selekce příznaků a vybrat pro ně optimální nastavení přísnosti filtrování. Experiment byl proveden na datovém vzorku o velikosti 120 000 recenzí, při křížové validaci bylo použito 90% recenzí k trénování a 10% k testování. Jako příznaky byla zvolena lemmata a bigramy. Použitým algoritmem strojového učení byl MaxEnt. Měřenými veličinami byly úspěšnost klasifikace (udaná F1 measure) a

velikost redukce vektoru příznaků.

Abychom mohli určit, jaká přísnost filtrování je pro daný algoritmus optimální, byl každý algoritmus testován pro 10 různých filtru od 0 (žádné filtrování) až po 0,9 (nejpřísnější filtrování).Toto nastavení neznamená, že se například při použití hodnoty přísnosti 0,7 odfiltruje 70% všech příznaků. Každý algoritmus generuje jiné rozdělení a je třeba určit experimentálně, kolik procent příznaků algoritmus odfiltruje při použití zadané přísnosti.

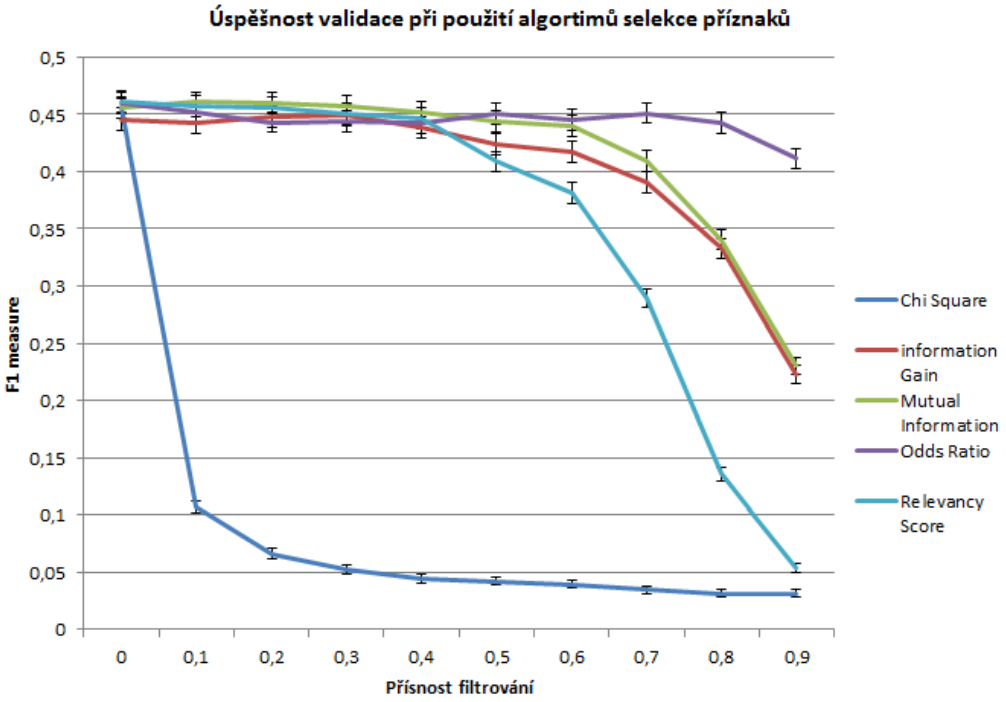
Graf 3.8 ukazuje závislost úspěšnosti klasifikace na přísnosti filtrování. Tento graf nicméně nezohledňuje procento vyfiltrovaných příznaků, pouze nastavení přísnosti. Tento problém dobře vystihuje graf 3.9. Nejlepší výsledky tak sice dostáváme při použití algoritmu Odds Ratio, nicméně se ukazuje, že je příliš „měkký" - odfiltrovává pouze malé množství příznaků. Chi Square trpí přesně opačným problémem - již pro nízké hodnoty přísnosti odfiltruje valnou většinu příznaků.

Pokud chceme porovnat účinnost algoritmů, musíme použít zobrazení, které zohledňuje procento vyfiltrovaných příznaků a úspěšnost klasifikace. Graf 3.10 udává projekci těchto dvou veličin v prostoru.

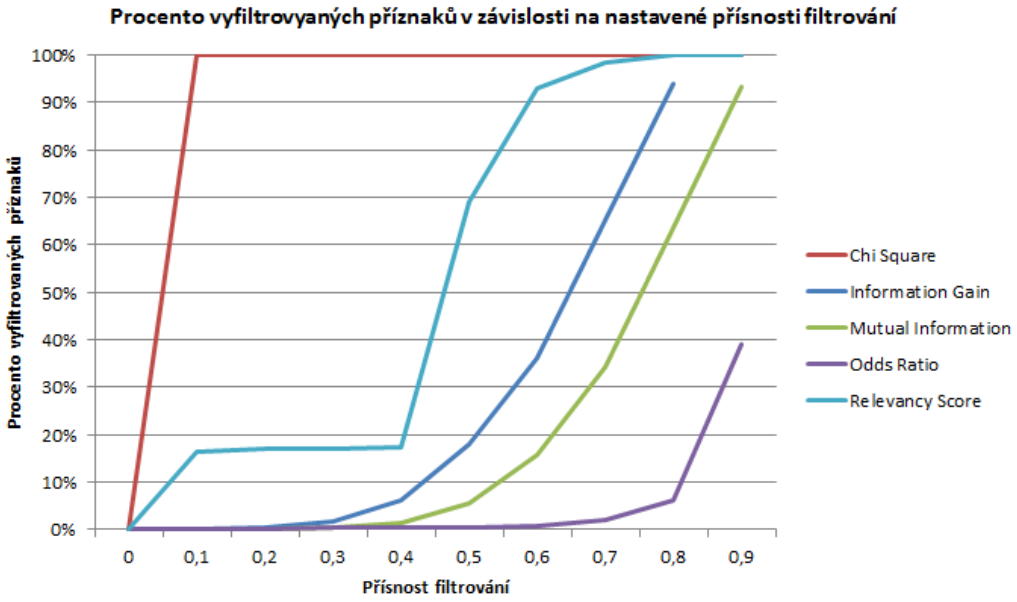
Pro mírné nastavení filtrování z tohoto porovnání vychází nejlépe algoritmus Relevancy Score, který při odfiltrování přibližně 20% příznaků generuje výsledky s prakticky shodnou úspěšností. Hodnoty při použití přísnosti 0,1-0,4 jsou prakticky totožné. Jen o málo horší výsledky produkují algoritmy Mutual Information a Information Gain s přísností 0,6.

Pro hodnotu filtrování okolo 40% příznaků vychází nejlépe trojice algoritmů Information Gain (přísnost 0,7), Mutual information (0,7) a Odds Ratio (0,9).

Nastavení, která generují nejlepší výsledky při odfiltrování největšího množství příznaků, se nacházejí co nejblíže pravému hornímu rohu grafu 3.10. Z tohoto porovnání vychází nejlépe algoritmus Relevancy Score (0,6), který i při odfiltrování 93% příznaků stále dává použitelné výsledky. Další dobré výsledky se nacházejí v pravém horním kvadrantu grafu. Konkrétně se jedná o algoritmy Relevancy Score (0,5 a 0,7), Information Gain (0,8 a 0,9) a Mutual



Obrázek 3.8: Úspěšnost klasifikace v závislosti na přísnosti nastavení filtrování.



Obrázek 3.9: Procento vyfiltrovaných příznaků v závislosti na přísnosti filtrování.

information (0,8).

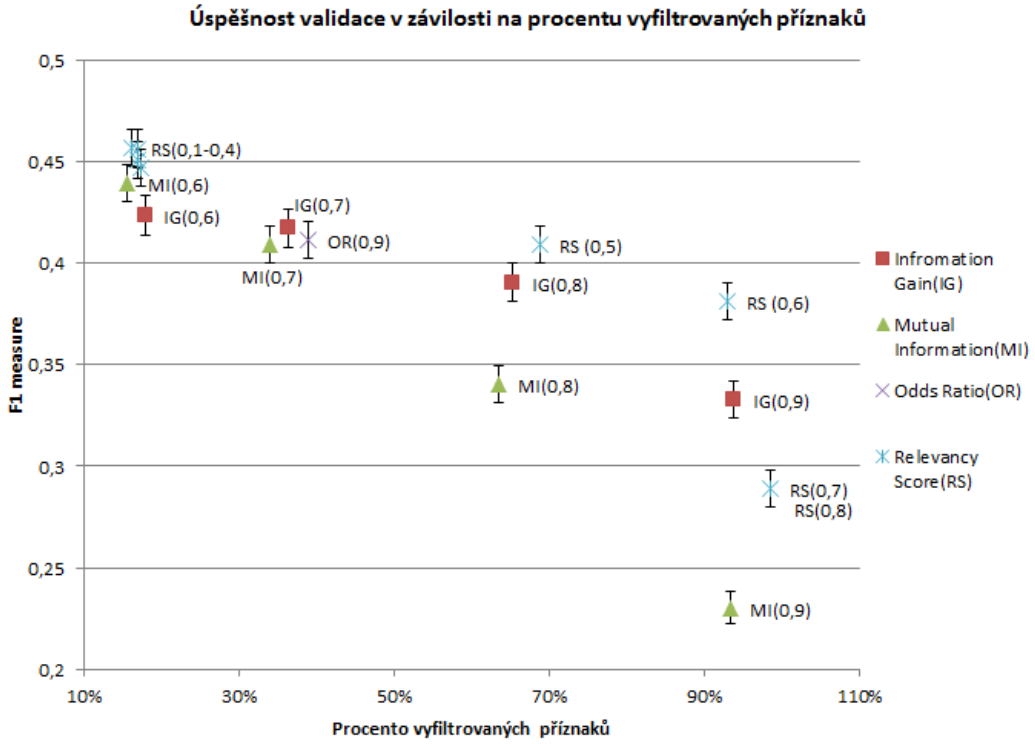
Tyto vypočtené hodnoty jsou silně závislé na použitých datech a z nich vyextrahovaných příznacích. Pro jiný klasifikační problém mohou být úspěšná zcela rozdílná nastavení.

3.6 Použití nejlepších výsledků

V tomto experimentu byly použity nastavení generující nejlepší výsledky v předešlých experimentech.

Klasifikátor Maximum entropy

Předzpracování Použití celých slov (žádné předzpracování)



Obrázek 3.10: Závislost úspěšnosti validace na procentu vyfiltrovaných příznaků. Nejlepší výsledky se nachází blízko pravého horního rohu (vysoká úspěšnost validace a velká redukce vektoru příznaků). Čísla v závorkách udávají použitou přísnost filtrování. Byly stanoveny nenulové počátky os grafu k odfiltrování nezajímavých hodnot. Algoritmus Chi Square nemá ani jednu hodnotu v takto omezeném intervalu.

Algoritmus selekce	**Accuracy**	**F1 measure**	**Konfidence 95%**	**Počet příznaků**	**Redukce vektoru příznaků**	**Čas[s]**
no selector	0,75	0,63	0,01	375470	0	4505
RS(0,1)	**0,77**	**0,65**	0,01	375470	0,186	4172
RS(0,5)	0,61	0,46	0,01	375470	0,937	2961
RS(0,6)	0,36	0,34	0,01	375470	0,982	3779
RS(0,7)	0,17	0,21	0,01	375470	0,994	3843
MI(0,6)	0,73	0,59	0,01	375470	0,003	4863
MI(0,7)	0,71	0,55	0,01	375470	0,026	4695
MI(0,8)	0,68	0,50	0,01	375470	0,204	5772
IG(0,6)	0,73	0,59	0,01	375470	0,003	4547
IG(0,7)	0,71	0,55	0,01	375470	0,026	3634
IG(0,8)	0,68	0,50	0,01	375470	0,204	4022
IG(0,9)	0,51	0,36	0,01	375470	0,715	3847
OR(0,9)	0,74	0,56	0,01	375470	0,318	5090

Tabulka 3.7: Výsledky klasifikace při použití nejlepších nastavení algoritmů selekce příznaků.

Příznaky Kombinace celých slov (nulový threshold) s počtem pozitivních a negativních připomínek.

Selektory příznaků Použito 12 nejlepších identifikovaných nastavení algoritmů selekce příznaků.

Pro křížovou validaci byl použit celý dostupný dataset (141 691 recenzí), 90% dat bylo použito k trénování a 10% k testování. Tabulka 3.7 ukazuje vypočtené výsledky. Na první řádce tabulky jsou pro srovnání uvedeny výsledky dosažené při vynechání algoritmu selekce příznaků.

Nejlepších výsledků opět dosáhl algoritmus Relevancy Score. Tento algoritmus při nízkém nastavení přísnosti filtrování dosáhl vyšší úspěšnosti klasifikace, než při klasifikaci bez algoritmu selekce (a to přesto, že odfiltroval

18,6% příznaků). Vyšší nastavení filtru se ukázalo jako již poměrně přísné, výsledky při odfiltrování více než 90% příznaků jsou nicméně stále použitelné.

Jediné testované nastavení Algoritmu Odds Ratio dosáhlo rovněž velmi dobrých výsledků - při odfiltrování 31,8% příznaků klesla úspěšnost klasifikace pouze o 6%.

Algoritmy Mutual Information a Information Gain dosáhly prakticky shodné úspěšnosti, která byla v obou případech nižší než při použití Relevancy Score nebo Odds Ratio.

Poznámka

Na tomto místě je třeba upozornit na to, že většina výše uvedených experimentů je velice výpočetně náročná. Pro většinu experimentů bylo proto využito buď omezené velikosti dat, nebo parametrů, které umožňují získání výsledků na běžném stroji v rozumném čase. Pro experiment porovnávající algoritmy selekce příznaků bylo pro větší přesnost výsledků využito prostředků výpočetního metacentra.

Kapitola 4

Závěr

Hlavním záměrem této práce bylo prostudovat metody strojového učení a použít je k analýze sentimentu. V kapitole 1 jsou definovány základní pojmy týkající se této problematiky, motivace pro vytvoření celého projektu a případy jeho užití. Dále jsou představeny výzkumy zabývajících se obdobnou tematikou a problémy týkající se analýzy sentimentu v češtině.

Kapitola 2 je rozdělena do čtyř částí. V první části (2.1) je uvedena definice příznaků a jsou zde popsány způsoby, jimiž lze příznaky získat. Rovněž jsou zde představeny metody předzpracování příznaků a základní postupy pro zvyšování jejich váhy.

Druhá část této kapitoly (2.2) se věnuje samotným algoritmům strojového učení. Je zde uvedena formální definice tří algoritmů (Naive Bayes, Maximum entropy a SVM), které jsou používány napříč uvedenými experimenty.

V kapitole 2.3 je definován algoritmus selekce příznaků a je zde uvedeno pět konkrétních příkladů takového algoritmu. U každého z nich je uveden jeho formální předpis a případná omezení týkající se jeho použití.

Poslední část kapitoly (2.4) obsahuje informace týkající se testování úspěšnosti klasifikace. Je zde představena confusion matrix a F1 measure. Tyto prostředky jsou použity k získání relevantních výsledků i při použití nevyváženého datasetu.

Kapitola 3 obsahuje šest experimentů. Tyto experimenty byly koncipovány tak, aby pokryly všechny problematické oblasti v analýze sentimentu a otestovaly slibné kombinace nastavení parametrů.

Nejprve je předveden způsob získání označkovaných dat (3.1), kterých je využito k testování úspěšnosti klasifikace. Jsou zde uvedeny základní statistiky těchto dat a jejich rozdělení do kategorií.

Experiment 3.2 se zabývá problematikou předzpracování příznaků. U každé použité techniky předzpracování je uvedena úspěšnost klasifikace při jejím použití. Z tohoto porovnání lze vyčíst, že se příznaky vyplatí předzpracovávat pouze tehdy, pokud jsme omezeni výpočetní náročností klasifikace.

Sekce 3.3 porovnává úspěšnosti klasifikace při použití různých postupů konstruování příznaků. Tato sekce je rozdělena do dvou částí. Nejprve je představena základní množina příznaků, poté jsou její prvky kombinovány tak, aby bylo dosaženo co nejlepších klasifikačních výsledků. Těch bylo dosaženo při použití příznaků založených na nikterak nezpracovaných slovech v kombinaci s příznakem udávajícím počet pozitivních a negativních připomínek uvedených v recenzi.

Porovnání algoritmů strojového učení je obsahem experimentu 3.4. Z tohoto porovnání vychází jednoznačně nejhůře algoritmus Naive Bayes, jehož výsledky byly prakticky nepoužitelné. Nejvyšší úspěšnosti klasifikace dosáhl algoritmus SVM, nicméně jeho vysoká výpočetní náročnost znemožňuje jeho použití pro klasifikaci větších dat. Vítězem tohoto srovnání se proto stal algoritmus Maximum entropy, který poskytl dostatečně přesné výsledky klasifikace a rozumný výpočetní čas.

Nejdůležitějším experimentem je ovšem experiment 3.5, který porovnává účinnost algoritmů selekce příznaků. Jsou zde popsána nejlepší identifikovaná nastavení selektorů, bereme-li jako určující faktory úspěšnosti procento vyfiltrovaných příznaků a dosaženou úspěšnost klasifikace.

Z porovnávaných algoritmů dosáhl jednoznačně nejlepších výsledků algoritmus Relevancy score. Algoritmy Mutual information a Information gain do-

sáhly obdobné úspěšnosti a v porovnání skončily jako druhé nejlepší.

U algoritmů Odds ratio a Chi square byl identifikován problém týkající se jejich přílišné „měkkosti" nebo naopak „tvrdosti". Algoritmus Odds ratio je nicméně se správným nastavením použitelný, zatímco u algoritmu Chi square nedosáhlo žádné testované nastavení použitelných výsledků.

V rámci posledního experimentu (3.6) jsou použita nastavení, která generovala nejlepší výsledky v předchozích experimentech (nejlepší algoritmus strojového učení, metoda předzpracování, metoda tvorby příznaků a metoda selekce příznaků). Jsou zde ukázány celkově nejlepší výsledky klasifikace, jakých bylo v rámci této práce dosaženo.

V porovnáním s náhodným určováním kategorií (která by pro tři kategorie měla dosahovat úspěšnosti 33% F1 measure), se nám podařilo dosáhnout přibližně o 32% F1 measure lepších výsledků. Implementace algoritmů selekce příznaků byla rovněž úspěšná. Podařilo se dosáhnout přibližně shodných výsledků klasifikace i při odfiltrování části trénovacích dat. Rovněž se podařilo minimalizovat ztrátu úspěšnosti klasifikace i při odfiltrování valné většiny příznaků.

Námětem pro další práci je otestování účinnosti algoritmů selekce příznaků na jiném než námi používaném datasetu. Rovněž je prostor pro další výzkum, který se může věnovat zlepšování dosažených výsledků.

References

Abbasi, A., Chen, H., and Salem, A. (2008). Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Trans. Inf. Syst.*, 26(3):12:1–12:34.

Abdul-Mageed, M., Kübler, S., and Diab, M. (2012). Samar: a system for subjectivity and sentiment analysis of arabic social media. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '12, pages 19–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arun Kumar, M. and Gopal, M. (2010). Fast multiclass svm classification using decision tree based one-against-all method. *Neural Process. Lett.*, 32(3):9160:311–9160:323.

Biber, D. (1988). *Variation across Speech and Writing*. Cambridge University Press, Cambridge, UK.

Debole, F. and Sebastiani, F. (2005). An analysis of the relative hardness of reuters-21578 subsets: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(6):584–596.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Comput. Surv.*, 27(3):326–327.

Forman, G., Guyon, I., and Elisseeff, A. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

Guyon, I. and Elisseeff, A. (2006). An introduction to feature extraction. In Guyon, I., Nikravesh, M., Gunn, S., and Zadeh, L., editors, *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 1–25. Springer Berlin Heidelberg.

Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, EACL '97, pages 174–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hearst, M. A. (1992). Text-based intelligent systems. chapter Direction-based text interpretation as an information access refinement, pages 257–274. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Huettner, A. and Subasic, P. (2000). Fuzzy typing for document management. In *ACL 2000 Companion Volume: Tutorial Abstracts and Demonstration Notes*, pages 26–27.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Ls8-report 23, Universitat Dortmund. URL: http://www.cs.cornell.edu/People/tj/publications/joachims_97b.pdf.

Joshi, S., Jayadeva, Ramakrishnan, G., and Chandra, S. (2012). Letters: Using sequential unconstrained minimization techniques to simplify svm solvers. *Neurocomput.*, 77(1):253–260.

Karlgren, J. and Cutting, D. (1994). Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of the 15th conference on Computational linguistics - Volume 2*, COLING '94, pages 1071–1075, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kessler, B., Numberg, G., and Schütze, H. (1997). Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the Eu-*

ropean Chapter of the Association for Computational Linguistics, ACL '98, pages 32–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kohavi, R. and Provost, F. (1998). Glossary of terms. *Machine Learning*, 30(2-3):271–274.

Manning, C. (2013). Maxent models and discriminative estimation. Stanford University, Natural Language Processing.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Mayfield, E. and Penstein-Rosé, C. (2010). Using feature construction to avoid large feature spaces in text classification. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1299–1306, New York, NY, USA. ACM.

Mejova, Y. and Srinivasan, P. (2011). Exploring feature definition and selection for sentiment classifiers. In Adamic, L. A., Baeza-Yates, R. A., and Counts, S., editors, *ICWSM*. The AAAI Press.

Meyer, D. and Wien, T. U. (2001). Support vector machines. the interface to libsvm in package e1071.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Ptaszynski, M., Rzepka, R., Araki, K., and Momouchi, Y. (2012). Automatically annotating a five-billion-word corpus of japanese blogs for affect and sentiment analysis. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '12, pages 89–98, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sack, W. (1994). On the computation of point of view. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, AAAI'94,

pages 1488–, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Saif, H., He, Y., and Alani, H. (2012). Semantic sentiment analysis of twitter. In *Proceedings of the 11th international conference on The Semantic Web - Volume Part I*, ISWC'12, pages 508–524, Berlin, Heidelberg. Springer-Verlag.

Shao, X., Cherkassky, V., and Li, W. (2000). Measuring the vc-dimension using optimized experimental design. *Neural Comput.*, 12(8):1969–1986.

Singh, S. R., Murthy, H. A., and Gonsalves, T. A. (2010). Feature selection for text classification based on gini coefficient of inequality. *Journal of Machine Learning Research - Proceedings Track*, 10:76–85.

Tong, R. M. (2001). An operational system for detecting and tracking opinions in on-line discussion. In *Proceedings of the Workshop on Operational Text Classification (OTC)*.

Tromp, E. (2012). *Multilingual Sentiment Analysis on Social Media: An Extensive Study on Multilingual Sentiment Analaysis Performed on Three Different Social Media*. Lambert Academic Publishing.

Tsytsarau, M. and Palpanas, T. (2012). Survey on mining subjective data on the web. *Data Min. Knowl. Discov.*, 24(3):478–514.

Turney, P. D. and Littman, M. L. (2002). Unsupervised learning of semantic orientation from a hundred-billion-word corpus. *CoRR*, cs.LG/0212012.

Uchyigit, G. (2012). Experimental evaluation of feature selection methods for text classification. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 1294–1298.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Wang, H., Can, D., Kazemzadeh, A., Bar, F., and Narayanan, S. (2012). A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 115–120, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zheng, Z., Wu, X., and Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89.

Příloha A

Seznam použitých stopwords

... . , : ; ! ? ( ) / * "% & že než a s k o i u v z že dnes cz těmto budeš budem byli jseš můj svůj ta tomto tohle tuto tyto jej zda proč máte tato kam tohoto kdo kteří mi něm tom tomuto mět nic proto kterou byla toho protože asi ho naši napište re což těm takže svých její svými jste aj tu tedy teto bylo kde ke pravě ji nad nejsou či pod těma mezi přes ty pak všem ani když však neg jsem tento aby jsme před pta jejich byl ještě až bez také pouze prvně vaše který nás nový tipy pokud strana jeho své jiné zprávy nové není vás jen podle zde už být více bude již než které by která co nebo ten tak mě při od po jsou jak další ale si se ve to jako za zpět ze do pro je na atd atp jakmile přičemž jí on ona ono oni ony my vy ji mi mne jemu tomu těm němu němuž jehož jelikož jež jakož načež má zatím já všechen

Příloha B

Programátorská dokumentace

Aplikace použitá k výpočtu experimentů v této práci obsahuje jednoduché API, umožňující konstrukci dalších algoritmů selekce příznaků. Tato dokumentace demonstruje, jak takový algoritmus vytvořit.

Základem aplikace je rozhraní `Filter` obsahující jedinou metodu `filter`.

```
/**
 * Basic interface for implementing
 * feature selection algoritms.
 */
public interface Filter {
  /**
   * Method for filtering unwanted features.
   * @param tfvl feature vector
   * @param minWeight minimum feature weight
   * @param trainer instance of FilterTrainer
   * @return percentage of filtered features
   * @throws IllegalArgumentException should be thrown when
       minWeight is out of selected bounds or trainer is
      not trained
   */
  public double filter(TrainingFeatureVectorList tfvl,
     double minWeight, FilterTrainer trainer)
      throws IllegalArgumentException;
```

```
}
```

Implementacemi tohoto rozhraní jsou jednotlivé algoritmy selekce příznaků. Ukázkové implementace všech algoritmů použitých v této práci naleznete v balíku `selectors`. Společná logika těchto algoritmů se nachází ve třídě `AbstractFeatureSelector`.

Implementace metody filter by měla obsahovat tyto 3 kroky.

1. určení váhy jednotlivých příznaků
2. normalizace vah na požadovaný interval (není nezbytně nutné, ale je vhodné pro vyšší přehlednost)
3. odfiltrování příznaků majících nižší váhu než je požadovaná

Návratovou hodnotou této funkce by mělo být číslo udávající podíl vyfiltrovaných příznaků. Tato funkcionalita není nutně vyžadovaná, pokud nepotřebujeme statistiku filtrování.

Vektor příznaků obsahuje proměnná `tfvl`. Informace o pravděpodobnostech extrahovaných z tohoto vektoru nám poskytne proměnná `trainer`, což je libovolná implementace rozhraní `FilterTrainer`.

```
/**
 * Interface for constructing feature selection filters.
 */
public interface FilterTrainer {

  /**
   * Extracts probibalites from feature vector list.
   * Should allways be called first.
   * @param tfvl feature vector list
   */
  public void train(TrainingFeatureVectorList tfvl);

  /**
```

```
     * Get number of labels.
     * @return number of labels
     */
    public int getLabelCount();

    /**
     * Get number of features.
     * @return number of features
     */
    public int getFeatureCount();

    /**
     * Gets probability of selected label.
     * @param label selected label
     * @return probability of label
     */
    public double getLabelProbability(int label);

    /**
     * Gets improbability of selected label.
     * In most cases this will be complementary to selected
         label probability.
     * @param label selected label
     * @return probability of label
     */
    public double getLabelImprobability(int label);

    /**
     * Returns probability that selected feature (represented
         by its index)
     * will appear in selected label.
     * @param index index of selected feature in feature
         vector
     * @param label selected label
     * @return probability that given feature is in selected
         label
     */
    public double getFeatureProbability(int index, int label)
        ;
```

```
    /**
     * Returns improbabbility that selected feature
     * will appear in selected label. In most cases
     * this probability will be complementary to
     * feature probability.
     * @param index index of selected feature in feature
        vector
     * @param label selected label
     * @return improbability that given feature is in
        selected label
     */
    public double getFeatureImprobability(int index, int
       label);

    /**
     * Cumulates probability that feature will exists.
     * In most cases this will be 1 / labelcount.
     * @return existence probability
     */
    public double getExistenceProbability();

    /**
     * Cumulates probability thatl feature will not exist.
     * In most cases this will be 2 / labelcount.
     * @return nonexistence probability
     */
    public double getNonexistenceProbability();

    /**
     * Returns iterator over vector of features.
     * This iterator should be used for filtering out
        features.
     * @return iterator over feature vector
     */
    public Iterator<Integer> iterator();

}
```

Můžete použít vlastní implementaci tohoto rozhraní, nebo již hotové řešení obsažené ve třídě `BasicFilterTrainer`. Metody této třídy vracejí informace o pravděpodobnostech potřebných k vypočtení vah příznaků v algoritmech selekce (bližší informace o funkcionalitě metod obsahuje JavaDoc). Nejprve je však v třídě implementující toto rozhraní zavolat metodu `train`, která zajistí vypočtení těchto pravděpodobností. Součástí tohoto rozhraní je také iterátor, který umožňuje mazání příznaků z třídy implementující rozhraní `Filter`.

V okamžiku, kdy máme připravené implementace těchto dvou rozhraní, můžeme provést vlastní filtraci.

```
FilterTrainer filterTrainer = new BasicFilterTrainer();
filterTrainer.train(tfvl);
Filter filter = new MutualInformation();
tfvlReduction = filter.filter(tfvl, minWeight,
   filterTrainer);
```

Nejprve vytvoříme instanci rozhraní `FilterTrainer` (zde je použita třída `BasicFilterTrainer`) a zavoláme metodu `train`, které předáme jako parametr vektor příznaků. Takto natrénovanou třídu pak předáme algoritmu selekce příznaků (zde je použita instance třídy `MutualInformation`) tak, že zavoláme metodu `filter`. Tato metoda požaduje jako parametry vektor příznaků, natrénovanou instanci rozhraní `FilterTrainer` a minimální váhu pro filtraci. Návratovou hodnotou této metody je podíl vyfiltrovaných příznaků.