Mendelova univerzita v Brně
Provozně ekonomická fakulta

Masivní sémantická analýza on-line textů strojovým učením

Diplomová práce

Vedoucí práce:
doc. Ing. Jan Žižka, CSc.

Bc. Pavel Netolický

V Brně, 2016

Zadání práce

Rád bych poděkoval svému vedoucímu práce doc. Ing. Janu Žižkovi, CSc. za poskytnutí důležitých materiálů, množství cenných rad a věnovaného času při vypracovávání této diplomové práce. Dále bych chtěl poděkovat doc. Ing. Františkovi Dařenovi, Ph.D. za poskytnutí zdrojových textových dat a spolupráci. V neposlední řadě chci také poděkovat své rodině a přátelům za pomoc a podporu při tvorbě závěrečné práce, stejně jako v průběhu celého mého studia.

Abstract

Netolický, P. *Massive semantic analysis of on-line texts in machine learning*. Diploma thesis. Brno: Mendel University, 2016.

The diploma thesis deals with mining knowledge from massive amounts of text data. The thesis is based on a comparison between incremental (Hoeffding tree) and batch (J48) approach to training. In the work is designed a series of experiments aimed at comparing algorithm J48 and Hoeffding tree. On the basis of the performed experiments are evaluated different factors, which influence results and course of classification using these algorithms.

Keywords

Text mining, Hoeffding tree, J48, data stream, sentiment analysis, Weka, MOA

Abstrakt

Netolický, P. *Masivní sémantická analýza textů strojovým učením*. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.

Diplomová práce se zabývá oblastí dolování znalostí z masivního objemu textových dat. Práce je založena na srovnání inkrementálního (Hoeffdingův strom) a dávkového (J48) přístupu k tréninku. V rámci práce je navržena série experimentů zaměřených na porovnání algoritmu J48 a Hoeffdingova stromu. Na základě provedených experimentů jsou zhodnoceny různé vlivy, které ovlivňují výsledky a průběh klasifikace těmito algoritmy.

Klíčová slova

Text mining, Hoeffdingův strom, J48, datový tok, analýza názorů, Weka, MOA

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Masivní sémantická analýza on-line textů strojovým učením**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně, dne 23. května 2016 .....

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

1.1 Úvod

Vznikají tak nové informace a organizace, které jsou schopny data z různých zdrojů propojit a uvědomit si jejich cenu, dokáží této příležitosti využít. Velkým úložištěm informací je především internet. Data jsou ukládána v různých systémech a formátech, takže jejich spojování může být problém. Objem dat je tak velký, že je není možné analyzovat ručně ani se dívat do vytvořených tabulek či reportů a hledat v nich odpovědi na pokládané otázky. Do popředí se tak dostává problematika masivního (obrovského) objemu dat. Ukazuje se jako nutnost tato data zpracovávat automatizovaně. Použití manuální techniky zpracování není možné aplikovat tak, abychom dosáhli výsledků v přijatelném čase.

Použití analytických procesů přináší další problém, kterým je nedostatek dostatečně kvalifikovaných analytiků, kteří jsou schopni s daty pracovat a získávat z nich informace. Analytické metody se neustále vyvíjí, což analytikovi poskytuje řadu možností jak k problému přistupovat. S technickým pokrokem a nově dostupnými typy dat (jako jsou různá prostorová data) vznikají otázky jakým způsobem nejlépe tato data prozkoumat. Velkého významu nabývá dolování znalostí z nestrukturalizovaných dat (textu, zvuku, obrazu), která není možné jednoduše zpracovávat jako data strukturovaná. Velké popularitě se těší různé sociální sítě, které se stávají shromaždištěm názorů a komentářů. Tato data zůstávají do značné míry nevyužitá. Z těchto dat lze vyvodit různá hodnocení či doporučení. Nové techniky a metody jak získat potřebné znalosti jsou proto nezbytné. (Acrea, 2016)

V současnosti je patrný narůstající rozdíl mezi generováním dat a jejich porozuměním. Množství dat roste ve velkým tempem a tím je patrný proporcionální nesoulad porozumění dat člověkem. Odhalování znalostí v získaných datech se tak stává ceněnou výhodou. Důležité je hledat určité vzory v datech, což ovšem není nic nového. Lidé již od počátku hledají určité vzory v datech. Již staří lovci hledali vzory v chování migrace zvířat. Tento poznatek lze převést i na dnešní moderní svět.

Prostřednictvím data miningu jsou uložena data elektronicky a vyhledávána automatizovaně nebo alespoň za využití osobního počítače. Ekonomové, statistici, prognostici si již dlouho pohrávají s myšlenkou automatizovaného hledání, které dopomáhá k identifikování, potvrzení a predikci. Nárůst databází v posledních letech způsobuje, že nás databáze provází při každodenní činnosti. Díky data miningu se dostávají do popředí nové obchodní možnosti.

Stroje, na kterých bychom mohli provádět hledání, se stávají samozřejmostí. Zde přichází na řadu příležitost pro zvyšování významu dolování dat. Dolování znalostí se stává naší jedinou nadějí na odhalení skrytých vzorů v datech. Inteligentně analyzovaná data jsou cenným zdrojem. To vše může ve své podstatě vést k získání nových znalostí. (Witten, Frank a Hall, 2011; Shmueli et al., 2011)

1.2 Cíl práce

Cílem této práce je dolování znalosti z dat (data mining) pro mimořádně rozsáhlá (masivní) data. Jedná se o data, u nichž nastává problém tréninku klasifikačních algoritmů v celku. V současnosti vznikají nové algoritmy a systémy, které umožňují strojové učení prostřednictvím tzv. proudu dat (*data stream*), kdy se algoritmus trénuje průběžně (iterativně) s přicházejícími daty, aniž by se data musela předem akumulovat do jedné velké dávky. Jedná se o typický moderní problém, mezi který patří např. dolování znalosti z dat přicházejících ze sociálních sítí.

Jedním z hlavních cílů této práce je prozkoumat přístup inkrementálního zpracování a provést komparaci s dávkovým přístupem aplikovaným na zvolená textová data. Zaměřeno bude především na inkrementální trénink vybraných klasifikačních algoritmů a jeho porovnání s tréninkem dávkovým. Důraz bude kladen především na hledisko objemu dat a obměny dat.

Dále bude navržena a uskutečněna série experimentů s reálnými daty, která budou detailně prozkoumána. Porovnány budou vlastnosti zvolených inkrementálních a dávkových algoritmů. Důraz bude kladen především na vhodné alternativní reprezentace vstupních textových dat a zjištění jejich vlivu na průběh tréninku. U jednotlivých klasifikátorů bude zjištěno, co a do jaké míry ovlivňuje výsledky klasifikace pro vhodně stanovená kritéria.

Práce bude zaměřena na návrh vhodné metodiky přípravy dat a experimentů pro dolování znalosti ze zmíněných velmi objemných textových dat a testování vlivu přípravy na získanou výslednou informaci a znalost (např. alternativní algoritmy, závislost na objemu dat aj.).

2 Teoretická část

Data mining neboli dolování znalostí z dat je v současnosti stále více využívaným prostředkem k získávání důležitých a tím zajímavých (skrytých) znalostí z nejrůznějších dat nashromážděných v rámci rozlišných oborů. Zisk nových a zároveň cenných informací v dnešním konkurenčním prostředí nabývá na důležitosti. Nejprve je vhodné čtenáře uvést do obecného kontextu oblasti strojového učení a popsat základní pojmy týkající se dané oblasti.

2.1 Strojové učení

Strojové učení (*Machine Learning*) umožňuje řešení problémů za pomocí stroje – počítače. Je navíc jedním z mnoha odvětví umělé inteligence. Hlavní výzkumnou oblastí je určení jak se mohou počítačové programy automaticky naučit rozpoznávat složité vzory a dělat inteligentní rozhodnutí na základě dat. Často uváděným příkladem je rozpoznávání spamu od definované emailové adresy. Vstupem je emailový dokument, jenž je souborem znaků. Výstupem algoritmu jsou poté dvě hodnoty určující, zda zadaný email můžeme automatizovaně rozpoznat, zda daný email je spam na základě naučení se ze série případů, které označíme za spam (Han a Micheline, 2011).

Jak vysvětluje (Alpaydin, 2010) strojové učení umožňuje využít výpočetní techniku k optimalizaci výkonnosti kritérií na základě použití vzorových dat či předchozích zkušeností. Dostáváme tím k dispozici model, který je definovaný parametry. Učící proces v tomto případě představuje spuštění programu, který optimalizuje parametry modelu pomocí trénovacích dat nebo předešlých zkušeností. Model může být **prediktivní** (k vytvoření predikcí využitelných pro budoucnost) či **deskriptivní** (k získání znalostí z dat) či také oboje zároveň. Pro potřeby strojového učení je užívána teorie statistiky pro tvorbu matematických modelů. Smyslem úloh je vytvořit závěr ze vzorku dat. Zde lze ilustrovat klasické problémy strojového učení, které velmi úzce souvisí s dolováním dat.

2.1.1 Učení s učitelem

Učení s učitelem se stává synonymem pro klasifikaci. Je to učení založené na učení se z trénovacích dat. Algoritmus je založen na analyzování trénovacích dat a vytvoření funkce, již lze použít pro analýzu nových neznámých vzorků. Schopnost algoritmu zobecňovat z trénovacích dat na data neznámá je nezbytnou součástí k tomu, abychom mohli takovou funkci vytvořit. Tento případ lze přirovnat k učení lidí a zvířat, pak se jedná o učení označované *konceptuální učení*. (Zhu a Goldberg, 2009)

Jednou z variant je tzv. **učení on-line**, kdy data jsou předkládána algoritmu postupně vzorek po vzorku. Taková situace nastává v případě, kdy máme streamovaná data, která algoritmus musí zpracovat za chodu. On-line učení je užitečné v případech omezené výpočetní rychlosti a paměťového prostoru, který nám zabraňuje zpracovat celá data najednou.

2.1.2 Učení bez učitele

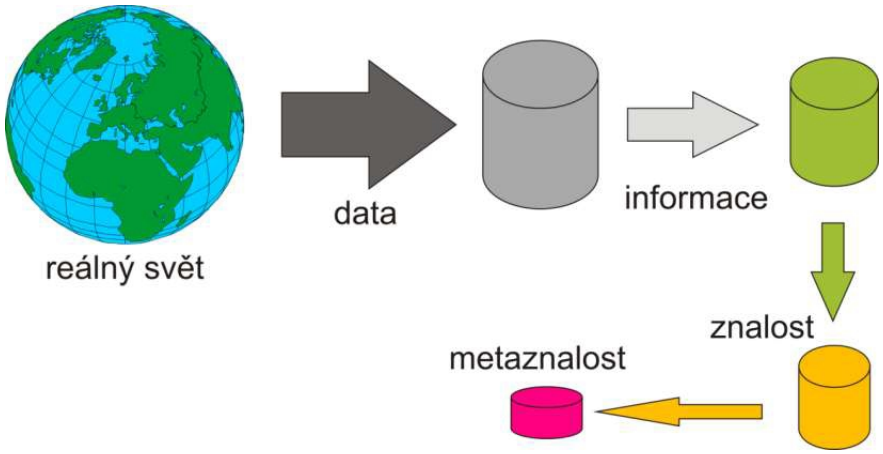
Je procesem učení se bez vzorů, protože uvedené případy nejsou označeny jako vstupní třída. Algoritmus se snaží nalézt skryté vzory. Zkoumány jsou vzorky, které jsou neohodnocené, či není zjištěna metrika na základě, které by se dalo potenciální řešení ohodnotit. Typická aplikace tohoto přístupu je v úlohách týkajících se shlukování. (Abu-Mostafa et al., 2012)

2.1.3 Semi-supervised učení

Je označováno tzv. kombinované učení, jenž je poměrně novou záležitostí. Je to kombinace předchozích dvou konceptů učení. Již z názvu je zřejmé, že se jedná o koncept, ve kterém trénování probíhá na ohodnocených i neohodnocených datech zároveň. Cílem je natrénovat klasifikátor, jak z případů ohodnocených, tak i neohodnocených. Díky této skutečnosti jsou poskytovány lepší výsledky než pro klasifikátor natrénovaný pouze na ohodnocených datech. Pokud máme dostatečné množství ohodnocených případů, nemá smysl aplikovat metody semi-supervised. Metoda poté neposkytne výrazně lepší výsledek než předchozí koncepty (Zhu a Goldberg, 2009).

2.2 Data mining

V následujících sekcích budou v textu zmíněny obecné pojmy týkající se dolování z dat. Následně se více zaměříme na techniky získávání znalostí z dat (*data mining*) a textu (*text mining*).



Obr. 1: Vyjádření vztahu mezi daty, informacemi a znalostmi pomocí hierarchického uspořádání, Zdroj: (Žižka, 2011)

Na uvedeném Obr. 1 je možné vidět, že základem hierarchie jsou data, která ve většině případů obsahují i nepotřebný a nežádoucí šum (irelevantní data). Naší snahou je vždy šum odstranit, abychom dostali pouze požadovaná data.

Za **data** považujeme údaje získávané libovolným pozorováním a měřením okolního reálného světa. Hodnoty údajů mohou být číselné (numerické) i nečíselné (nenumerické, tj. nominální, vyjmenované), jak uvádí (Žižka, 2011).

Samotná data jsou však stále neužitečná. Je třeba z nich získat či případně v nich najít informace tzn. vyfiltrovat irelevantní data (výběr dat zajímavých pro řešený problém). Z toho vyplývá definice **informace** jako části dat, která je relevantní k řešení nějakého stanoveného konkrétního problému.

Dle Shannovy definice: *„Informaci lze definovat jako míru množství neurčitosti, nejistoty o nějakém náhodném ději, odstraněnou realizaci tohoto děje.“* Pro měření množství informace se poté používají pojmy entropie a pravděpodobnost jevu.

Entropie představuje míru neurčitosti (neuspořádanosti) o daném jevu. Dle jiné definice je entropie vyjádřena jako střední hodnota informace pro všechny realizace jevu (Kullback, 2013).

Pro výpočet informace obsažené ve všech realizacích jevu lze použít vzorec entropie:

$$H(x) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i). \tag{1}$$

kde x_i značí výsledek i-té realizace jevu z množiny výsledků náhodného děje, $p(x_i)$ značí pravděpodobnost výsledku x_i, n značí počet realizací jevu, $H(x)$ značí množství informace obsažené ve všech realizacích jevu.

Dalším termínem je **znalost**. Znalostí se rozumí organizovaná informace využitelná k řešení problémů (Brožová a Houška, 2011). Znalost tedy lze chápat jako takovou informaci, kterou jednotlivec po jejím získání a osvojení dokáže začlenit do souvislosti s ostatními informacemi a dokáže na jejím základě jednat, rozhodovat.

Pro doplnění souvislostí **metaznalost** je jisté zpřesnění získané znalosti. Jedná se o tzv. „znalost o znalosti“ (Aguayo, 2010).

Data mining lze definovat jako proces získávání užitečných informací a znalostí ze shromážděných dat, které má podnik či organizace k dispozici a jsou určitým způsobem organizovány a uloženy. Postupy dolování dat jsou založeny na induktivním získávání znalostí zobecněním nějakých typických společných vlastností jednotlivých konkrétních vzorků dat, které vícerozměrně (formou vektorů) popisují entity reálného světa. Získaná znalost může sloužit např. ve formě pravidel, podle nichž se podnik bude řídit, bude jimi lákat zákazníky nabídkami pro ně atraktivními. Pravidla vlastně tvoří jednu z mnoha forem znalosti, která je pro člověka srozumitelná.

Většina problémů řešených metodami data miningu, by se dala rozdělit do několika hlavních skupin: klasifikace, predikce, shlukování, tvorba asociačních pravidel, popis a profilování). (Witten, Frank a Hall, 2011)

2.3 Text mining

Dle definice (Uldrich, 2011) je text mining metodou, která umí nestrukturovaná textová data zpracovat a poskytne nám stěžejní informaci obsaženou v textu dokumentu. Setřídí dokumenty podle podobnosti bez toho, aby je musel někdo číst. Celý tento automatizovaný proces bez potřeby lidských zdrojů je v současné době velmi ceněný a žádoucí.

Obor text miningu lze obecně zařadit pod soubor dataminingových metod, který vznikl jako další odvětví data miningu, pokrývající požadavky pro zpracování textů za souběžného vyhledávání informací v nich obsažených. Text mining je separován z důvodu skutečnosti, že data mining má obecnější záběr, vyhledává a zpracovává informace i v číselných, nominálních a ordinálních proměnných. Naopak text mining se specializuje na práci s nestrukturovaným textem.(Atkinson, 2000)

Jak uvádí (Sedláček, 2010) na dolování v textech může být nahlíženo jako na činnost skládající se ze dvou částí. První částí procesu je předzpracování, kdy je vstupní dokument převáděn do určité mezilehlé podoby, se kterou se provádí další zpracování. V této části procesu je zpravidla ze vstupního dokumentu extrahován pouze text, který je následně převeden na stejný druh písma (velikost, font, zvýraznění). Vynechány jsou jakékoli obrázky nebo elementy prezentující informace v jiné než textové podobě. Zachována může zůstat struktura textu, která při následné analýze může napomoci kvalitnějšímu určení významu daných termů (základních objektů, s nimiž se provádí další zpracování). Dále jsou z tohoto čistého textu vyjmuta tzv. stopslova. Jedná se o slova stanovená ruční analýzou jazyka a zařazená do nějakého slovníku. Tato slova nemají žádný podstatný význam (předložky, spojky, …). Pojem term nemá v dolování v textech pevně stanovený charakter. Jedná se o základní prvek, s nímž probíhá zpracování, ale jeho tvar se může lišit podle metody, která ho využívá, například věta při sumarizaci textu nebo jednotlivá slova (sousloví) při extrakci informací. Druhou částí je získávání znalostí, kdy jsou znalosti nebo vzory odvozovány z mezilehlé formy. V této části dochází podle účelu procesu k analýze vygenerovaných termů a k rozhodovacímu procesu vedoucímu k poskytnutí požadovaných výsledků – zařazení dokumentu do kategorie, poskytnutí abstraktu dokumentu, naplnění tabulky daty.

Obr. 2: Typický průběh dolování v textových datech, Zdroj: (Feldman a Sanger, 2007)

Dle daného schématu na Obr. 2 lze znázornit typický průběh pro dolování z textových dat. Schéma může být popsáno následující posloupností kroků:

- Zpracování začíná získáním kolekce dokumentů na vstupu. Vstup je nejdříve označen pojmy extrahovanými přímo z dokumentů.
- Následuje fáze předzpracování (*preprocessing*), kdy převádíme dokumenty do podoby vhodné pro zpracování. Jednotlivé metody předzpracování budou podrobněji popsány v následující sekci 2.6. Následně navazuje vygenerování potřebných atributů pro další činnost systému.
- Dále navazuje fáze procesu dolování v textech pomocí provedení operace vyhledávání vědomostí tzv. *knowledge discovery* mezi než lze zařadit např. klasifikační techniky či metody objevování shluků podobných dokumentů aj. Na závěr jsou výsledky zobrazeny pomocí vizualizačních nástrojů. Existuje velké množství profesionálních nástrojů (disponující dobře propracovaným vizuálním rozhraním jako jsou například Weka, SPSS Modeler).
- S procesem vizualizace úzce souvisí i následné zpracování výsledků z procesu dolování znalostí (*postprocessing*). Pro úlohy text miningu jsou využívané především operace prořezávání, generalizace, seskupování či potlačování šumu. Cílem je vždy získat zobecněnou informaci – znalost, o čemž je podrobněji pojednáno v (Feldman a Sanger, 2007).

2.4 Úlohy text miningu

Pro oblast text miningu se typicky využívá několik základních úloh, které lze rozdělit do jednotlivých kategorií. Každá z kategorií má své specifické využití.

2.4.1 Klasifikace dokumentů

Touto úlohou se podrobněji zabývá sekce 2.7. Co se týká klasifikace v textech, představuje klasifikace určování druhů textů (*text categorization*), jehož cílem je přiřazení dokumentů do jedné či více kategorií z předem daného výčtu.

2.4.2 Shlukování dokumentů (*Clustering*)

Shlukování je založeno na rozdělení textových dokumentů do předem daných kategorií tak, aby každá z nich obsahovala podobné dokumenty. Vytvořené shluky dokumentů mohou, ale nemusí odpovídat očekávaným kategoriím. Shlukování je zařazováno mezi deskriptivní úlohy. Využití nachází při klasifikaci případů (zatřídění do skupin) takovým způsobem, že si jsou jednotlivé instance patřící do stejné skupiny navzájem podobnější než instance z různých skupin. Shlukování ve své podstatě vychází z podobnosti (vzdálenosti) objektů za pomocí různých metrik např. Euklidovské vzdálenosti, Manhattanské metriky či Kosinovy podobnosti, (Weiss et al., 2010).

2.4.3 Analýza mínění (*Sentiment analysis*)

Analýza mínění je používána především pro data mining názoru široké veřejnosti na základě citově zabarvených a klíčových slov. Výhodou tohoto přístupu je, že tak může být učiněno i bez vědomí lidí, kteří se k dané věci vyjadřují. Dle (Koktan, 2012) lze pomocí analýzy sentimentu zjistit o autorovi mnohem více informací, než jen samotný postoj. Podle způsobu vyjadřování je možné odhadnout například věk, způsoby myšlení, dosažené vzdělaní a samozřejmě i pohlaví tvůrce článku.

Sentiment nalezený uvnitř komentářů, v odezvách nebo v nějaké kritice poskytuje užitečné indikátory pro mnoho účelů. Dokumenty mohou být z hlediska sentimentu klasifikovány buď do dvou kategorií (pozitivní nebo negativní) nebo také do n-kategorií (např. pro n = 5: velmi dobrá, dobrá, uspokojivá, špatná, velmi špatná). V tomto ohledu může být analýza sentimentu popisována jako úkol třídění, tedy zařadit daný dokument do předem zadaných kategorií sentimentu.

V době sociálních sítí není problém získat velké množství příspěvků obsahujících požadovaná klíčová slova. Zdrojem sentimentu se proto stávají především blogy nebo fóra vyjadřující něčí názor. Mezi další významné zdroje sentimentu lze zařadit recenze filmů a komentáře u jednotlivých filmů ve filmových databázích.

2.4.4 Vyhledávání informací (*Information Retrieval*)

Cílem této úlohy je na základě zadaného dotazu najít relevantní (dotazu nejlépe odpovídající) dokumenty. Dotazem mohou být jak jednotlivá slova, tak kompletní dokument. Tento postup je využit především ve vyhledávačích.

2.4.5 Extrakce informací (*Information Extraction*)

Cílem této úlohy je převést nestrukturovaný dokument do strukturované formy. To lze udělat nalezením a vyplněním určitých hodnot předem definovaných atributů. Dalším způsobem může být nalézt entity, které se v dokumentu vyskytují a jejich vlastnosti, vztahy mezi nimi nebo události, kterých se účastní, jak uvádí (Feldman a Sanger, 2007).

2.4.6 Síťová analýza (*Link Analysis*)

Tato úloha si dává za cíl převést nestrukturovaný dokument do strukturované podoby. Proces převodu lze buď realizovat nalezením a vyplněním určitých hodnot předem definovaných atributů. Dalším způsobem může být nalézt entity, které se v dokumentu vyskytují a jejich vlastnosti, vztahy mezi nimi či události, kterých se účastní (Feldman a Sanger, 2007).

2.4.7 Automatická sumarizace

Smyslem sumarizace je hledání uceleného popisu podmnožiny dat. Pomocí zmíněné úlohy dochází k redukci rozsáhlého dokumentu (souhrnu) při zachování jeho hlavního

významu. Základem jsou dvě metody. Extraktivní metody (*summary extraction*) vybírající z původního dokumentu slova, fráze a věty, z kterých vytvoří požadovaný souhrn. Abstraktní metody (*summary abstraction*) vytvářející vnitřní sémantickou reprezentaci a poté pomocí technik pro generování přirozeného jazyka vytvoří požadovaný souhrn (Feldman a Sanger, 2007).

2.5 Reprezentace textových dokumentů

Vhodná volba reprezentace nestrukturovaných textových dat je významným předpokladem k tomu, aby data mohla být podrobena metodám text miningu, v této práci reprezentovanými klasifikačními algoritmy. Před transformací textových dokumentů do strukturované podoby je vhodné si zvolit určitou elementární sémantickou stavební jednotku – *term*. Termy představují ve své podstatě jednotlivá slova. Alternativně bývají za termy zvoleny dvojice či n-tice (*n-gramy*), či určité fráze, ba dokonce celé věty.

Rozlišovány jsou hlavně dvě reprezentace, model bag-of-words a model vektorové reprezentace. Tyto dva přístupy budou podrobněji rozebrány v následujících odstavcích.

2.5.1 Model bag-of-words

Nejvíce užívanou transformací kolekce dokumentů je *bag-of-words*. Tato reprezentace umožňuje pohled na data jako na matici, kde řádky představují vektor dokumentů a sloupce zmiňované termy (slova z dokumentu), kde jejich pořadí nehraje roli. S tím souvisí nevýhoda tohoto přístupu. Při použití modelu bag-of-words není možné zpětně sestavit původní dokument a tím dochází ke ztrátě čitelnosti a sémantiky dokumentů.

Jedná se čistě o posloupnost termů, kde pořadí ani interpunkce nemají žádný význam. Dle (Srivastava et al., 2009) výzkumy posledních let přesto ukazují, že tato reprezentace přináší velice dobré výsledky a bývá často využívána v úlohách text miningu. Lze říci, že „bag of words“ je základním stavebním kamenem pro mnoho dalších metod.

Jakmile převedeme textový dokument do podoby množiny jednotlivých slov, můžeme již aplikovat metody strojového učení. Množina všech unikátních slov nalezených ve všech dokumentech, se označuje jako slovník (Paruchuri, 2013).

2.5.2 Model vektorového prostoru

Nejčastější způsob reprezentace textových dokumentů je na základě vektoru, který se vyznačuje právě tolika složkami, kolik je počet slov ve slovníku. Principiálně vychází z metody „bag of words“. Dokument je převeden do N-rozměrného prostoru, kde N je rovno počtu unikátních slov ze všech zkoumaných dokumentů, což uvádí (Zhang a Jin, 2010). Jednotlivé dokumenty jsou reprezentovány vektory, jež jsou zpravidla docela řídké (obsahují velké množství nulových polí).

Množina všech dokumentů je poté reprezentována v podobě matice. Dle (Paruchuri, 2013; Zhang a Jin, 2010) nejčastěji je pro uchování použita matice T-D:

T-D matice

Matice tohoto typu má na řádcích jednotlivé dokumenty ve vektorovém prostoru a sloupce reprezentují jednotlivé termy vyskytující se v dokumentech. Hodnoty mohou být vyjádřeny buď svojí přítomností (binární vyjádření) nebo frekvencí daných termů dokumentu.

V souvislosti s termy je vhodné taktéž zmínit určování váhy termu, jenž indukuje jejich důležitost. K určování váhy termu můžeme přistupovat z různých pohledů. Touto problematikou se zabývá následující kapitola 2.5.3.

2.5.3 Výpočet váhy termů

Jak bylo zmíněno výše, termům mohou být přiřazeny statistické váhy. Úkolem vážení frekvence výskytu slov v dokumentu je lepší vyjádření jejich důležitosti. Pokud se vyskytuje nějaký term vícekrát v dokumentu, je jeho důležitost bezesporu vyšší než v případě jednoho výskytu. Avšak důležitost slova neroste lineárně s frekvencí jeho výskytu. Pro vyjádření této skutečnosti se ve vektorové reprezentaci používají lokální a globální váhy, jak uvádí (Weiss et al., 2010). Nejhojněji využívané jsou především dva přístupy.

- **Přítomnost termu** (TP, *Term presence*)
  Přítomnost termu představuje jednoduchý způsob reprezentace textových dokumentů. Hodnoty vektoru nabývají binární hodnoty (hodnoty 0 v případě nepřítomnosti termu ve slovníku dokumentu a hodnoty 1 v případě, že se term v dokumentu vyskytuje alespoň jednou).
- **Frekvence termu** (TF, *Term frequency*)
  Toto je varianta představující rozšířený způsob vektorové reprezentace textových dokumentů, konkrétně pomocí počtu výskytů termů. Vektory se skládají z hodnot reprezentovaných celými přirozenými čísly vyjadřujícími frekvence výskytů termu v dokumentu. Tato lokální váha se často používá ve spojení s globální váhou IDF, která je zmiňována dále. Pro delší dokumenty (s vyšším počtem termů) může dojít k nadhodnocení termů. Tomu lze zabránit výpočtem relativní důležitosti i-tého slova v j-tém dokumentu.

$$TF_{ij} = \frac{n_{ij}}{\sum_i n_{ij}} \tag{2}$$

kde n_{ij} je frekvence výskytu i-tého termu v j-tém dokumentu, $\sum_i n_{ij}$ je počet výskytů všech termů v j-tém dokumentu.

Pokud se rozhodneme identifikovat celkové váhy termu v dokumentu, je lokální vážení frekvence jeho výskytu v dokumentu často nedostačující. V tomto případě

je vhodné brát v potaz i globální váhu, kdy se bere v úvahu významnost termu v rámci všech dokumentů ve vektorovém prostoru. Způsobů jak určit globální váhu existuje opět několik. Nejpoužívanější je způsob inverzní frekvence výskytu termu (IDF, *inverse document frequency*)

Inverzní frekvence výskytu termu je založena na myšlence, že čím větší množství dokumentů obsahuje nějaký term (slovo), tím nižší by měla být jeho důležitost. Jinými slovy čím nižší je frekvence výskytu daného termu ve všech dokumentech, tím více je daný term specifický a tudíž důležitější. Důležitost termu by tedy měla být redukována hodnotou, která roste s nižším výskytem termu v celé kolekci dokumentů, čehož lze dosáhnout použitím převrácené hodnoty četnosti termu ve všech dokumentech.

$$IDF(t_i) = \frac{\log N}{n(t_i)} \tag{3}$$

kde t_i představuje i-tý term, N je počet všech dokumentů,$n(t_i)$ je počet dokumentů, které obsahují term t_i.

Při určování vah v textových dokumentech je však nejčastěji používaná kombinace lokální váhy frekvence výskytu termu a globální váhy inverzní frekvence výskytu termu, tedy váha TF-IDF.

- **TF-IDF** (*term frequency-inverse document frequency*)
  Váha TF-IDF je dána součinem složky TF a složky IDF:

$$TF - IDF(t_i) = TF \times IDF = \frac{n_{ij}}{\sum_i n_{ij}} \times \frac{\log N}{n(t_i)} \tag{4}$$

2.6 Předzpracování textových dat

S kvalitou získané znalosti úzce souvisí předzpracování dat. V návaznosti na diplomovou práci (Kotiková, 2014), kde jsou přehledně popsány jednotlivé metody předzpracování zde ve stručnosti bude zmíněno o daných metodách. Tím že data předzpracujeme, můžeme získat kvalitnější a hodnotnější vytěženou znalost.

Jak uvádí (Han a Micheline, 2011) lze zobecnit jednotlivé metody na metody zajišťující:

- Očištění dat (*Data cleaning*), jenž odstraňují šum a redukují nekonzistentnost v datech.
- Datová integrace (*Data integration*), snahou je sjednocení dat z různých zdrojů do jednotné formy.
- Datová redukce (*Data reduction*), tím se rozumí snížení (redukce) datového slovníku za pomocí eliminace redundantních vzorů v datech či redukce velikosti dat.

- Datová normalizace (*Data normalization*), stanovení určitého rozmezí, kterého mohou data nabývat. Toto vše se týká obecně data miningu. Pro text mining je třeba použít konkrétnější metody.

2.6.1 Stemming

Je metodou převodu textových dat do základního tvaru. Tento proces je nazýván stemming. Konkrétně se jedná o transformaci prostřednictvím odstranění morfologických koncovek, předpon a přípon, případně odvozené tvary slov nahradit tvarem jedním nazvaným „stem“ (kmen). Princip stemmingové metody je založený na aplikaci různých přepisovacích pravidel (Miner, 2012). Přepisovací pravidlo představuje nahrazení určité sekvence sekvencí jinou. Stemming oproti lemmatizaci pracuje s jednotlivými slovy a v případě shody slova s levou stranou přepisovacího pravidla se dané pravidlo aplikuje. V návaznosti na tuto skutečnost se projevuje nevýhoda tohoto přístupu. V některých případech může docházet k nesprávné transformaci a tím ke vzniku nesprávných (nekorektních) slov. Nejčastěji je aplikovaná stemming metoda na základě Porterova algoritmu. Porterův algoritmus používá pro transformaci slov na jejich kořenovou formu řadu heuristických nahrazovacích pravidel. Pozitivem tohoto přístupu je redukce počtu odlišných slov v dokumentu a tím zvýšení frekvence výskytu jednotlivých slov. Tento fakt vede k efektivnějšímu a především rychlejšímu zpracování textových dokumentů. Na druhou stranu se zde vyskytuje nevýhoda v podobě ztráty informace obsažené v původním tvaru upraveného slova (Weiss et al., 2010).

2.6.2 Lemmatizace

Principiálně lemmatizace souvisí z předchozí metodou. Lemmatizace je převod slova na jeho kmen. Tento proces je nasměrován k podobnému cíli jako stemming, avšak používá jiné prostředky. Základem je lematizér, jenž provádí převod slov vzniklých odvozováním, skloňováním či časováním do jejich základního tvaru (*lemma*) a to s použitím morfologické analýzy. Pomocí ní lze tento základní tvar slova určit. Implementace kvalitního lematizéru je poměrně náročná činnost. Častěji se pro převádění slov do základního tvaru používá spíše jednodušší přístup – stemming.

2.6.3 Odstranění stopslov

Stopslova představují slova, která se vyskytují v textu velice často, ale nemají významovou hodnotu. Pro český jazyk se nejčastěji se jedná o různé předložky, spojky, zájmena. Pro zahraniční jazyky jako je např. anglický jazyk lze zařadit navíc neurčité členy (a, an, the). Často se lze setkat s označením *běžná* slova. Jedná se tedy o irelevantní slova vzhledem k dolování znalostí. Jak dokazuje (Krupník, 2014) hlavním problémem je identifikace těchto slov. Lze rozlišit dvě kategorie stop slov. První zmiňovaná kategorie jsou **obecná** stop slova. Tato stop slova lze veřejně dohledat pro všechny zkoumané oblasti. Druhou kategorií jsou tzv. **doménově orientovaná** stop

slova. Doménově orientovaná stop slova musíme identifikovat samy. Pro identifikaci existuje však celá řada statistických metod, které napomáhají při určení důležitosti jednotlivých slov v textu (např. *F-measure*, *Chí Statistic*, *Odds ratio* a jiné).

2.6.4 Tokenizace

Tokenizací rozumíme rozdělení souvislého textu do smysluplných částí. Textový dokument lze rozdělit na diskrétní části (slova), tím se rozumí jednotlivé instance prvků slovníku (*tokeny*). Samotný proces se zdá být poměrně jednoduchý. Mohou však nastat situace, kdy řešení není zcela triviální. Je nutné dbát na specifika přirozených jazyků: diakritika, slovní spojení, interpunkce, identifikace oddělovačů či zkrácené tvary a jiné (Feldman a Sanger, 2007).

2.6.5 Case folding

Case folding představuje proces normalizace textů do uniformní podoby – slova v dokumentu jsou převedena buď na minusky (malá písmena) či verzálky (velká písmena) a tím je dosaženo jednotné formy. Jak uvádí ve své práci (Novák, 2013) tímto způsobem může však dojít i ke sjednocení způsobu zápisu slov, která vyjadřují různé věci a měla by tedy zůstat rozlišitelná. Příkladem jsou jména osobností či zkratky. Jak je vidět na následujícím příkladu nejedná se o zcela bezchybnou metodu předzpracování.

```
Jack Black (jméno herce) ---> jack black (překlad: hever černý)
F.E.D. (jméno úřadu) ---> fed (překlad: nakrmený)
```

2.6.6 Kontrola pravopisu

S problematikou textových dat je úzce spojena nutná kontrola pravopisu. V textových datech se velmi často objevuje velké množství překlepů či gramatických chyb. Kontrola pravopisu (*spell check*) má své opodstatnění. Díky překlepům narůstá datový slovník a vznikají nadbytečná slova, která nemají pro získání znalostí význam. Vždy je nejprve nutné chybu nalézt a následně ji opravit. Většina nástrojů pro kontrolu pravopisu je založena na algoritmu používajícím korpus slov se správným pravopisem, z něhož se vybírají alternativy za chybná slova.

Při kontrole pravopisu je vždy nutné nejprve chybu nalézt a následně ji opravit. Existuje velká řada nástrojů pro korekci slov v různých jazycích. Je možné je dělit dle jejich funkcionality při korekci slova, a to na skupinu nástrojů, které vyžadují interakci uživatele při opravování v podobě manuálního výběru nejvhodnější alternativy chybného slova a na druhé straně existují nástroje provádějící korekci zcela automatizovaně. První zmiňovaný způsob je pro velké množství dat zcela nevhodný. Oproti tomu je druhý způsob vhodnější pro obrovské množství dat. Ve většině případů algoritmy kontroly a korekce pravopisu používají korpus slov se správným pravopisem, z něhož se vybírají alternativy za chybná slova (Wu, Xiong a Shekhar, 2004).

2.6.7 N-tice slov

Na rozdíl od předchozích metod, které se snaží redukovat počet slov ve slovníku. Existují případy, kdy je naopak prospěšné rozšířit slovník o nové fráze tzv. n-tice slov, které mohou hrát významnější roli než slova samotná. Často dochází k záměně pojmosloví, proto považuji za důležité vymezit rozdíl mezi **n-gramem** a **n-ticí**. N-gramy představují hledaná slova, která jsou bezprostředně u sebe a n-tice jsou obecně vybraná libovolná slova věty či celého dokumentu.

2.6.8 Odstranění slov mající nízkou frekvencí výskytu

Jak uvádí ve své práci (Žižka a Dařena, 2011) slova s nízkou frekvencí zastoupení jsou v dokumentu charakterizována pouze velice malým vlivem na zpracování a výsledek dolování z textových dat obecně. Odstraněním slov, která mají v dokumentech nízké zastoupení výskytu, dokážeme poměrně jednoduchým způsobem snížit velikost slovníku slov téměř o polovinu a to, aniž by to negativně ovlivnilo výkonnost klasifikátoru.

2.7 Klasifikace

Cílem klasifikace je přiřazení správné vlastnosti (tj. třídy) pro danou instanci (vzorek). Pro trénovací data jsou jejich vlastnosti známy, takže jsou známy odpovědi na možné dotazy – kvalitní dotazy mají odpovědi, které pro trénovací data vyvolají co nejmenší „překvapení“. Mírou „překvapení“ je zde entropie, která ji měří pomocí pravděpodobnosti.

S růstem pravděpodobnosti výskytu jevu klesá „překvapení“ nad jeho výskytem, tj. je-li pravděpodobnost `p = 1.0` pak lze daný jev očekávat zcela jistě. Podobně je tomu naopak pro `p = 0.0`, kdy nevzniká překvapení (Sebastiani, 2002).

Překvapení (*surprise*) dvou nezávislých jevů s pravděpodobnostmi p a q lze vyjádřit rovnicí:

$$S(pq) = S(p) + S(q) \tag{5}$$

Odtud dle Shannonovy definice odvodit vztah a samotný vzorec pro výpočet entropie:

$$S(p) = -\log_b p, \tag{6}$$

kde b vyjadřuje míru informace v jednotkách bit.

$$H(x) = -\sum_n p_n S(p_n) = -\sum_n p_n \log_2(p_n). \tag{7}$$

Klasifikace textu spočívá v přiřazení booleovské hodnoty každému páru $\langle d_j, c_i\rangle \epsilon\ D \times C$, kde D představuje doménu dokumentů a $C = c_1, c_2, \ldots, c|C$ je množinou předdefinovaných kategorií. Hodnota T (pravda) přiřazená dvojici $\langle d_j, c_i\rangle$ označuje rozhodnutí zařadit dokument d_j do kategorie c_i, zatímco hodnota F (nepravda) znamená rozhodnutí nezařadit dokument d_j do kategorie c_i. Úlohou je tedy

aproximovat neznámou cílovou funkci $\Psi : D \times C \rightarrow \{T, F\}$ (která popisuje, jak jednotlivé dokumenty mají být klasifikovány) funkcí $\Phi : D \times C \rightarrow \{T, F\}$ nazývanou klasifikátor (hypotéza, model) tak, aby funkce Ψ a Φ byly v co největším „souladu“. Předpokládáme, že kategorie jsou pouze symbolická označení bez dodatečných informací o jejich významu a nejsou dostupné žádné vnější znalosti o datech (například zdroj, datum vydání).

Klasifikace tak musí být dosažena zcela obecně, výhradně na základě vnitřních znalostí získaných přímo z analyzovaných textových dokumentů. Dle využití může být výstupem klasifikace dokumentu buď jedna třída (*single-label text classification*), kdy každý dokument je klasifikován právě do jedné kategorie. Na druhou stranu existuje rozdělení do více tříd (*multilabel text classification*), u nichž každý dokument musí náležet buď do kategorie c_i, nebo do jejího doplňku $\bar{c_i}$. (Sebastiani, 2002)

2.7.1 Trénink a testování

Při odhadování chyby klasifikátoru na budoucích datech, která nebyla k dispozici během trénování, je nutné klasifikátor otestovat. K testování lze použít vyčleněná data, která se tréninku nezúčastní (testovací data), či se dá využít části trénovacích dat tím, že se použijí jako testovací, což vede ke snížení počtu trénovacích příkladů.

Lze použít i metodu krosvalidace (*cross-validation*). Tato metoda je založena na náhodném výběru dat na pokud možno stejně velké částí – k podmnožin (k může záviset na celkovém množství trénovacích instancí). Standardně se používá na 10 podmnožin. Poté proběhne 10 tréninků tak, že v každém z nich se použije 9 podmnožin jako trénovacích a 1 jako testovací. Každý trénink používá pro testování různou z podmnožin, takže všechna data jsou postupně využita na trénování a testování. Každé testování zjistí chybu klasifikátoru a výsledná očekávaná chyba je průměrem. Tato chyba je tzv. *pesimistická*, protože byla odhadnuta na menším počtu trénovacích příkladů (např. 90 %). Výsledný klasifikátor se následně natrénuje pomocí všech příkladů.

V (Bifet et al., 2011) se lze seznámit s alternativou, jak zacházet s trénovacími daty jako s proudem dat – potenciálně nekonečným tokem dat, který přichází v pořadí, které nelze kontrolovat. Klasifikační algoritmus by měl pro tento koncept však splňovat několik požadavků, které jsou vhodné k učení se z datového proudu. Požadavky podrobněji popíšeme níže:

1. **Zpracovat příklad najednou a kontrolovat (maximálně) jen jednou**
   Mezi hlavní charakteristiku datového toku patří především to, že data „tečou“ v proudu dat po příkladech. Každý případ je akceptován v pořadí, jak přijde v toku dat. Jakmile je příklad prozkoumán či ignorován, je vyřazen a znovu se již neuvažuje. Ačkoliv je tento požadavek aplikován na vstupu do algoritmu, neexistuje žádné pravidlo algoritmu pamatovat si příklady interně v krátké době.

2. **Použití omezeného množství paměti**
   Hlavní motivací pro použití modelu datového toku je především to, že umožňuje zpracování dat, která jsou mnohonásobně větší než dostupná pracovní paměť. Potýkáme se zde s nebezpečím, že při zpracování tak velkého množství dat je možné, že se paměť snadno vyčerpá, pokud není záměrně omezeno nastavení na její použití. Paměť používaná algoritmem může být rozdělena do dvou kategorií: paměť používaná pro ukládání provozních statistik a paměť sloužící k ukládání aktuálního modelu. Efektivní algoritmy nerozlišují mezi těmito pamětmi. Statistika chodu přímo vytváří model použitý pro predikci.

3. **Práce v omezeném množství času**
   Algoritmus by měl být schopen pracovat v reálném čase. Měl by zpracovat příklady pokud možno co nejrychleji. Absolutní čas není tak kritická veličina v méně náročných aplikacích jako je klasifikace pro velké, ale trvalé zdroje dat. Nicméně čím pomalejší algoritmus máme, tím nižší hodnotu dokáže přinést pro uživatele, kteří vyžadují převážně výsledky v rozumném čase.

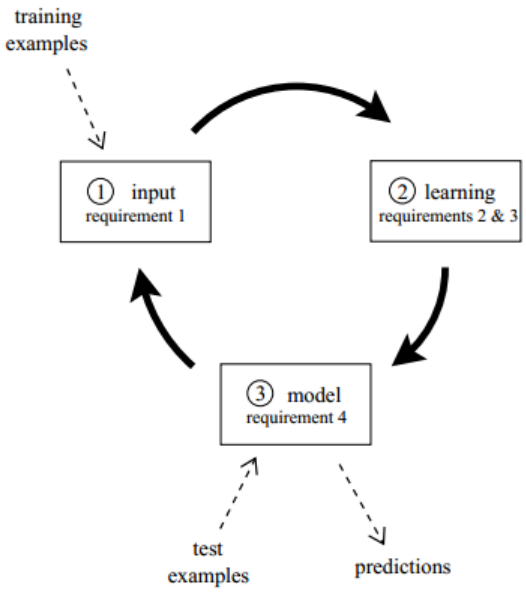
4. **Být připraven předvídat v každém bodě**
   Ideální algoritmus by měl být schopen vytvářet co nejlepší možný model z dat, která byla pozorována poté, co jsme viděli libovolný počet příkladů. Proces generování modelu by měl být tak efektivní, jak je to možné. V nejlepším případě tak, že není nutný žádný překlad navíc. Tímto rozumíme, že s konečným modelem je přímo manipulováno v paměti dle algoritmu, který zpracuje příklady, spíše než přepočítá model založený na běžících statistikách.

Pro data v proudu lze klasifikaci dat popsat následujícím Obr. 3, jenž popisuje typické použití algoritmu klasifikace toku dat a jaké požadavky vcházejí dovnitř je nutné splňovat. V obecném modelu klasifikace toku dat následují po sobě tyto tři hlavní kroky v opakujícím se cyklu:

1. Algoritmus předává dostupné příklady z toku dat (požadavek 1).

2. Algoritmus zpracuje příklad, provede aktualizaci svých datových struktur. Učiní tak bez překročení meze paměti nastavené na něm (požadavek 2) a při takové rychlosti, jak jen to je možné (požadavek 3).

3. V posledním kroku je algoritmus připraven přijmout další příklad. Na žádost je schopen poskytnout model, který může být použit pro predikci třídy neznámých příkladů (požadavek 4).



Obr. 3: Schéma klasifikace toku dat, Zdroj: (Bifet et al., 2011)

2.7.2 Měření kvality klasifikátoru

Pro klasifikaci je důležitá výkonnost daného klasifikátoru. Je vhodné vzít v úvahu míru chybovosti (*error rate*). Klasifikátor totiž predikuje třídu pro každou instanci. Pro případ, kdy je instance správná, je započítána jako úspěšná. Může nastat i opačný případ a instanci zařadíme jako chybovou.

Matice záměn

Vyhodnocování úspěšnosti klasifikátoru obvykle probíhá na základě provedení experimentu a z něho získaných údajů o správně či nesprávně klasifikovaných případech. Tyto výsledky experimentování lze zobrazit pomocí tzv. matice záměn (*konfúzní matice*), jenž je schematicky uvedena prostřednictvím Tab. 1.

Tab. 1: Matice záměn pro dvě třídy

	klasifikovány jako pozitivní	klasifikovány jako negativní
skutečně pozitivní	TP (true positive)	FN (false negative)
skutečně negativní	FP (false positive)	TN (true negative)

Při klasifikování do dvou tříd je možné tyto třídy označit jako *pozitivní* a *negativní*. Pozitivní, resp. negativní instance, které jsou klasifikovány správně, lze označit jako správně pozitivní (*true positive*, TP), resp. falešně negativní (*true negative*, TN). Nesprávně klasifikované instance se nazývají falešně pozitivní (*false positive*, FP) a falešně negativní (*flase negative*, FN). Na základě těchto hodnot je umožněno definovat řadu metrik pro hodnocení úspěšnosti klasifikátoru:

Správnost *(Accuracy)*

Správnost je celkově nejvíce užívaná metrika, se kterou se lze setkat, když děláme klasifikaci. Je to v podstatě údaj o tom, jak blízko je algoritmus ke skutečným hodnotám klasifikace. Výpočet dle následujícího vzorce:

$$accuracy = \frac{TP + TN}{TP + FT + TN + FN} \tag{8}$$

Míra úplnosti *(Recall)*

Míra úplnosti představuje metriku, která objasňuje do jaké míry všechny příklady, které měly být pozitivně klasifikovány, ve skutečnosti takto klasifikovány jsou.

$$recall = \frac{TP}{TP + FN} \tag{9}$$

Přesnost *(Precision)*

Přesnost udává ve své podstatě procento příkladů, které jsou klasifikovány jako pozitivní a jsou skutečně pozitivní.

$$precision = \frac{TP}{TP + FP} \tag{10}$$

F1-score (*F-measure*)

Poslední zmíněná metrika F1-score kombinuje hodnoty přesnosti a míry úplnosti. F1-score vyjadřuje vážený harmonický průměr přesnosti a míry úplnosti. Je to tedy jedna z dalších metrik, kterou lze pro přesnost klasifikace určovat.

$$F1score = \frac{2 \times precision \times recall}{precision + recall} \tag{11}$$

V následující části práce se zaměříme na jednotlivé metody klasifikace, které budou využity v této práci.

2.8 Rozhodovací stromy

Rozhodovací stromy (*Decision trees*) lze zařadit mezi jedny z klasifikačních algoritmů. Své využití nachází v případech řešení data mining problémů, které jsou založeny na předpovědi. Každý vnitřní uzel rozhodovacího stromu obsahuje test atributu. Každá větev s uzly odpovídá možnému výsledku testu. Každý list obsahuje predikci třídy.

Rozhodovací stromy se obecně učí dle metody shora-dolů (*TOP-DOWN*) dle rekurzivního nahrazení testovacími uzly. Vždy začínáme u kořene. Atribut pro testování je vybrán na základě porovnání všech dostupných atributů. Vybrán je vždy ten nejlepší podle určitého heuristického měřítka. Klasické rozhodovací stromy (C4.5,

ID3 (*Iterative Dichotomiser* 3), CHAID (*Chi-squared Automatic Interaction Detector*)) jsou založeny na předpokladu pro učení říkající, že všechny trénovací příklady jsou uloženy současně v hlavní paměti. Nastává zde silné omezení počtu příkladů, ze kterých se lze učit. Klasické rozhodovací stromy tak nejsou použitelné pro datové proudy, pro případy kdy dochází k potenciálu minimálního omezení pro počet příkladů, které přichází postupně. Na druhé straně pak vznikají nové algoritmy, mezi něž lze zařadit Hoeffdingův rozhodovací strom, které využívají pro své zpracování technologii datových toků. Tyto algoritmy pak dokáží zpracovat větší objemy dat, v kratším časovém intervalu. (Bifet a Gavaldà, 2009)

2.8.1 J48

Algoritmus J48 je open source implementací algoritmu C4.5 v prostředí Weka využívaného v této práci. Princip algoritmu J48 je založen na výběru atributů (tzv. termů – v případě analýzy textových dokumentů) na základě jejich schopnosti snížit entropii. Algoritmus se snaží rozdělit původní různorodou (heterogenní) množinu objektů na homogennější podmnožiny, které mají nižší entropii (míru neuspořádanosti) než množina původní. Množina obsahující prvky výhradně jedné třídy je dokonale homogenní, tj. má nulovou entropii. Hodnotu entropie lze vypočítat podle vzorce (1).

U rozhodovacích stromů tohoto typu dochází též často k případu, že listy stromu obsahují pouze jeden prvek – tento stav je znám jako přeučení (*overfitting*). Výsledný strom po přeučení je velmi košatý a nesrozumitelný a postrádá obecnost. Lze však získat redukovaný strom prořezáním (*pruning*). U algoritmu J48 s počtem příkladů lineárně roste horní hranice výpočetní složitosti. Toto vše má značný vliv na výpočetní složitost, která je v závislosti na počtu příkladů polynomická. (Žižka a Svoboda, 2015; Bhargava et al., 2013).

2.8.2 CVFDT

Z důvodu efektivity zpracování větších objemů dat je vhodné použití efektivnějších algoritmů. Mezi jedny z efektivních algoritmu patři algoritmy typu CVFDT (*Concept-adapting Very Fast Decision Tree learner*), mezi tento typ algoritmů lze zařadit i Hoeffdingův strom, o němž bude zmíněno dále. Nejprve si musíme popsat základní princip CVFDT. Tyto algoritmy vynikají především způsobem učení. Dokáží se během procesu učení postupně doučovat tzv. „za běhu“. Způsob učení je založen na získání znalosti o novém pojmu a zahození předchozí znalosti. Koncept se zdá být neefektivní, ale je dosaženo rychlejšího zpracování oproti standardním algoritmům.

CVFDT udržuje model, jenž se učí v synchronizaci s měnícími se koncepty. Tímto dochází k průběžnému sledování kvality předchozích nalezených rozhodnutí vzhledem k posuvnému oknu dat z datového toku a jejich aktualizaci „*fine-grained way*“ – jemnozrnným způsobem. Případně když je zjištěno, že rozdělením se údaje změní (Khan et al., 2014).

Dostatečný objem statistických údajů se udržuje v průběhu času pro každý případ M posuzovaného při každém kroku vyhledávání. Po prvních w příkladech, kde w představuje šířku okna, se odečte nejstarší příklad z těchto statistik poté, co je přidán nový příklad. Po každých Δ N nových příkladech, se zase určuje nejlepší kandidát dle předchozího rozhodovacího hledání případu. Pokud jeden z nich je lepší než starý případ $\Delta*$, nastává jeden ze dvou případů. Buď je původní rozhodnutí nesprávné (což se stane jen ve zlomku času Δ), či nastane koncept posunu dále. V obou případech začne alternativní hledání od nových případů, přičemž nadále je usilováno o původní vyhledávání. Pravidelně se využívá řada nových příkladů jako potvrzení o nastavení v porovnání s výkonností modelů vyráběných novým a starým vyhledáváním. Následně dochází k prořezání starého vyhledávání (nahrazujeme jej novým), když je nový model v průměru lepší než starý. Vždy je dodržováno pravidlo, že nová hledání jsou prořezávána, pokud po maximálním počtu validací daný model selže a v průměru je dané hledání přesnější než starší hledání. Proběhne-li více než maximální počet nových vyhledávání, dochází k prořezání těch nejméně výkonných hledání (Hulten et al., 2001; Bifet et al., 2011).

2.8.3 Hoeffdingův strom

Hoeffdingův strom (*Hoeffding Tree*) je algoritmus vyvolávající rozhodovací strom z toku dat. Algoritmus je založen na krátkých kontrolách každého příkladu v proudu právě jednou, aniž by daný příklad musel být ukládán poté, co byl použit pro aktualizaci stromu. Jediná informace ukládaná v paměti je strom samotný, jenž ukládá dostatečné informace ve svých listech tak, aby mohlo být přistoupeno k růstu a mohl být použit k vytvoření předpovědí na jakémkoliv místě v čase během procesu trénování příkladů. Jedná se o rozhodovací strom založený na proudění (*streamování*) dat. Hoeffdingův strom (HT) je indukčním algoritmem implementujícím rozhodovací strom, který je schopný se učit z masivního datového toku. Hoeffdingův strom je vhodný k využití toho, že postačuje malý vzorek dat proto, abychom mohli zvolit optimální tzv. *splitting* neboli dělící atribut. Myšlenka algoritmu je podporována Hoeffdingovskou hranicí (*Hoeffding bound*), která vypočítává počet pozorování potřebných pro odhad statistik s předepsanou přesností. Přesněji daný problém popisuje následující vzorec (12). Hoeffdingova hranice uvádí, že s pravděpodobnosti $1-\delta$ se skutečný průměr náhodné proměnné o rozsahu R nebude lišit od odhadovaného průměru po n nezávislých pozorováních o více než ϵ. (Hoeglinger a Pears, 2007)

$$\epsilon = \sqrt{\frac{R^2 \ln(1/delta)}{2n}} \tag{12}$$

kde R je rozsah skutečné hodnoty náhodné proměnné r, n je počet nezávislých pozorování, $1-\delta$ je chyba pravděpodobnosti, ϵ je požadovaná tolerance chyby.

Hoeffdingův strom se vyznačuje velkým množstvím parametrů, které je možné nastavit. Podrobněji si tedy zde jednotlivé parametry rozebereme:

-L: Použitý klasifikátor pro listy

Lze zvolit z trojice voleb: *Majority class*, *Naive Bayes*, *Naive Bayes Adaptive*.

- Majority class (Většinová třída): nejvíce frekventované označení v listech.
- Naivní Bayes: Naivní Bayes umožňuje získat extra informace a tím dosáhnout přesnější predikce. Technika výpočtu je založena na určení přesnosti pro každý list stromu viz (Carela-Español et al., 2015).
- Adaptivní Naivní Bayes: kombinuje výhody většinové třídy a Naivního Bayese. Díky kombinování výpočtu *error rate* pomocí většinové třídy a použití Naivního Bayese pro listy lze predikovat případy s vyšší správností a především rychleji.

-S: Použité rozdělovací kritérium (*splitCriterion*)

Použít lze z dvojice obměn (0 = Gini, 1 = Info gain).

-E: Přípustná chyba pro rozdělovací rozhodnutí (*splitConfidence*)

Přípustná chyba je nastavována v exponenciálním tvaru čísla. Pro hodnoty blížící se 0 bude trvat rozhodnutí delší čas.

-H: Práh rozdělení (*TieThreshold*)

Práh rozdělení, kdy dochází k porušení vazeb. Často se totiž stává, že máme v listech dva či více atributů, které nemůžeme rozdělit, protože mají identické hodnoty. Tyto atributy se poté stávají ideálním rozdělovacím uzlem. Rozhodovací kritérium o rozdělení je odsunuto až do doby, než nabude rozdílné hodnoty a než dojde ke snížení přesnosti.

-M: Minimální podíl váhy (*minimumFractionOfWeightInfoGain*)

Tento parametr určuje minimální podíl váhy, jenž je zapotřebí pro rozdělení do dvou větví při použití *InfoGainSplitting*.

-G: Počet instancí listu, které by měly být dodržovány mezi rozdělovacími pokusy (*Grace period*)

Jedná se o odbornou frázi. Tato dvě slova dohromady mají zcela specifický význam, který se česky dá napsat jako „doba odkladu“. Doba odkladu má význam v tom, aby se s příchodem každé nové špatně klasifikované instance nemusel stále přetrénovat daný algoritmus. Připouští se určitá „nefunkčnost“. Vždy se počítá kolikrát došlo k selhání. Když je překročen povolený počet selhání, natrénuje se algoritmus znovu. Tímto se sledují např. změny během času, ale zároveň se zabraňuje přílišné „citlivosti“ na nečetné (řídké, náhodné) změny.

-N: Počet instancí listu

Parametr měřící počet instancí listu, jenž by měl být dodržen předtím než je provedena předpověď.

Podrobný pseudokód pro Hoeffdingův strom je vyjádřen na následujícím schématu Obr. 4.

Algorithm 2 Hoeffding tree induction algorithm.

1: Let HT be a tree with a single leaf (the root)
2: **for all** training examples **do**
3: Sort example into leaf l using HT
4: Update sufficient statistics in l
5: Increment n_l, the number of examples seen at l
6: **if** $n_l \ mod \ n_{min} = 0$ **and** examples seen at l not all of same class **then**
7: Compute $\overline{G}_l(X_i)$ for each attribute
8: Let X_a be attribute with highest $\overline{G}_l$
9: Let X_b be attribute with second-highest $\overline{G}_l$
10: Compute Hoeffding bound $\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n_l}}$
11: **if** $X_a \neq X_\emptyset$ **and** $(\overline{G}_l(X_a) - \overline{G}_l(X_b) > \epsilon$ **or** $\epsilon < \tau)$ **then**
12: Replace l with an internal node that splits on X_a
13: **for all** branches of the split **do**
14: Add a new leaf with initialized sufficient statistics
15: **end for**
16: **end if**
17: **end if**
18: **end for**

Obr. 4: Pseudokód algoritmu pro Hoeffdingův strom, Zdroj: (Bifet et al., 2010)

Řádek 1: Inicializace stromové struktury začínající od jediného kořenového uzlu.

Řádky 2–18: Představují cyklus, který je prováděn pro každý trénovací případ.

Řádek 3: Každý případ je filtrován ze stromu na příslušný list v závislosti na testech přítomných v listech stromu.

Řádek 4: Provedení aktualizace každého listu stromu, kde dosaženo postačující statistiky proto, aby mohl být odhadnut informační zisk rozdělení pro každý atribut.

Řádek 5: Poukazuje na n_l počet příkladů v listu, kde hodnota $_l$ se aktualizuje.

Řádek 6: Vyhodnocuje podmínku $n_l \ mod \ n_{min} = 0$ a zároveň příklady v listu l nejsou ze stejné třídy.

Řádek 7: Vypočítána hodnota $G_l(X_i)$, což je funkce kritéria rozdělení (*GracePeriod*) pro každý příklad. Jedná se o odhadovanou hodnotu.

Řádek 8–11: Hodnota X_a přičtena k nejvyšší hodnotě G_l a hodnota X_b přičtena k druhé nejvyšší hodnotě G_l, následně je spočítána Hoeffdingova hranice dle vzorce (12).

Řádek 11: Vyhodnocení testů pro hodnotu X_0 (nulový atribut), test pro *tie-breaking* τ (rozdělovací kritérium). Následně je daný atribut vybrán jako nejlepší volba.

Řádky 12–15: Rozřazení uzlů a tím růst stromové struktury.

2.8.4 Hoeffdingův alternativní strom

Jak je uvedeno v (Holmes et al., 2007), Hoeffdingův alternativní strom (*Hoeffding Option Tree*), je vylepšenou verzí základní verze Hoeffdingova stromu s možností volby rozhodovacího uzlu použitého pro proudění („streaming“) dat. Jedná se o pravidelný Hoeffdingův strom obsahující dodatečný volitelný uzel. Díky tomu je umožněno využít k průchodu více cest a navracet se po více listech. Volitelný uzel tak spojuje rozhodovací cesty několika způsoby. Volitelný uzel se vyskytuje v několika různých dílčích stromech, které jsou postupně procházeny. Tyto dílčí části by samy o sobě mohly obsahovat více volitelných uzlů. Tímto jevem je potenciál pro dosažení různých listů vynásoben každou z možností. Rozhodování u tohoto typu Hoeffdingova stromu je založeno na kombinaci předpovědí použitelných listů na konečný výsledek. Dle myšlenky (Holmes et al., 2012) by standardní soubor reprezentace mohl vyžadovat sto kopií stromu, které by se lišily jen v listech. Efektní reprezentace prezentovaná jako libovolný strom by pak vyžadovala téměř stokrát méně místa pro případ, že by mohly být různé listy nahrazeny volitelnými uzly rozdělujícími sto různých vedoucích cest do sta různých listových variant.

2.8.5 Hoeffdingův adaptivní strom

Další vylepšenou obměnou je Hoeffdingův adaptivní strom (*Hoeffding Adaptive Tree*), který opět dodržuje základní koncept Hoeffdingova stromu. Obsahuje v sobě však několik vylepšení. Především se jedná o zavedení pozdějšího přesného výpočtu výsledku. Tohoto efektu je dosaženo díky tomu, že každý list ukládá odhad současné chyby klasifikace a váhu každého uzlu při rozhodovacím procesu, který je úměrný k druhé mocnině inverzní chyby. Dle (Bifet a Gavaldà, 2009) je definován jako Hoeffdingův strom, jenž je založen na adaptivním učení se z datového proudu, které se mění v průběhu času bez nutnosti pevné velikosti posuvným oknem. Optimální velikost výsuvného okna je pro uživatele velmi obtížně odhadnutelný parametr. Je to zapříčiněno tím, že jeho závislost je odvozena od rychlosti změny distribuce datového souboru.

2.9 Bayesovské klasifikátory

Jak uvádí (Berka, 2003) metody bayesovské klasifikace vycházejí z Bayesovy věty o podmíněných pravděpodobnostech. Ačkoliv se tedy jedná o metody pravděpodobnostní, jsou intenzivně zkoumány v souvislosti se strojovým učením a uplatňují se rovněž v systémech pro dobývání znalostí. Bayesův vzorec pro výpočet podmíněné pravděpodobnosti říká, že platí hypotéza H, pokud pozorujeme evidenci E, která má podobu:

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \tag{13}$$

Apriorní pravděpodobnost hypotézy P(H) odpovídá znalostem o zastoupení jednotlivých hypotéz (tříd) bez ohledu na nějaké další informace. Podmíněná pravděpodobnost P(H|E), též nazývaná *aposteriorní*, vyjadřuje, jak se změní pravděpodobnost hypotézy, pokud víme, že nastala evidence E a P(E) vyjadřuje pravděpodobnost evidence (pozorování).

2.9.1 Naivní Bayes

Naivní Bayes (*NaiveBayes*) je pravděpodobně jeden z nejběžněji používaných modelů založený na Bayesovu teorému ve strojovém učení. Odhadování podmíněné pravděpodobnosti P(x|y) není triviální, protože spočívá v nalezení exponenciálního množství sdružených pravděpodobností jednotlivých atributů. Za určitých předpokladů lze tento problém zjednodušit. Cenou zjednodušení je možné snížení přesnosti. Pragmatický přínos je však velmi výrazný pro úlohy popsané mnoha atributy (desítky a mnohem více), jak uvádí (Carela-Español et al., 2015). Předpokládá se nezávislost mezi atributy tzn., že polohy slov v dokumentu nemají význam.

2.9.2 Naivní Bayes Multinomial

Jak popisuje (Raschka, 2014; Mccallum a Nigam, 1998) jedná se o obměnu Naivního Bayese, která je určena spíše pro data textového charakteru. Naivní Bayes Multinomial explicitně modeluje počet slov. Navíc upravuje základní výpočty k řešení oproti Naivnímu Bayesovi, který modeluje dokument jako přítomnost či nepřítomnost jednotlivých slov.

3 Metodika práce

3.1 Metodika a postup práce

Výstupem této diplomové práce bude reprezentace výsledků jednotlivých experimentů prováděných nad zvolenými daty. V datech budou identifikována významná slova a slovní spojení, která jsou typická pro určitou třídu dokumentů. Důraz bude kladen na nalezení níže uvedených charakteristik na objemných datech. Na data bude postupně aplikována celá řada různých algoritmů s různými parametry. Celý postup práce lze shrnout do následujících etap:

1. Výběr oblasti zkoumání,
2. Získání a shromáždění dostatečného množství textových dat.
3. Volba vhodných algoritmů a metod strojového učení pro získání informace.
4. Důkladná příprava a normalizace dat.
5. Provádění jednotlivých experimentů a měření.
6. Shromažďovaní, ukládání a porovnávání průběžných výsledků.
7. Interpretace dosažených výsledků a formulace následných doporučení.

3.2 Výběr oblasti zkoumání

Nejdříve je nutné vybrat data, na kterých lze dokázat rozdíly při klasifikaci různými klasifikátory. V návaznosti na prováděnou výzkumnou činnost na univerzitě bylo rozhodnuto o výběru dat týkajících se hodnoticích recenzí hotelů. Autorovou vášní je cestování a s tím souvisí i hledání vždy vhodného ubytování. Je zde tedy větší motivace ke zkoumání těchto dat a vyvození, co nejlepších výsledků. V této části je vhodné se zmínit o použitých datech pro experimentování. Experimenty provedené v této práci jsou navrženy a aplikovány za účelem srovnání různých přístupů ke klasifikování a jejich vlivu na výsledek dolování znalostí z textových dat.

3.3 Charakteristika vstupních dat

Jednotlivé experimenty jsou provedeny nad textovými daty obsahujícími názory zákazníků, kteří byli ubytováni v různých hotelech po celém světě a svoji rezervaci i následné hodnocení služeb provedli přes Internet prostřednictvím hotelového rezervačního systému. Hotely jsou na webu organizovány v hierarchii kontinent-stát-město-(městská část)-hotel a kromě informací o cenách, vybavení a podmínkách zde lze nalézt již zmíněné recenze zákazníků týkající se jejich pobytu v daném hotelu. Každá recenze se skládá z identifikačního prvku recenzenta, jeho celkového hodnocení (na desetibodové stupnici) a samotné recenze, která je zapsána v přirozeném jazyce a má dvě části: negativní a pozitivní zkušenost s hotelem.

Data byla shromážděna v rámci výzkumu na Ústavu informatiky Provozně ekonomické fakulty Mendelovy univerzity v Brně. Data stažena na základě země původu, kterou recenzent uvedl, a na základě toho určen i jazyk recenze. Byla vypuštěna informace o identifikaci recenzenta a o jeho celkovém bodovém hodnocení. Takto vznikly kolekce textových dat v různých přirozených jazycích obsahující hotelové recenze, z nichž většina zahrnuje nedostatky typické pro text psaný v přirozeném jazyce: překlepy, chybějící písmena, gramatické chyby apod. (Žižka a Dařena, 2010)

Pro příklad zde bude zmíněna ukázka jedné pozitivní (třída 1) a jedné negativní recenze (třída 2) pro anglický jazyk:

Tab. 2: Příklad anglické recenze

třída	Obsah hodnocení
1	We had a short but comfortable stay. We were having car trouble and the staff were particularly helpful in giving us details and directions to a garage, and a translation for us to help us when we arrived. Thank you.
2	The room was a bit dark and the rate should include breakfast. I wouldn't recommend it without seeing the other rooms. There were no other guest reviews on the site.

Řádek textového dokumentu reprezentuje jednu recenzi mající následující podobu:{*třída znak tabulátoru text recenze znak konce řádku*}.

V tomto formátu jsou recenze v mnoha světových jazycích. Z důvodu velkého množství nasbíraných dat a internacionalizace anglického jazyka byly pro experimentální činnost v této práci zvoleny anglicky psané recenze. Tvořící podstatnou část z dolovaných dat. Jedná se o 2 mil. anglicky psaných recenzí z celkového množství 5 mil. Tento výběr dat můžeme proto považovat za adekvátní pro provádění experimentů.

3.4 Převod dokumentů na vektory

Pro provádění textových dokumentů do vektorové podoby je vhodné nalézt vhodný nástroj. Řadou experimentů je ověřená aplikace **VecText**[1] implementovaná v jazyce Perl. Aplikace poskytuje jak grafické uživatelské rozhraní (*GUI*), tak i rozhraní textové (CLI). Mezi výhody aplikace patří mnoho možností pro vstupní a výstupní soubory. Podporováno je i mimo jiné filtrování a vážení atributů. K dispozici se nabízí dále např. stemming či odstranění stop slov. Vstupem pro tento nástroj jsou data z předchozí sekce. Soubor je následně načten a z každého souboru jsou odstraněny různé nevhodné znaky typu speciálních znaků, jenž pro rozhodování bezesporu nemají žádný vliv. Následně dochází k převedení dokumentů do vektorové reprezentace, kde jednotlivé prvky mají hodnoty dány nastavením vah. V konečné fázi jsou vektory zapsány do souborů ve zvoleném výstupním formátu.

[1] https://akela.mendelu.cz/đarena/VecText/

Vybrat si lze z formátů:

- **C5** formát vyžadovaný pro softwarové balíčky C5/See5 (soubor s příponou „.names“ rozšířeně popisuje atributy, třídy a soubor s příponou „.data“ obsahující trénovací data pro klasifikaci),
- **SPARSE** formát řídké matice, čárkami oddělený seznam atributů číselných dvojic hodnot po kterých následuje označení třídy,
- **CSV** čárkami oddělený seznam hodnot atributů následovaný značkou třídy se jmény atributů na prvním řádku,
- **CLUTO** formát řídké matice pro softwarový balík Cluto, vytvářející rovněž soubor s popisky třídy, pokud je to vhodné,
- **SVMlight** formát řídké matice pro softwarový balík *SVMlight*,
- **ARFF** (*Attribute-Relation File Format*) používaný v SW WEKA.

V této práci je použit formát ARFF [2]. ARFF soubor má dvě odlišné části. První částí je záhlaví souboru, po kterém následuje informace o datech. Takto vypadá hlavička souboru:

```
@RELATION data

@ATTRIBUTE SPEND NUMERIC
@ATTRIBUTE POSITION NUMERIC
@ATTRIBUTE TEMPERATURE NUMERIC
@ATTRIBUTE PROBLEMS NUMERIC
@ATTRIBUTE SITUATION NUMERIC

...

@ATTRIBUTE _CLASS_ {2,1}
```

a v následující formě je interpretována datová část:

```
@DATA
0,0,5.651,0,0,0,0,0,0,0,...,0,0,0,0,0,0,0,0,0,0,1.336
0,0,0,5.988,0,0,0,0,0,0,...,0,0,0,0,0,0,0,0,1.211,0
0,0,2.738,0,0,0,0,0,0,0,...,0,0,3.554,0,0,0,0,0,0,0
0,2.421,0,0,0,0,0,0,0,0,...,0,0,1.336,0,0,0,0,0,0,0
0,0,0,0,3.307,0,1.336,0,3.155,0,...,0,0,0,0,0,0,0,0,0,0
```

Tento formát má jednu velkou nevýhodu, data jsou ukládána v podobě „úplných“ vektorů. Soubory tohoto formátu, tak oproti formátům používající pro interpretaci řídké matice, alokují mnohem více diskového prostoru.

Důležitou součástí je volba vhodného filtrování, což má vliv na to kolik a především jaké atributy budou výsledné vektory obsahovat. Bezesporu je žádoucí, aby

[2] http://www.cs.waikato.ac.nz/ml/weka/arff.html

množství atributů bylo co nejmenší a bylo možné snazší vyhodnocení. Dle experimentální činnosti provedené s těmito daty dříve bylo přistoupeno k nastavení minimální délky slova (*minimal word length*). Daný nástroj nabízí dále možnost použití filtrace pro *global term* (kolikrát se daný term vyskytuje ve všech dokumentech) či *document term* (v kolika dokumentech se term vyskytuje). Další klíčovou volbou je volba vah. Lze zvolit z několika možností, o kterých bylo pojednáno v sekci 2.5.3. Obecně nejvíce užívanou metodou je použití vah TF-IDF. Pro srovnání bude použita lokální váha TP. Naskytuje se v tomto nástroji navíc možnost zkoumat nejen jednotlivá slova (*unigramy*), ale i uspořádané n-tice slov (*n-gramy*). Problematikou n-gramů se zabývalo několik prací. Ukázalo se, že použití bigramů (dvougramů) výrazně nezvyšuje přesnost klasifikace sentimentu. Použitím n-gramů navíc výrazně narůstá počet atributů. Mezi velkou výhodu této aplikace patří, že v režimu příkazové řádky lze veškeré možnosti zadat pomocí parametrů z příkazové řádky. Aplikaci lze poté snadno začlenit do složitějšího dolování dat při integraci několika softwarových balíků či při provádění více konverzí v dávkovém režimu. Snadno si lze také v grafickém uživatelském rozhraní nadefinovat všechny potřebné příkazy pro režim příkazové řádky. Aplikace dokáže v podobě textového řetězce vygenerovat příkazy pro ovládání pomocí příkazové řádky, čehož lze využít pro dávkové zpracování či pro skript.

3.5 Používané softwarové prostředky

Po převedení dokumentů do vektorové reprezentace můžeme na daný soubor aplikovat vybrané algoritmy strojového učení. Existuje velké množství algoritmů umožňující klasifikaci, viz sekce 2.9. Vhodné se především jeví rozhodovací stromy poskytující možnost snadné interpretace výsledků. Vycházet lze zde z řady úspěšných výsledků prováděných za pomocí rozhodovacích stromů především využití algoritmu J48 při mnoha experimentech.

Existuje velké množství komerčních i nekomerčních aplikací pro dolování znalostí. Mezi komerční aplikace lze zařadit IBM SPSS Modeler, SAS Enterprise Miner. Nekomerční aplikace jsou např. RapidMiner, KNIME či WEKA. V řadě programovacích jazyků lze také nalézt knihovny implementující zvolené algoritmy strojového učení. Jejich hlubší zkoumání není smyslem této práce. V této práci je tedy spíše uvažováno o aplikačním řešení. Hlavním kritériem volby je využití metody pro inkrementální učení, které je v této oblasti prozatím novou záležitostí. Pro potřeby práce je snahou zvolit prostředí, které umožňuje zároveň inkrementální, tak i dávkové učení. Dle zvolených kritérií se tak ukázalo prostředí Weka nejvhodnějším prostředím splňujícím dané požadavky. Prostředí Weka lze doplnit o prostředí MOA umožňující práci s více hodnotícími modely.

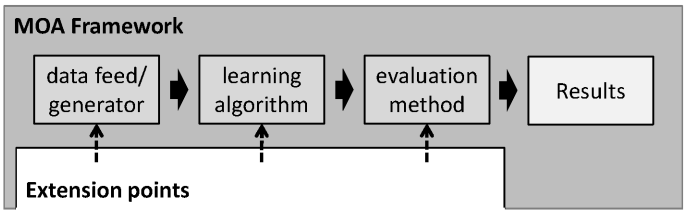
3.5.1 WEKA

Weka je otevřeným softwarový balík využívaný ve velkém množství pro data mining. Tento systém je vyvinut na Univerzitě Waikato na Novém Zélandu. Systém Weka využívá objektově orientovaný jazyk Java. Díky tomu je umožněn bezproblémový chod na všech dostupných operačních systémem. Weka poskytuje implementaci *state-of-the-art data mining* a algoritmů strojového učení. Mezi výhodu tohoto systému patří možnosti jako asociační pravidla, filtrování, klasifikace, shlukování, vizualizace, regrese atd. a to v jediném prostředí. Nástroj je možné ovládat jak prostřednictvím grafického uživatelského rozhraní, tak z příkazové řádky. Druhý jmenovaný způsob použití je méně paměťově náročný a nabízí funkcionalitu, která není dostupná přes grafické uživatelské rozhraní.

3.5.2 MOA

Pro masivní zpracování dat se velice dobře hodí softwarový balík MOA (*Massive online analysis*). Obsahuje řadu nástrojů pro hodnocení a spoustu algoritmu strojového učení. Tento software vznikl v souvislosti s projektem WEKA. Je jakýmsi rozšířením funkcionality pro potřeby vyhodnocování experimentů pro masivní objemy dat. Prostředí MOA je opět založeno na platformě Java, což umožňuje řešení velice složitých problému. Cílem MOA je poskytnout vylepšené rozhraní pro běh experimentů založených na dolování dat pomocí toku dat (*data stream*). Pomocí technologie toku dat je umožněno:

1. Uložení nastavení pro datové proudy jak reálné, tak syntetické. Díky tomu pak lze provádět opakovatelné experimenty.
2. Je jakousi sadou existujících algoritmů a měřícího rámce pro srovnání.
3. Jednoduše rozšiřitelným prostředím pro nové proudy (*data streams*), algoritmy a hodnotící metody.



Obr. 5: Schéma fungování prostředí MOA, Zdroj: (Bifet et al., 2010)

Základem je generátor dat (*data feed/generator*) pomocí, kterého dochází k generování dat pro algoritmus. Následuje fáze aplikace učícího algoritmu (*learning algorithm*). V součinnosti s tím je aplikována hodnotící metoda (*evaluation method*). V konečné fázi je vygenerován report s výsledky (*Results*).

Pro běh úloh lze využít jak grafické prostředí (GUI), tak i příkazovou řádku (*command line execution*). V současné verzi MOA je podporováno několik možností využití:

1. klasifikace toku dat (*stream classification*)
2. shlukování toku dat (*stream clustering*)
3. detekce odlehlých hodnot (*outlier detection*)

V této práci bude omezeno na část týkající se klasifikace toku dat.

Klasifikace toku dat (*stream classification*)

Prostředí MOA nabízí velké množství algoritmu pro klasifikaci, které jsou rozděleny do klasifikačních skupin:

1. Bayesovské klasifikátory (*Bayesian classifiers*)
   - Naive Bayes
   - Naive Bayes Multinomial
2. Rozhodovací stromy (*Decision trees classifiers*)
   - Decision Stump (*Decision Stump*)
   - Hoeffdingův strom (*Hoeffding Tree*)
   - Hoeffdingův alternativní strom (*Hoeffding Option Tree*)
   - Hoeffdingův adaptivní strom (*Hoeffding Adaptive Tree*)
3. Meta klasifikátory (*Meta classifiers*)
   - Bagging
   - Boosting
   - Bagging using ADWIN
   - Bagging using Adaptive-Size Hoeffding Trees
   - Perceptron Stacking of Restricted Hoeffding Trees
   - Leveraging Bagging
4. Funkční klasifikátory (*Function classifiers*)
   - Perceptron
   - SGD: Stochastic Gradient Descent
   - SPegasos

5. Klasifikace prouděním (*Drift classifiers*)
    - SingleClassifierDrift

Není možné se zaměřit na všechny algoritmy zároveň a zcela jistě to není ani účelem zadané práce. V této práci se tedy zaměříme na pevně vymezenou část *Rozhodovacích stromů* a to přesněji na část týkající se *Hoeffdingových stromů*, které budou podrobeny hlubší analýze.

Modely vyhodnocení

MOA ve svém běhovém prostředí nabízí velké množství modelů pro vyhodnocování. Experimentální činností tak budou vhodné jednotlivé způsoby prozkoumány a vybrat nejlépe vypovídající model pro interpretaci. Nachází se zde několik modelů, které si nejprve rozebereme:

- **LearnModel** – Model, který se dokáže naučit z přicházejícího toku a výsledkem je vygenerovaná stromová struktura.
- **EvaluateModel** – Hodnotící statický model dle toku dat.
- **EvaluatePeriodicHeldOutTest** – Zhodnocení klasifikátoru na toku dat s pravidelným testováním založeným na vybraných datech pro testování.
- **EvaluateInterleavedTestThenTrain** – Vyhodnocení klasifikátoru na toku dat s otestováním a natrénováním pro každý příklad.
- **EvaluatePrequential** – Vyhodnocení klasifikátoru na proudu testování a trénování s každým příkladem v sekvenci. Mohou se používat posuvná okna (*sliding window*) či mechanismu ztrácejícího faktoru zapomnění (*fading factor forgetting mechanism*).
- **EvaluateInterleavedChunks** – Vyhodnocení klasifikátoru na toku dat testováním a trénováním pro každou sekvenci „chunk“ dat v daném pořadí.

Nejčastěji používanými modely jsou *EvaluateInterleavedTestThenTrain* či *EvaluatePrequential*. Oba tyto modely jsou založeny na principu, že každý jednotlivý příklad je předem nejprve otestován, než je použit pro trénování. Vždy dochází k inkrementální aktualizaci přesnosti. Model je založen na neustálém testování na příkladech, co ještě nezná. Výhodou daného modelu je, že není třeba vyčleňovat určitou sadu dat potřebných pro testování maximálního využití dostupných zdrojů dat a dochází zde k doučování průběžně tzv. za běhu. S tím úzce souvisí fakt, že model zajišťuje hladký graf přesnosti v průběhu času. Pro každý příklad bude neustále měněn význam při zohlednění celkového průměru (Gama et al., 2009; Bifet a Gavaldà, 2009).

3.6 Používané hardwarové prostředky

Bylo nutné provést velkou řadu experimentů z danými daty. Do experimentování byly tak zařazena dvojice zařízení.

Prvním je notebook, kde bylo vždy prováděno samotné předzpracování dat a ověřování dosažených výsledků. Disponuje následující konfigurací:

- název modelu: Acer TravelMate 5744
- operační systém: Windows 10 Professional 64-bit
- procesor: Intel Core i3-370M, 2.4 GHz
- operační paměť: 2 × 4 GB, DDR3 RAM (1333 MHz)
- pevný disk: Hitachi, 500 GB, 5400 ot./min

Pro experimentální činnost s ohledem na množství pokusů byl využit stolní počítač s platformou Windows, kde byla prováděna většina experimentální činnosti:

- název modelu: Hewlett-Packard HP xw4600 Workstation
- operační systém: Windows 7 Professional 64-bit
- procesor: Intel Core2 Quad Q9400, 2.66 GHz
- operační paměť: 4 × 2 GB, DDR2 RAM (800 MHz)
- pevný disk: Seagate, 500 GB, 7200 ot./min

4 Výsledky

4.1 Příprava dat

Pro provádění experimentování s textovými daty bylo nutné textová data patřičně předzpracovat. Předzpracování textových dat (*text preprocessing*) se ukázalo být jako velice důležitou aktivitou celé práce. Předzpracování umožňuje zároveň redukovat šum i irelevantní obsah původních textových dokumentů. Jak je zmíněno v sekci 2.6, existuje celá řada metod předzpracování textových dat.

Data jsou rozdělena na reprezentativnější celky. Pro srovnání je zvolen dvojí výběr vah termu: (TF-IDF), jenž je nejčastěji užívanou kombinací vah termu a váhy termu TP (*Term Presence*), jenž je často volena pro textová data. Rozdělení dat do data setů je znázorněno v Tab. 3. Uvedeno je pro potřeby podrobnější analýzy rozděleny na data pro 2 tis., 5 tis., 10 tis., 25 tis., 50 tis. dokumentů. Rozdělení se ukázalo pro obě váhy termu totožné.

Tab. 3: Rozdělení data setů

Počet dokumentů	Počet unikátních slov
2 000	934
5 000	1 734
10 000	2 550
25 000	4 205
50 000	6 117

Z celého balíku různojazyčných recenzí byly vybrány anglicky psané recenze. Anglicky psané recenze tvoří 2 mil. recenzí z celkového počtu 5 mil. vícejazyčných recenzí. S použitím algoritmu pro náhodný výběr byla vybraná data. Celkem byly takto vybrány zvolené obměny recenzí, na kterých byla následně prováděna další experimentální činnost. Algoritmus náhodného výběru implementovaného v jazyce Python je umístěn na přiloženém CD.

Data byla následně předzpracována za pomocí programu **VecText**. Tento SW poskytl velice dobrý podklad pro experimentování s daty. Pro předzpracování byla aplikována pravidla pro snížení frekvence slov v dokumentu (`min_global_frequency`) dle dříve ověřených experimentů na četnost 5 a dále bylo aplikováno pravidlo pro výběr slov s délkou 3 a více znaků (`min_word_length`) – tím se odstraní nepodstatné předložky a spojky, jenž působí při experimentování jako šum a nemají tak význam pro klasifikování.

Na dané datasety je následně aplikována metoda `case-folding`, všechna slova nahrazena verzálkami slov (velkými písmeny). Dle preferencí byly nastaveny zvolené parametry. Zvoleno bylo kódování utf-8, představující univerzální řešení, co se týče kódování dat. Při nevhodně zvoleném kódování bylo jinak nutné pro prostředí WEKA provést konverzi. Důvodem bylo, že některé znaky

byly špatně interpretovány či nastaly problémy s duplicitami. Důležitým parametrem je dále výběr atributu klasifikační třídy (`class_position`). V našem případě se pozice třídy nachází na první pozici. Dalšími důležitými parametry jsou nastavení globální (`global_weights`) a lokální váhy (`local_weights`). Výstupní formát (`output_format`), vytvoření slovníku s četnostmi (`create_dictionary`), (`print_statistics`) pro vytištění statistických údajů: Příklad spouštěcího kódu pro předzpracování dat pro náhodně vybraný dataset s definovanými parametry:

```
--input_file="C:/Testy_Diplomka/my_select.text" --encoding="utf8"
--class_position=1 --output_dir="C:/Testy_Diplomka/"
--output_file="my_select" --local_weights="Term Frequency (TF)"
--global_weights="IDF" --min_word_length=3 --min_global_frequency=5
--normalization="none" --output_format="ARFF"
--create_dictionary="with frequencies" --print_statistics
--logarithm_type="natural" --output_original_texts
--output_tokens --sort_attributes=none
```

S ohledem na množství spouštěných experimentů je přistoupeno k dávkovému spouštění pomocí příkazové řádky. Jedním z faktorů, proč byl zvolen tento způsob, je především rozšířená možnost správy výsledků a přehled nad těmito experimenty. GUI Weka poskytuje jednoduší řešení pro provedení malého množství experimentů. Pro rozsáhlejší experimentální činnost se GUI řešení prokázalo nevhodné. Z tohoto důvodu byl zvolen dávkový přístup zpracování dat pomocí příkazové řádky. Pomocí příkazové řádky lze také spravovat snáze přidělenou paměť. Při spouštění algoritmu J48 byly paměťové nároky obrovské. Paměť 8 GB se ukázala být při několika experimentech nedostatečná a bylo nutné některé experimenty zopakovat.

$$-Xmx8g - XX : -UseGCOverheadLimit \tag{14}$$

Chod prostředí Weka běžící na platformě Javy lze ovlivnit pomocí dvou přepínačů pracujících s pamětí znázorněných dle vzorce (14): První zmiňovaný umožňuje nastavit maximální velikost operační paměti, kterou může program využít (v kódu tedy nastaveno na hodnotu 8 GB RAM, což odpovídá paměti testovaného stroje). Druhý v pořadí je přepínač umožňující vypnout omezení času virtuálního stroje.

Následně byl vytvořen dávkový skript ukazující, jakým způsobem lze prostředí Weka ovládat pomocí příkazové řádky v prostředí Windows. Využito zde je znalostí jazyka Python a vytvoření dávkového souboru pro spuštění, který byl různě modifikován. Obměny spouštěných skriptů umístěny na CD.

S ohledem na množství různých modifikací bylo nutné stanovit i systém spouštění jednotlivých spouštěcích skriptů. Jednotlivé experimenty byly několikrát opakovány v různých modifikacích pro lepší správnost dosažených výsledků. Bylo tak přistoupeno k systematickému třídění výsledků. Pro přehlednost experimentů bylo pro experimentování zavedená jednotná koncepce pojmenování.

`Jazyk_ObjemDat_Algoritmus_Parametr -- WEKA`
`Jazyk_ObjemDat_Algoritmus_Parametr_Model -- MOA`

4.1.1 Srovnání zvolených experimentů

Postupně budou v následujících sekcích rozebrány jednotlivé přístupy k daným parametrům. Zaměříme se především na dobu zpracování, správnost a také počet vygenerovaných slov při zadaném parametru. Pro třídění výsledků se nabízí jako vhodné použití skriptu pro „parsování“ naměřených výsledků pro zvolené algoritmy. Byl tak pro parsování výsledků vytvořen modul v jazyce Python – `ParserResult`. Modul obsahuje zadanou sadu skriptů:

- `parse_cli_result` – skript pro vyhodnocení výsledků z WEKA pro obecné charakteristiky,
- `parse_cli_MOA_result_EvalTT` – skript pro vyhodnocení výsledků z MOA pro model EvaluationInterLeavedTestThenTrain,
- `parse_cli_MOA_result_LearnModel` – skript pro vyhodnocení výsledků z MOA pro model LearnModel,
- `run_batch` – spouštěcí dávkový skript.

První zmiňovaný `parse_cli_result` dokáže pro vybrané algoritmy z vygenerovaných reportů výsledků vybrat důležité charakteristiky. Mezi něž lze zařadit {*Accuracy, TP Rate, FP Rate, Precision, Recall, F-Measure*}. Dále také {*Time taken to build model, Time taken to test model on training split, test_instances*}.

Druhý zmiňovaný skript je `parse_cli_MOA_result_EvalTT`. Tento skript lze aplikovat na výsledky vygenerované v prostředí Moa. Pomocí zvoleného skriptu si lze uložit do výsledného souboru charakteristiky zvoleného algoritmu. Důraz je kladen na rozdělení vygenerovaných reportů podle modifikací Hoeffdingova stromu a pro algoritmus J48. Algoritmus je uzpůsoben generování výsledků z modelů *EvaluateInterleavedTestThenTrain* či *EvaluatePrequential* generující dvouřádkový report s výsledky. Výsledný report je následně uložen do CSV souboru.

Třetím v pořadí je skript `parse_cli_MOA_result_LearnModel` vybírající důležité výsledky z *LearnModelu* generovaného prostředím MOA. Stejně jako předchozí skripty i tento využívá třídy `TextWriter` pro ukládání charakteristik do CSV souboru. Vybrány jsou charakteristiky pro J48: *Number of Leaves, Size of the tree* a pro Hoefdiingův strom: *tree size (nodes), tree size (leaves), active learning leaves, tree depth*. Soubor s vybranými charakteristikami je opět uložen v CSV formátu.

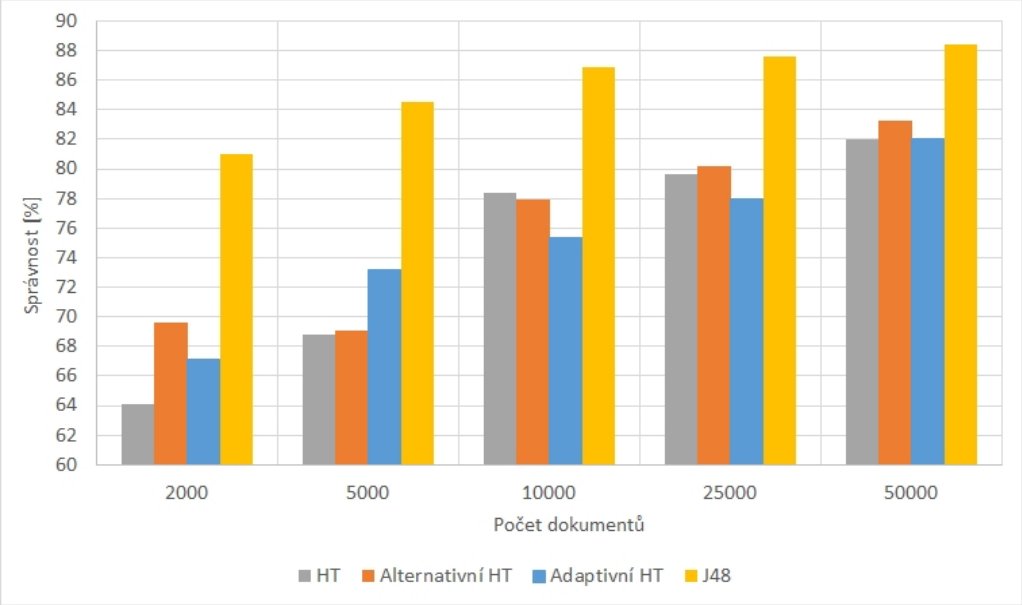
Následně byla vytvořena pomocná třída `TextWriter`, která umožňuje zápis výsledků do CSV souboru. Výsledky jsou průběžně zapisovány do CSV souborů. Na jejich základě je provedeno vyhodnocení. Získáno je takto velké množství výsledků, mezi nimiž je nutné se orientovat. Jako vhodný prostředek pro analyzování se nabízí prostředí Microsoft Excel, ve kterém lze využít funkce pro řazení či filtrování. Navíc je zde řada matematických operací, které lze aplikovat na získané výsledky.

4.2 Průběh experimentů

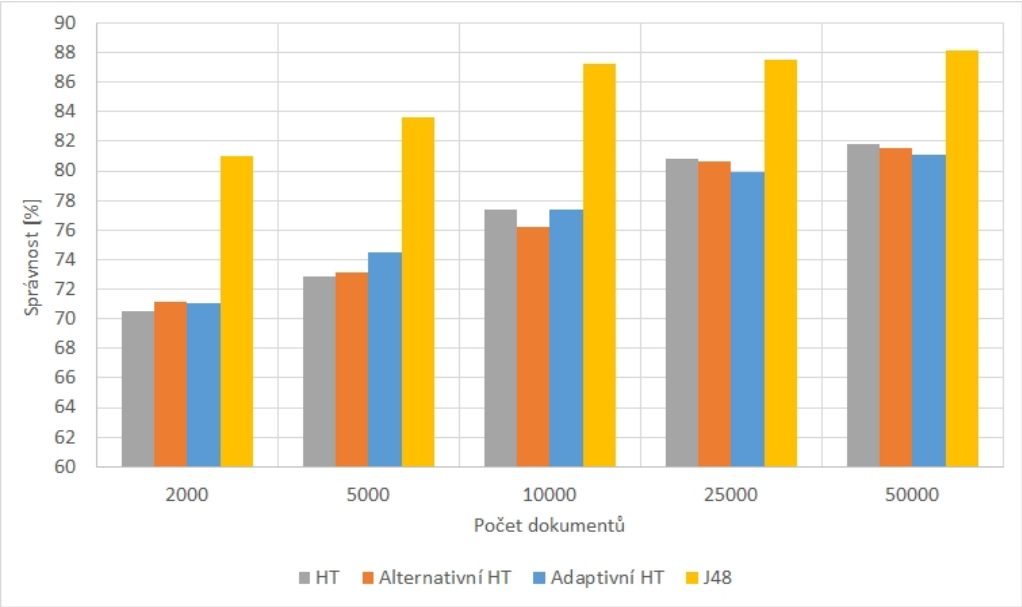
Pro experimentální činnost jsou zvoleny dva typy vah termu (TP) a (TF-IDF). Tyto dvě váhy budou srovnávány při zvolených kritériích experimentování spolu se zvolenými algoritmy. Pro trénování a následné testování je zvolena metoda rozdělení dat v poměru 66 % trénovacích a 33 % testovacích dat.

Prostředí Weka nedokázalo ve formě výsledného reportu vygenerovat stromovou strukturu. Není proto zcela znám přesný počet uzlů a listů pro dané obměny. Pro zjištění počtu uzlů je tedy aplikováno prostředí MOA umožňující interpretaci výsledků v podobě již zmíněného modelu *EvaluateInterleavedTestThenTrain*, jenž poskytuje informaci o počtu uzlů a listů stromové struktury. V prostředí MOA jsou k dispozici i obměny Hoeffdingova stromu, na které se srovnání v dané práci zaměřuje. Na druhou stranu prostředí Weka zajišťuje srovnání obměn metrik správnosti algoritmu Hoeffdingova stromu, ukázka viz příloha A.

Nejprve byly provedeny experimenty s váhou termu (TF-IDF). Váha termu (TF-IDF), jak je možné vyčíst z Obr. 6, vykazuje pro obměny Hoeffdingova stromu při nižším počtu instancí dokumentu nižší správnost u těchto algoritmů. S růstem počtu instancí v dokumentech však dochází k růstu správnosti algoritmů při určování důležitých slov. Při dosažení 25 tis. dokumentů lze pozorovat zvyšující se tendenci přiblížit se algoritmu J48. Při tomto objemu dat je dosahováno vyšší správnosti a je překonána hranice 80 % správnosti.



Obr. 6: Znázornění správnosti zvolených algoritmů při použití váhy termu (TF-IDF)



Obr. 7: Znázornění správnosti zvolených algoritmů při použití váhy termu (TP)

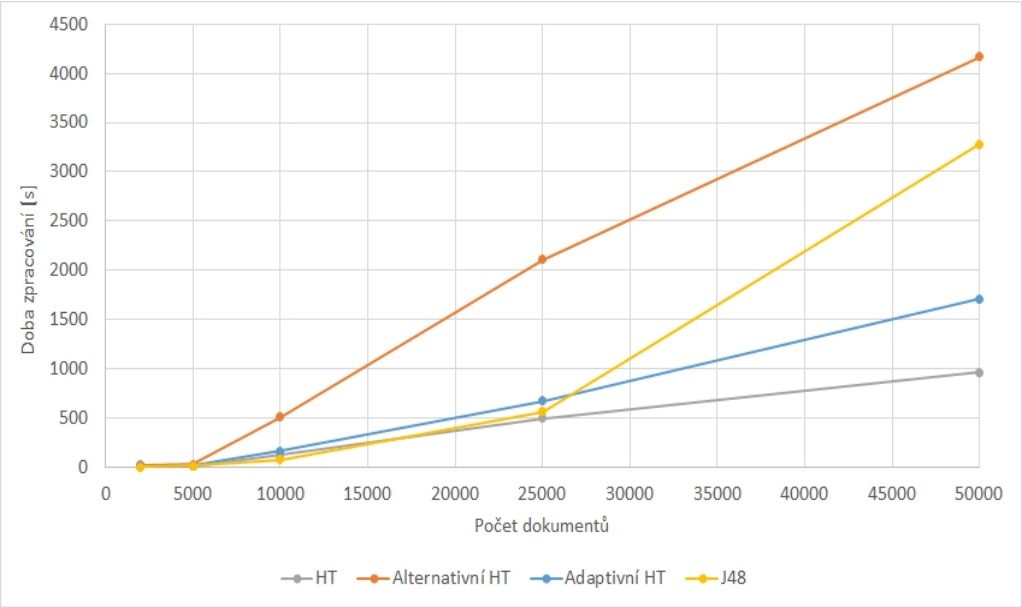
Oproti tomu na srovnávaném Obr. 7 pro váhu termu (TP) je možné vypozorovat, že při nižším množství dat je dosahováno o několik procent vyšší správnosti. Rozdíl mezi danými grafy není velký – jedná se o rozdíl v pouhých jednotkách procent.

Doba zpracování

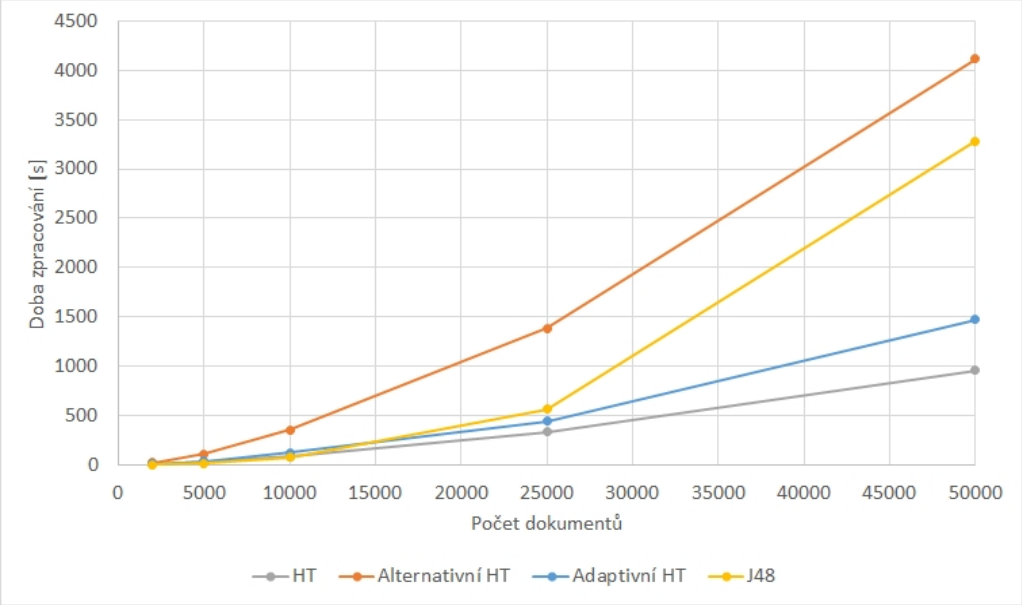
Na Obr. 8, 9 pro obě volby váhy termu (TF-IDF) i (TP) je viditelný velice podobný průběh grafu. Lze tedy dojít k názoru, že algoritmus Hoeffdingův strom vykazuje nižší dobu zpracování, stejně jako Hoeffdingův adaptivní strom, jenž mají přibližně 4 krát nižší dobu zpracování než je vidět u algoritmu J48. Algoritmus Hoeffdingův alternativní strom, zde ukazuje vyšší časovou náročnost než pro algoritmu J48.

Pro všechny obměny Hoeffdinogva stromu, jak je vidět z obou grafu je typický lineární průběh růstu doby zpracování. Oproti tomu algoritmus J48 má spíše průběh polynomiální, což by zcela jistě bylo více vidět při vyšším množství dat.

Zajímavé je pozorovat, že do množství 25 tis. dokumentů se doba zpracování Hoeffdingovým stromem, Hoeffdingovým adaptivním stromem a algoritmem J48 téměř neliší. Až při množství 50 tis. je znatelný rozdíl mezi těmito algoritmy.



Obr. 8: Vyhodnocení doby zpracování zvolených algoritmů (TF-IDF)



Obr. 9: Vyhodnocení doby zpracování zvolených algoritmů (TP)

Velikost vygenerovaného stromu

Vedle správnosti a doby zpracování, které jsou pouze jedny z mnoha možných kritérií experimentování, je možné pozorovat i související velikost vygenerovaného stromu. Na následujících stranách se tedy zaměříme na prezentaci výsledků z tohoto hlediska. Důraz bude kladen na velikost vygenerovaného stromu v závislosti na správnosti klasifikace pro algoritmus. Toto je velice důležitá část, podle které lze vyvodit závěry pro vhodnost využití algoritmů. Pro přehlednost zde bude uvedeno několik tabulek srovnávajících naměřené parametry pro dané algoritmy.

Tab. 4: Počet listů stromu pro algoritmus J48 (TF-IDF)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	80,97	157	79	1,37
5 000	84,50	307	154	12,23
10 000	86,87	503	252	76,56
25 000	87,62	1067	534	560,30
50 000	88,42	1813	907	3280,30

Tab. 5: Počet listů stromu pro algoritmus Hoeffdingův strom (TF-IDF)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	64,12	33	17	6,71
5 000	68,80	87	48	8,69
10 000	78,39	321	161	122,88
25 000	79,63	805	403	494,71
50 000	82,01	961	481	959,80

Tab. 6: Počet listů stromu pro algoritmus Hoeffdingův alternativní strom (TF-IDF)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	69,64	152	78	23,07
5 000	69,10	419	215	24,23
10 000	77,91	1492	748	507,03
25 000	80,16	3796	1900	2107,17
50 000	83,30	4478	2291	4170,69

Tab. 7: Počet listů stromu pro algoritmus Hoeffdingův adaptivní strom (TF-IDF)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	67,13	45	23	8,64
5 000	73,16	128	63	13,43
10 000	75,34	427	214	159,34
25 000	77,97	1041	521	670,43
50 000	82,05	1215	608	1708,69

Z naměřených hodnot v Tab. 4, 5, 6, 7 lze vyvodit, že s rostoucím počtem instancí roste postupně i správnost a počet uzlů a listů daného algoritmů. Nejlepších výsledků dosahuje s hlediska počtu uzlů a listů Hoeffdingův alternativní strom, u kterého roste počet uzlů i listů přibližně 4 krát více než u algoritmu J48 a u ostatních dvou variant Hoeffdingova stromu. S tímto úzce souvisí zmiňovaná delší doba zpracování pro Hoeffdingův alternativní strom.

Pro srovnání je zde uvedeno váhy termu (TP). Daná váha dosahuje podobných výsledků, jenž jsou představeny v následujících Tab. 8, 9, 10, 11. Rozdíl mezi počty uzlů a listů dosahuje jednotek uzlů (listů), což dokládají následující tabulky.

Tab. 8: Počet listů stromu pro algoritmus J48 (TP)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	80,97	153	77	1,37
5 000	83,62	259	130	12,23
10 000	87,22	505	253	76,56
25 000	87,55	1039	520	560,30
50 000	88,11	1851	926	3280,30

Tab. 9: Počet listů stromu pro algoritmus Hoeffdingův strom (TP)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	70,54	33	17	6,46
5 000	72,85	83	42	25,46
10 000	77,41	189	95	85,08
25 000	80,85	471	236	333,23
50 000	81,86	939	470	955,71

Tab. 10: Počet listů stromu pro algoritmus Hoeffdingův alternativní strom (TP)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	71,14	140	72	22,21
5 000	73,13	412	208	109,23
10 000	76,20	912	458	356,62
25 000	80,61	2318	1161	1380,62
50 000	81,58	4442	2273	4112,70

Tab. 11: Počet listů stromu pro algoritmus Hoeffdingův adaptivní strom (TP)

Počet instancí	Správnost	Počet uzlů	Počet listů	Doba zpracování
2 000	71,04	47	24	9,00
5 000	74,45	121	61	35,65
10 000	77,38	275	138	120,14
25 000	79,92	643	322	444,09
50 000	81,08	1263	632	1469,34

4.2.1 Závislosti na parametrech daných algoritmů

Během prováděných experimentů v návaznosti na článek (Netolický, 2015) bylo zjištěno, že podstatný vliv na na výsledky klasifikace mají změny různých parametrů, především pro Hoeffdingův strom změny parametrů ovlivňují výsledek klasifikace. Parametrů u Hoeffdingova stromu je velká řada. Bylo tak třeba věnovat velkou trpělivost a čas provádění jednotlivých experimentů. Díky této skutečnosti experimentování s těmito daty nabylo nového rozměru. V následujících odstavcích se tedy zaměříme na nejvýznamnější parametry. Následně jsou tyto parametry podrobeny detailnějšímu zkoumání. Srovnání parametrů Hoeffdingova stromu doplňuje experimentování s parametry algoritmu J48.

HT – parametrizace

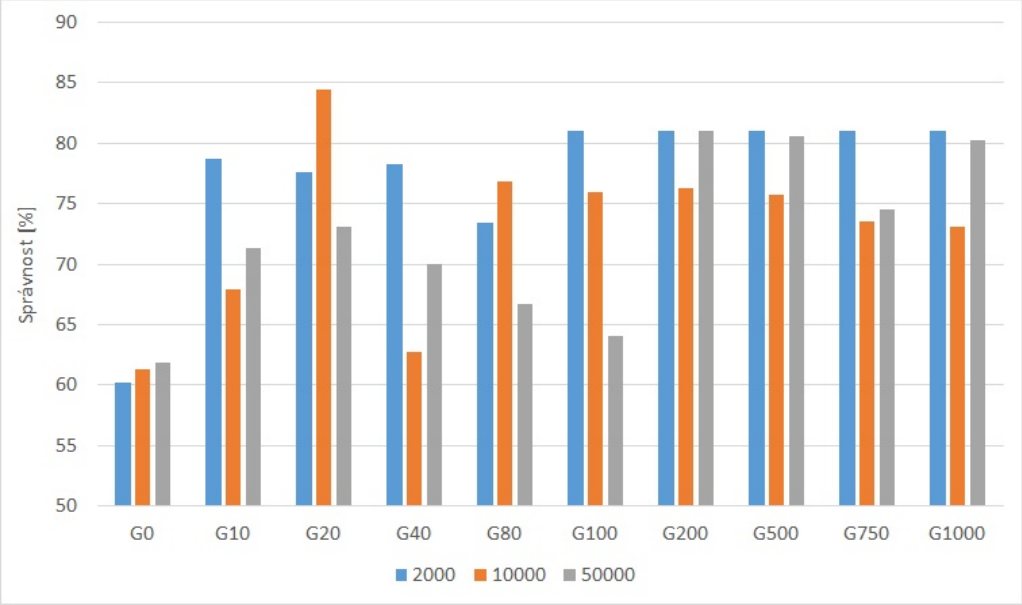
Dle provedených experimentů bylo zjištěno, že na celkovou kvalitu klasifikace mají podstatný vliv především tyto parametry a případná jejich kombinace:

- **GracePeriod**
- **TieThreshold**
- **Použitý klasifikátor pro listy**

GracePeriod

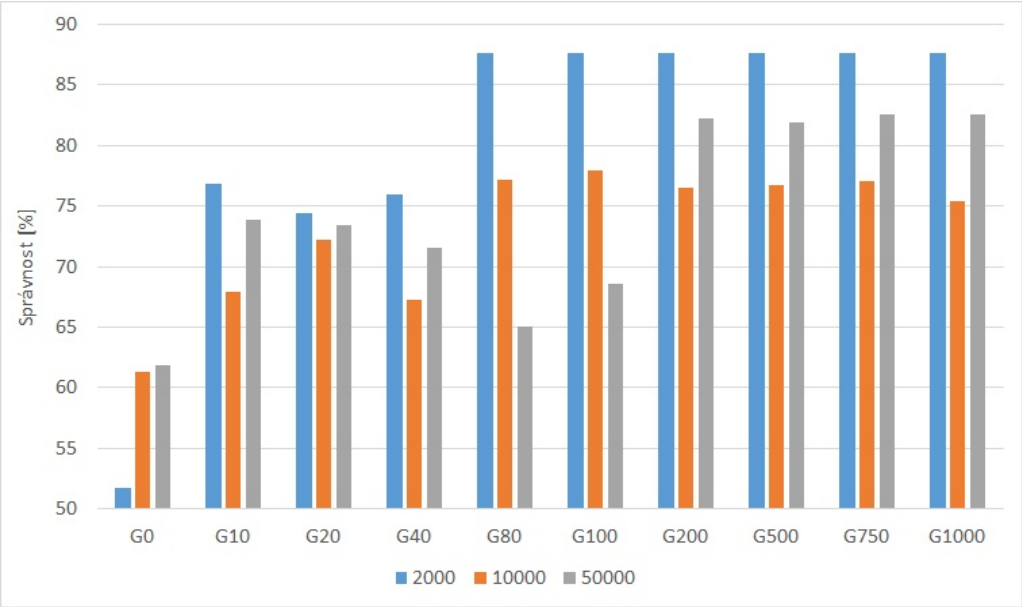
V rámci práce se snažíme zajistit přehlednost výsledků. Na základě této skutečnosti se autor rozhodl provést interpretaci výsledků pro 2 000, 10 000, 50 000 dokumentů. Ostatní výsledky jsou umístěny v Excel souborech na přiložením CD.

Vygenerovaný report z prostředí Weka poskytuje různé metriky týkající se měření kvality klasifikátoru. Pro potřeby analýzy je vybrána charakteristika *Accuracy* (Správnost), jenž je využívána i při předchozích experimentech. Zaměříme-li se na správnost (*Accuracy*).



Obr. 10: Srovnání správnosti parametru GracePeriod (TF-IDF)

Obr. 10 ukazuje na skutečnost, že hodnota parametru `G0` nemá zcela vypovídající hodnotu. Vysoké správnosti pro `G20` dosahuje Hoeffdingův strom při 10 tis. dokumentech. Tyto výsledky nejsou stabilní, jak je možné vidět z daného grafu. S důrazem na objem dat lze vyvodit, že výsledky pro 50 tis. dokumentů s ohledem na správnost jsou pro hodnoty parametru `G200` a `G500` nejvíce stabilní.



Obr. 11: Srovnání správnosti parametru GracePeriod (TP)

Srovnávaná váha termu (TP) na Obr. 11 nabízí výsledky podobného rázu s tím, že lze vypozorovat, že pro hodnoty v rozsahu `<G200,G500>` je patrný stejný průběh grafu. Nejvyšší správnosti je dosahováno pro 2 tis. dokumentů v rozsahu `<G80,G1000>`. Hodnota `G0` je i pro tuto frekvenci nevhodným nastavením parametru. Srovnáním lze odvodit, že největší rozdíl je pro `G20` a `G80`, kde se tyto váhy termu nejvíce liší v řádu 12–14 % správnosti.

Pro každé zvolené množství dokumentů je následně zvoleno srovnání statistických metrik v Tab. 12 a 13. Z následujících tabulek je vidět, že minimální hodnoty jsou 50 a 60 % správnosti. Tyto hodnoty odpovídají nastavení hodnoty parametru na `G0`. Průměrná správnost se pohybuje v rozmezí 78–80 % správnosti. Váha termu (TP) se ukazuje jako průměrně lepší s ohledem na správnost klasifikace v rozmezí 2–3 % správnosti. Zbylé charakteristiky lze vyvodit z daných tabulek.

Tab. 12: Statistické metriky srovnání správnosti pro GracePeriod (TF-IDF)

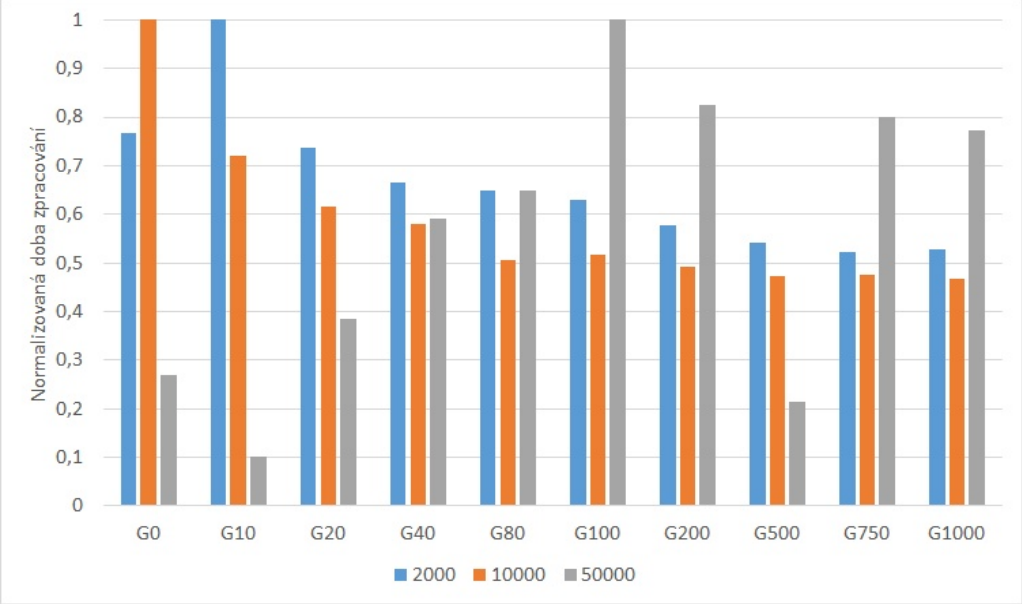
hodnota	2 000	10 000	50 000
MAX	81,06	84,45	81,03
MIN	60,23	61,31	61,92
Průměr	77,36	72,79	72,35
MED	79,89	74,63	72,20

Tab. 13: Statistické metriky srovnání správnosti pro GracePeriod (TP)

hodnota	2 000	10 000	50 000
MAX	87,60	77,91	82,52
MIN	51,71	61,30	61,86
Průměr	80,46	72,94	74,35
MED	87,60	75,94	73,65

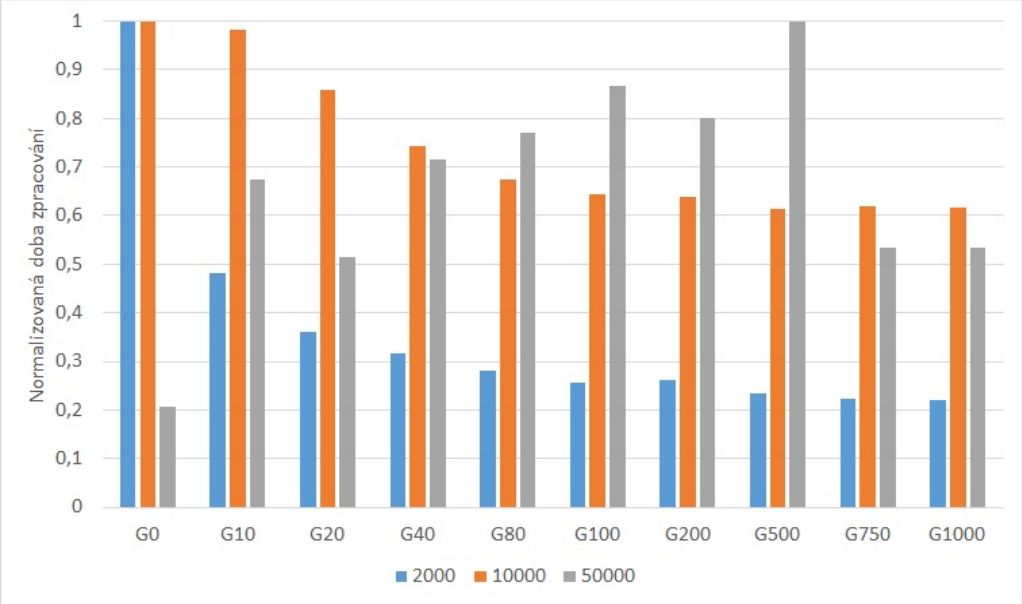
Normalizace doby zpracování

Z důvodu srovnatelnosti je doba zpracování normalizována vzhledem k objemu dat. Z Obr. 12 normalizované doby zpracování lze odvodit maximální hodnoty pro objem 2 tis. dokumentů, kdy dochází k nejdelší době zpracování daného objemu dat pro hodnotu parametru `G10`. Oproti tomu pro objem 10 tis. pro hodnotu parametru `G0`. Pro objem 50 tis. dokumentů je nejdelší doby zpracování dosaženo pro nastavení parametru na `G100`. Na druhé straně je poté nejkratší doby zpracování dosahováno pro 50 tis. dokumentů pro hodnotu parametru `G10`. Zajímavým výsledkem je, že pro rozptyl hodnot parametru `<G500;G1000>` je shodně nejkratší doby zpracování dosahováno pro objemy 2 a 10 tis. dokumentů.



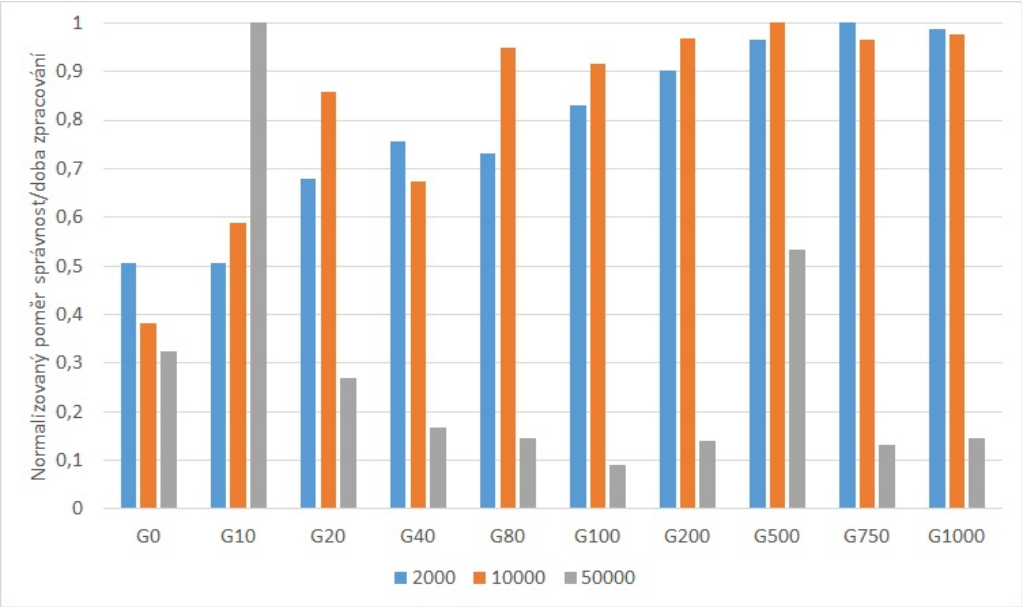
Obr. 12: Normalizovaná doba zpracování pro GracePeriod (TF-IDF)

Pro srovnávanou váhu termu (TP) na Obr. 13 mají výsledky pro nižší objem dat (2 a 10 tis.) podobný průběh. Větší rozdíly se projevují až při vyšším objemu dat. Pro objem 50 tis. dokumentů je nejdelší doby zpracování dosaženo pro nastavení parametru na `G500`. Na druhé straně je poté nejkratší doby zpracování dosahováno pro 50 tis. dokumentů pro hodnotu parametru `G20`. Zajímavým výsledkem je, že pro rozptyl hodnot parametru `<G500;G1000>` je shodně nejkratší doby zpracování dosahováno pro objemy 2 a 10 tis. dokumentů jako bylo naměřeno i pro frekvenci (TF-IDF).



Obr. 13: Normalizovaná doba zpracování pro GracePeriod (TP)

Následně je provedeno srovnání poměru správnost/doba zpracování v normalizované podobě vzhledem k objemu dat. Z Obr. 14 vyplývá, že pro 50 tis. dokumentů je vhodnější s ohledem na daný poměr nastavení parametru `G10`. Pro nižší objemy to tak ale nelze konstatovat. Pro nižší objemy dat (2 a 10 tis.) můžeme vyvodit, že je vhodnější nastavení parametru v rozsahu `<G500;G1000>`, kde je dosahováno nejvyšších hodnot.

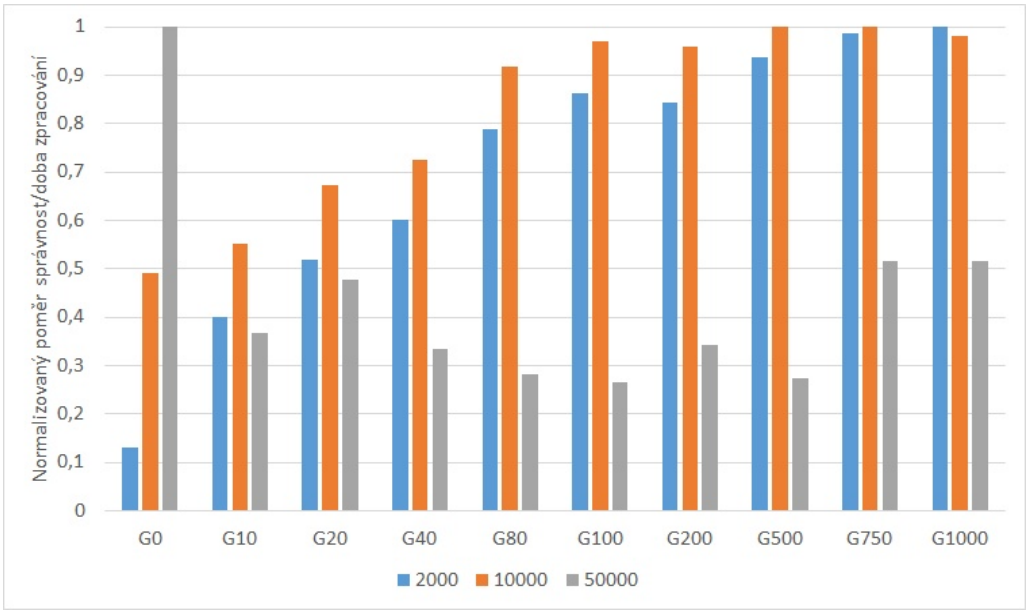


Obr. 14: Srovnání poměru správnosti/doba zpracování pro GracePeriod (TF-IDF)

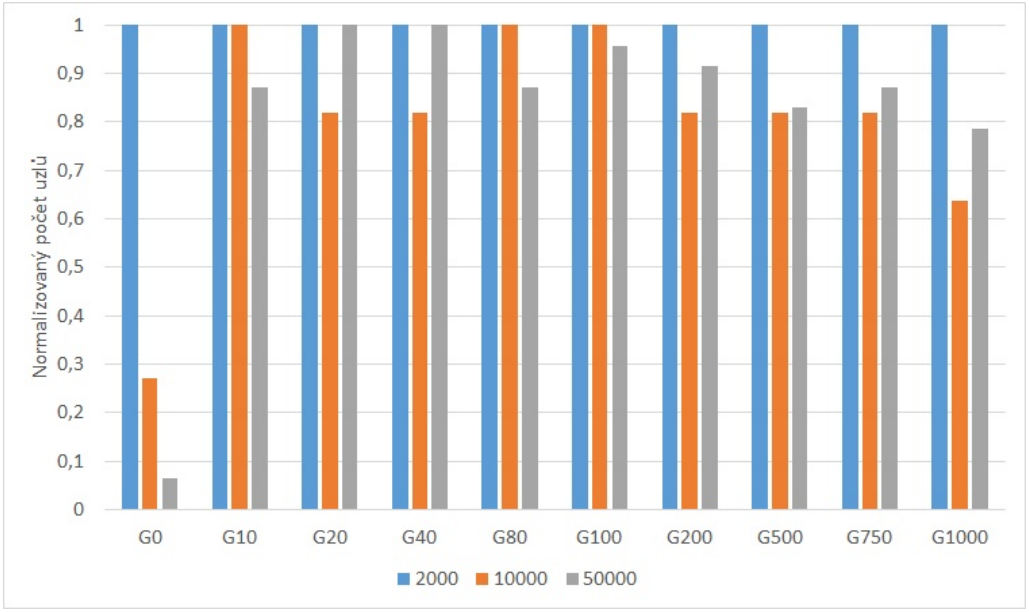
Z Obr. 15 pro frekvenci (TP) lze vyvodit podobné závěry jako pro frekvenci (TF-IDF). Jsou zde dosaženy pro nižší objemy dat (2 a 10 tis.) shodné výsledky pro rozsah `<G500;G1000>`, kde je dosahováno nejlepších hodnot. Rozdílný je však závěr pro 50 tis. dokumentů, kde je nejlepších hodnot daného poměru dosaženo pro `G0`. Nejhorších výsledků je na druhou stranu pro tento objem dosahováno pro nastavení parametru v rozsahu `<G80;G100>`.

Celé srovnání je doplněno o výsledky vygenerovaného počtu uzlů pro dané nastavení parametrů, zde jak bylo zmíněno v úvodu této části, je použito prostředí MOA, které disponuje stejnou implementací Hoeffdingova stromu i algoritmu J48. Lze tak provést dané srovnání napříč prostředími.

Následně je tedy více zaměřeno na naměřené výsledky z prostředí MOA poskytující informace o stromové struktuře daného algoritmu. Srovnání lze uzavřít počtem vygenerovaných uzlů pro dané obměny parametrů a různé změny množství dokumentů. Počet uzlů stromu se napříč srovnávanými váhami termu neliší. Lze tedy dané experimenty zobecnit do podoby normalizovaného grafu na Obr. 16. Z daného grafu lze vyčíst, že při změnách parametru G pro 2 tis. dokumentů byly vygenerovány stejné výsledky. Při tomto množství byl vygenerován pouze jeden uzel napříč všemi parametry. Pro 10 tis. dokumentů jsou dosahovány lepší výsledky. Nejvyšší počet uzlů je vygenerován pro hodnoty `G10, G80, G100`. Pro 50 tis. dokumentů je nejvyšších hodnot dosaženo pro hodnoty `G20, G40`.



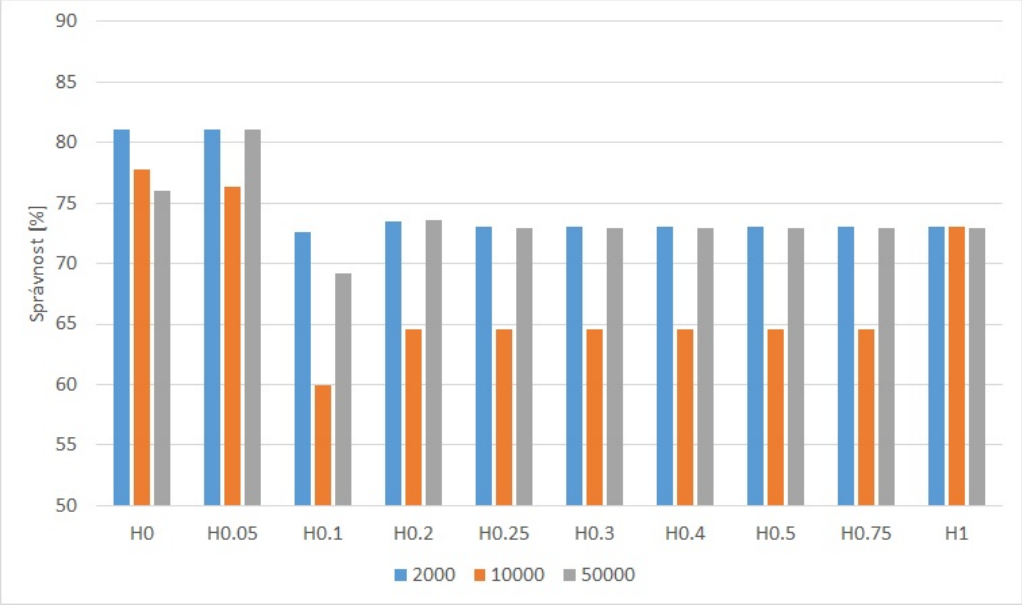
Obr. 15: Srovnání poměru správnosti/doba zpracování pro GracePeriod (TP)



Obr. 16: Normalizovaný počet uzlů stromu pro GracePeriod

TieThreshold

Další neméně důležitým parametrem se ukázal parametr *TieThreshold*. Více potřebných informací o závislostech lze vyčíst z následujících několika grafů.

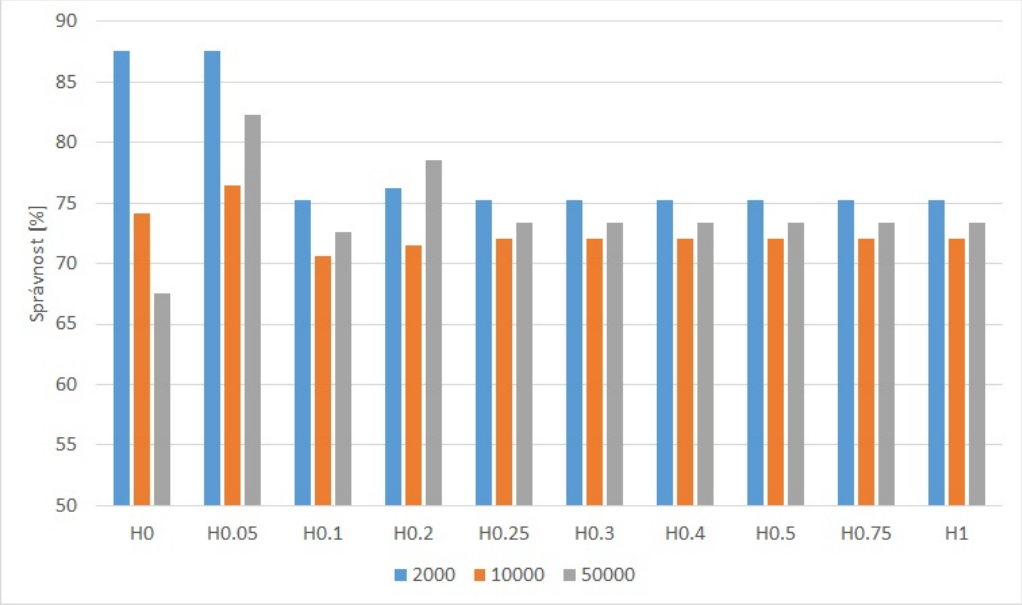


Obr. 17: Srovnání správnosti parametru TieThreshold (TF-IDF)

Pro hodnoty parametru `H` z Obr. 17 s ohledem na správnost lze konstatovat, že nejvyšších hodnot správnosti je dosahováno při hodnotách `H0` a `H0.05`. Naopak nejnižších hodnot je dosahováno pro `H0.1`, kde je správnost na nejnižší úrovni. Pro 2 tis. dokumentů je 72 % správnosti, pro 10 tis. dokumentů je dosaženo pouhých 60 % správnosti. Pro 50 tis. dokumentů je dosaženo necelých 70 % správnosti. V rozmezí hodnot parametru `<H0.2;H0.75>` bylo poté dosaženo stejné úrovně hodnot.

Srovnání s váhou termu (TP) na Obr. 18 ukazuje podobný průběh správnosti. Nejvyšších hodnot správnosti je dosahováno pro `H0` a `H0.05`. Pro 50 tis. dokumentů však významnosti dosahuje pouze nastavení parametru `H0.05`, kdy je správnost více jak 80 %. Ostatní hodnoty parametru vykazují pro ostatní obměny vyrovnané hodnoty, které se příliš od sebe neodlišují. Dosahují ale nižších hodnot kolem 5 % správnosti, což již může do jisté míry ovlivnit kvalitu klasifikace.

Pro každé zvolené množství dokumentů je následně zvoleno srovnání statistických metrik v Tab. 14 a 15. Z následujících tabulek je vidět, že minimální hodnoty jsou přibližně 60 % správnosti. Tyto hodnoty odpovídají nastavení hodnoty parametru na `H0.1`. Průměrná správnost se pohybuje v rozmezí 68–78 % správnosti. Váha termu (TP) se ukazuje jako průměrně lepší ve většině statistických charakteristikách. Rozdíly dosahují však jen jednotek procent správnosti. Zbylé charakteristiky lze vyvodit z daných tabulek.



Obr. 18: Srovnání správnosti parametru TieThreshold (TP)

Tab. 14: Statistické metriky srovnání správnosti pro TieThreshold (TF-IDF)

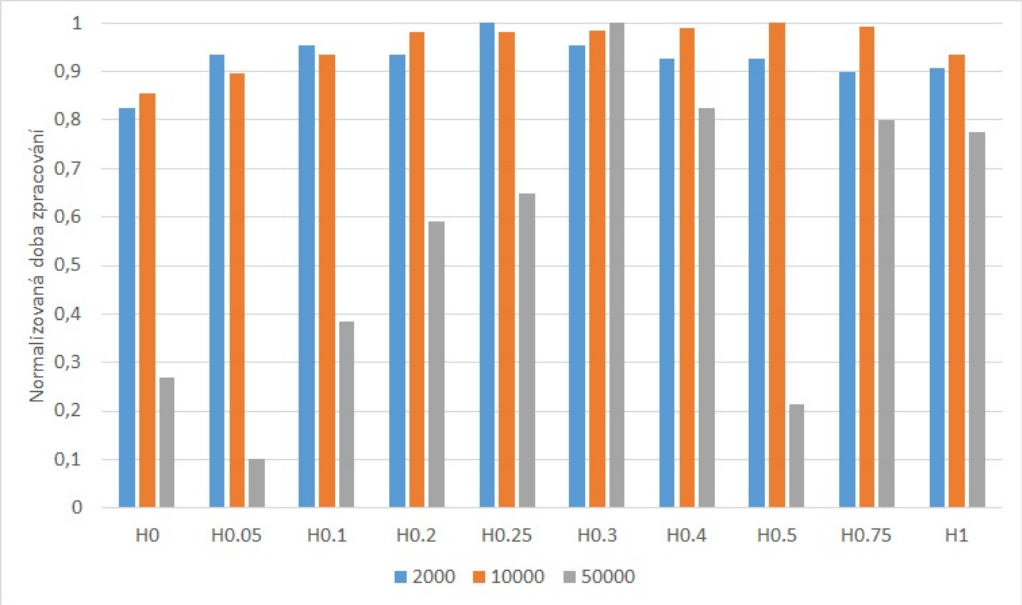
hodnota	2 000	10 000	50 000
MAX	81,06	77,77	81,03
MIN	72,62	59,93	69,17
Průměr	74,62	67,47	73,71
MED	73,00	64,60	72,88

Tab. 15: Statistické metriky srovnání správnosti pro TieThreshold (TP)

hodnota	2 000	10 000	50 000
MAX	87,60	76,48	82,29
MIN	75,21	70,66	67,53
Průměr	77,79	72,50	74,11
MED	75,21	72,04	73,37

Normalizace doby zpracování

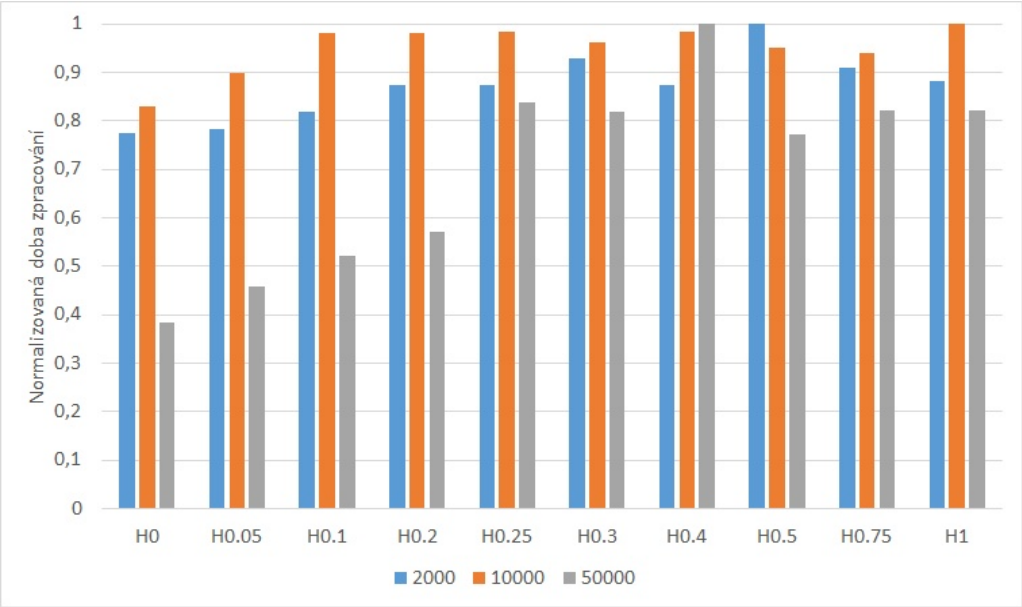
V následujícím kroku je stejně jako pro předchozí parametr provedena normalizace doby zpracování. Z Obr. 19 je ihned patrné, že nastavení parametru na hodnotu `H0.3` je ve všech srovnávaných datových obměnách zpracováno v nejdelším časovém úseku. Tato nastavení se tedy zdá být jako nejméně vhodné s ohledem na dobu zpracování. Z grafického vyjádření lze odvodit, že hodnoty `H0.05` a `H0.5` jsou pro 50 tis. dokumentů nejlépe zvolenými parametry. Pro nižší množství dat (2 a 10 tis.) je nejkratší doba zpracování naměřena pro hodnotu `GO` daného parametru. Pro 2 tis. dokumentů se ukazuje s ohledem na dobu zpracování nejméně vhodná volba parametru `H0.25`, kdy je dosaženo nejdelší doby zpracování.



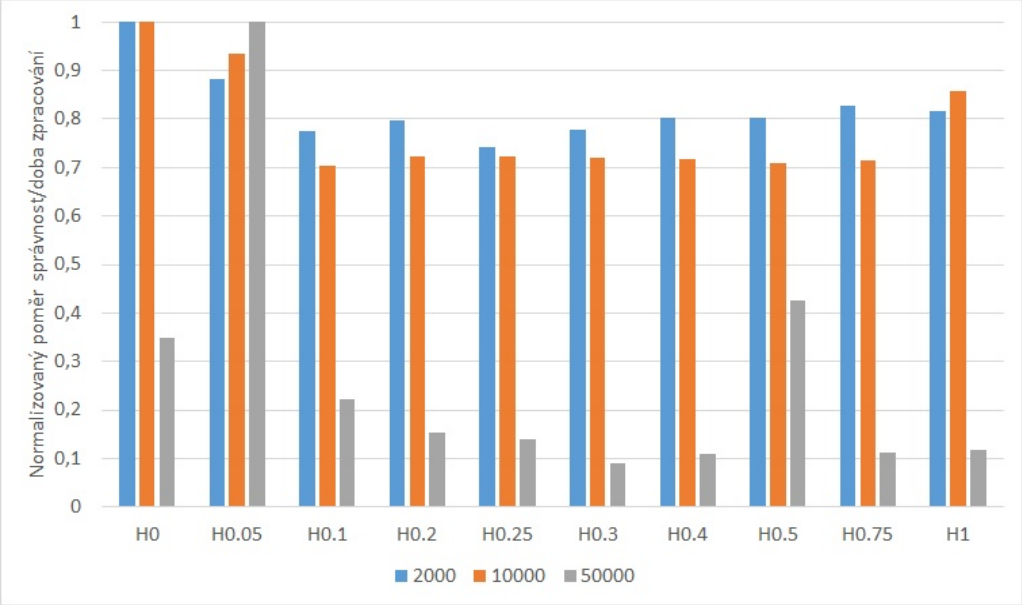
Obr. 19: Normalizovaná doba zpracování pro TieThreshold (TF-IDF)

Z Obr. 20 je znatelné, že výsledky dosahují pro nižší objem dat (2 a 10 tis.) podobného průběhu. Větší rozdíly se projevují při vyšším objemu dat. Pro objem 50 tis. dokumentů je nejdelší doby zpracování dosaženo pro nastavení parametru na `H0.4`. Na druhé straně je poté nejkratší doby zpracování dosahováno pro 50 tis. dokumentů pro hodnotu parametru `H0.05`.

Následně je provedeno srovnání poměru správnost/doba zpracování v normalizované podobě vzhledem k objemu dat. Z Obr. 21 vyplývá, že pro všechna vybraná množství dokumentů je nejvhodnější nastavení parametru na hodnotu `H0.05`. Při objemech 2 a 10 tis. dokumentů nejsou změny daného poměru patrné.



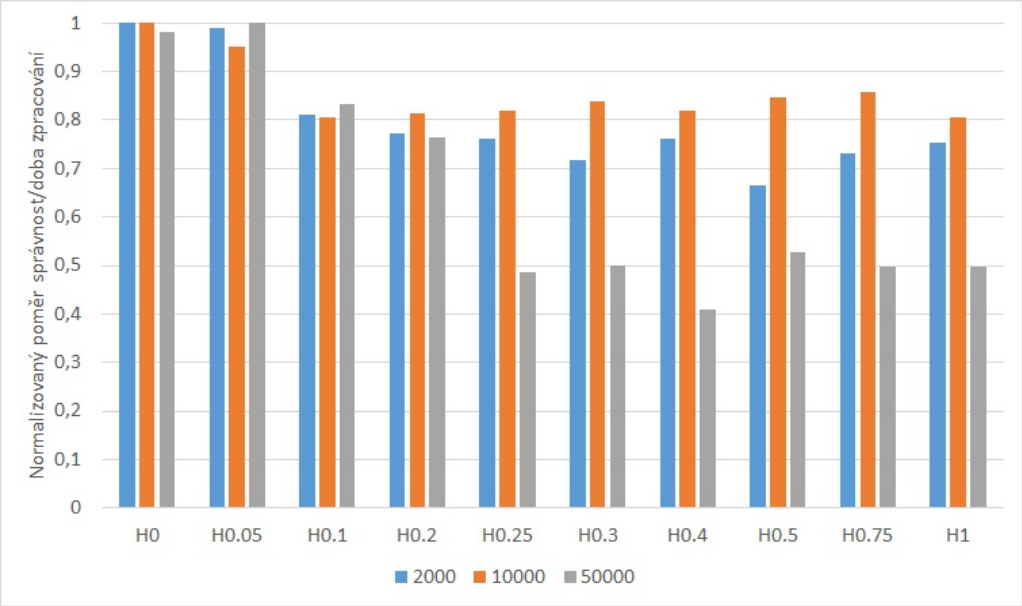
Obr. 20: Normalizovaná doba zpracování pro TieThreshold (TP)



Obr. 21: Srovnání poměru správnosti/doba zpracování pro TieThreshold (TF-IDF)

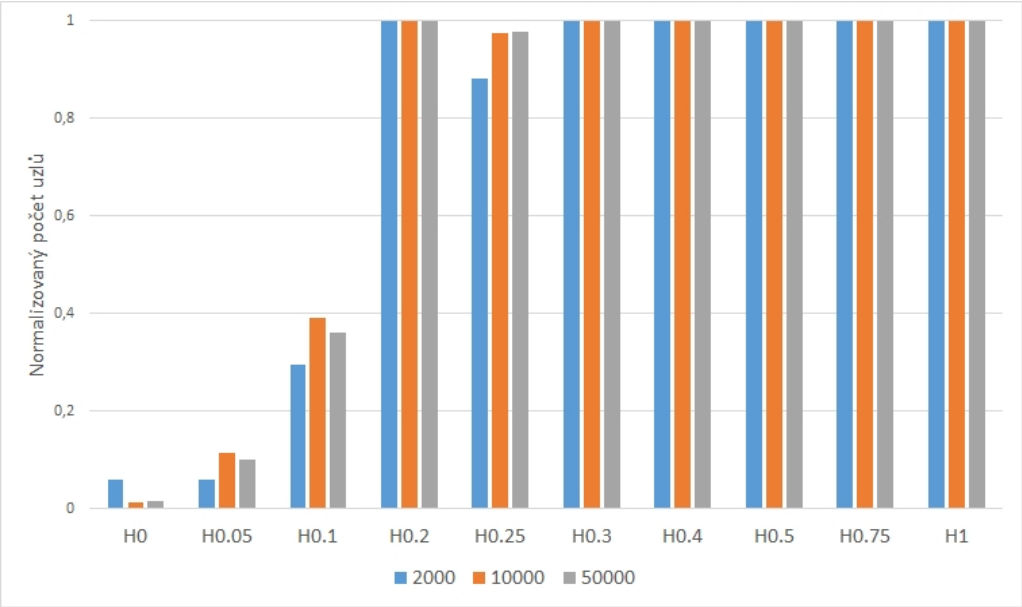
Z Obr. 22 pro váhu termu (TP) lze odvodit, že pro normalizovaný poměr správnosti a doby zpracování vychází nejvyšší hodnoty pro `H0` a `H0.05`, kdy je dosaženo nejvyššího poměru mezi naměřenými hodnotami.

Pro tento graf lze odvodit podobné závěry jako pro váhu termu (TF-IDF). Pro objemy 2 a 10 tis. dokumentů není rozdíl výrazný mezi daným poměrem. Pro nižší objem dat nedosahuje poměr správnosti a doby zpracování velkých rozdílu. S růstem hodnoty parametru dochází ke snižování daného poměru. Výrazné změny si lze všimnout až pro objem 50 tis. dokumentů, kdy pro hodnotu parametru `H0.4` je naměřeno nejnižší hodnoty poměru. Tato hodnota se jeví jako nejméně vhodná z hlediska tohoto poměru.



Obr. 22: Srovnání poměru správnosti/doba zpracování pro TieThreshold (TP)

Srovnání lze uzavřít počtem vygenerovaných uzlů pro dané obměny parametrů TieThresHold. Počet uzlů stromu se napříč srovnávanými vahami termu neliší. Lze tedy dané experimenty zobecnit v podobě normalizovaného grafu na Obr. 23. Pro 2 tis. dokumentů je naměřeno nejvyššího počtu uzlů pro hodnoty parametru `<H0.25;H1>`, kdy nedochází ke změnám v počtu uzlů. Z grafu lze vypozorovat, že v rozmezí hodnot `<H0;H0.1>` je malý počet uzlů. Následně je vidět jak pro hodnotu `H0.2` dochází k prudkému nárůstu hodnot. Při dosažení hodnoty `H0.25` je z průběhu grafu patrné, že již nedochází k růstu počtu uzlů a počet uzlů se stává konstantním. Tento poznatek lze zobecnit pro všechny obměny množství dokumentů.

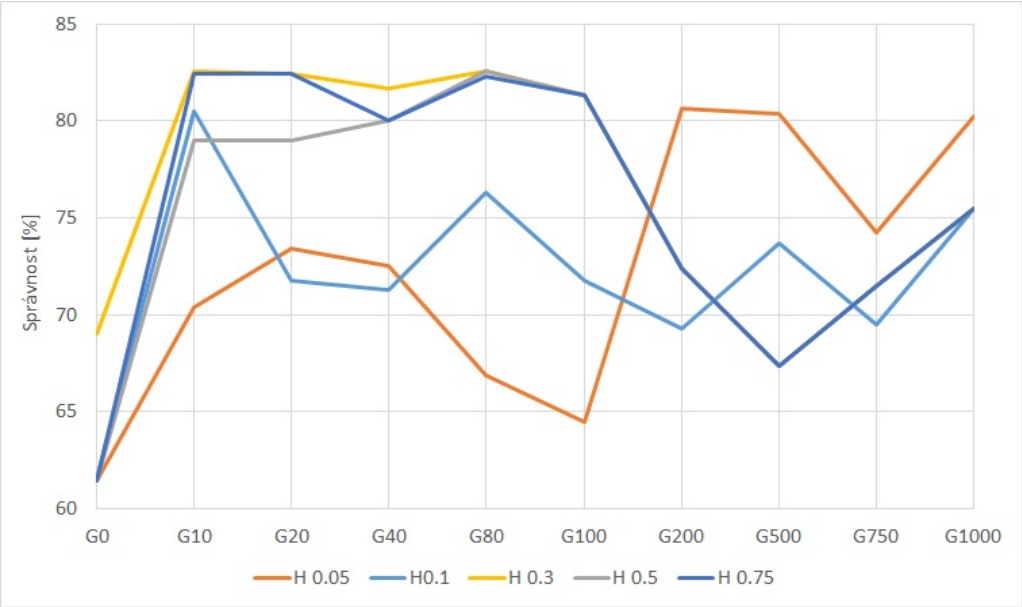


Obr. 23: Normalizovaný počet uzlů stromu pro TieThreshold

TieThreshold a GracePeriod

Při experimentování bylo zjištěno, že by bylo zajímavé zkoumat i závislost parametrů mezi sebou. Bylo tedy provedeno experimentování napříč parametry TieThreshold a GracePeriod. Pro přehlednost je vybrána váha termu (TF-IDF) pro 50 tis. recenzí, kde jsou nejlépe vidět naměřené rozdíly.

Pro přehlednost je v následujícím Obr. 24 vybráno pro vyjádření pro všechny hodnoty `<G0;G1000>` reprezentativních pět hodnot `H` tzn. `H0.05, H0.1, H0.3, H0.5, H0.75`, které charakterizují průběh experimentování. Z grafu je patrné, že kombinování hodnoty `H0.05, H0.1` s hodnotou parametru `G` dosahuje nízké správnosti. Zajímavé je pozorovat, že při hodnotě `G80` bylo dosaženo pro křivky `H0.3, H0.5, H0.75` stejné hodnoty správnosti. V této hodnotě lze konstatovat, že je dosaženo nejvyšší hodnoty a to 82 % správnosti. Dle autorova názoru je to zajímavý výsledek dokazující, že různé obměny parametrů mají také do značné míry vliv na celkovou správnost klasifikace.



Obr. 24: Závislost obměn parametrů TieThreshold a GracePeriod

Použitý klasifikátor pro listy

Následující Tab. 16, 17 naznačují rozdíl při použití klasifikátorů pro listy. *Naive Bayes Adaptive* vykazuje nejlepší výsledky s ohledem na správnost klasifikace a počet vygenerovaných listů a uzlů daného stromu. Na druhé straně se neadaptivní Naivní Bayes ukazuje jako nevhodný pro použití. Jeho správnost je pouze 65 % pro (TF-IDF) a pro (TP) dosahuje dokonce správnosti jen 61,8 %.

Tab. 16: Srovnání použitých klasifikátorů pro listy (TF-IDF, 50 tis. dokumentů)

Algoritmus	Klasifikátor	Váha	Doba zpracování	Správnost	Počet uzlů	Počet listů
HoeffdingTree	Majority Class	TF	297,37	73,07	67	34
HoeffdingTree	Naive Bayes	TF	604,93	65,65	67	34
HoeffdingTree	Naive Bayes Adaptive	TF	348,04	74,15	67	34

Tab. 17: Srovnání použitých klasifikátorů pro listy (TP, 50 tis. dokumentů)

Algoritmus	Klasifikátor	Váha	Doba zpracování	Správnost	Počet uzlů	Počet listů
HoeffdingTree	Majority Class	TP	352,23	74,39	61	31
HoeffdingTree	Naive Bayes	TP	600,49	61,83	61	31
HoeffdingTree	Naive Bayes Adaptive	TP	455,12	74,70	69	35

J48 – parametrizace

Algoritmus J48 poskytuje pouze dva důležité parametry, které lze nastavit pro potřeby klasifikace. Prvním parametrem je míra prořezání stromu – `C`. Druhým parametrem je minimální počet instancí v listu – `M`.

Dle provedených experimentů nemá druhý parametr podstatný vliv na průběh klasifikace. Vliv míry prořezání stromu (`C`) je charakterizován pomocí následujících Tab. 18, 19. Pro `C = 0`, nebyl vygenerován žádný uzel. Výsledky tohoto pokusu byly proto dále eliminovány. Problém nastal dále při nastavění parametru na hodnotu `C = 1`. Při tomto nastavení proběhlo chybové hlášení: `WARNING: confidence value for pruning too high. Error estimate not modified.` Varování má zcela logické opodstatnění. Nelze stoprocentně prořezávat daná data – tzn. se 100% spolehlivostí. O této problematice je podrobněji diskutováno v (Oshea, 2014). Podle tohoto zjištění lze usoudit, že už hodnota vyšší než 0.5 nedává smysl používat při klasifikování.

Výsledky v Tab. 18 a 19 dané zjištění potvrzují, že pro hodnotu $C \geq 0.5$ nedochází k růstu stromu a správnost klasifikace klesá. Při hodnotě $C = 0.5$ je dosahováno maximálního počtu uzlů (listů) pro vyšší hodnotu už počet neroste a zůstává konstantní.

Původní hodnota $C = 0.25$ je do značné míry stanovena jako nejvhodnější hodnota. O vyšší ani nižší hodnotě nemá význam příliš uvažovat. Rozdíl ve správnosti je pouze o jednotku procenta. Dle názoru autora nemá velký smysl nad změnami parametrizace u algoritmu J48 přemýšlet.

Tab. 18: Změna charakteristik algoritmu J48 při změně parametru míra prořezání stromu – C (TF-IDF, 50 tis. dokumentů)

Parametr	váha termu	Počet uzlů	Počet listů	Správnost
C0.1	TF-IDF	75	38	78,18
C0.25	TF-IDF	113	57	79,61
C0.5	TF-IDF	179	90	78,29
C0.75	TF-IDF	179	90	77,80

Tab. 19: Změna charakteristik algoritmu J48 při změně parametru míra prořezání stromu – C (TP, 50 tis. dokumentů)

Parametr	váha termu	Počet uzlů	Počet listů	Správnost
C0.1	TP	83	42	81,48
C0.25	TP	121	61	80,57
C0.5	TP	169	85	78,96
C0.75	TP	169	85	78,47

V Tab. 20 je ukázána rozdílnost napříč váhami termu (TF-IDF) a (TP). Rozdíl je v řádu jednotek uzlů/listů. Rozdíl ve správnosti opět dosahuje minimálních hodnot. Větším rozdílem je nastavení parametru pro `C0.1`, kde je vidět rozdíl ve správnosti přibližně 3 %.

Tab. 20: Srovnání rozdílu mezi stanovenými váhami termu pro algoritmus J48

Parametr	Počet uzlů	Počet listů	Správnost
C0.1	-8	-4	-3,29
C0.25	-8	-4	-0,96
C0.5	10	5	-0,67
C0.75	10	5	-0,67

4.3 Určení významných slov

Na závěr je vhodné uvést celkový seznam slov vygenerovaných danými algoritmy. Jelikož Weka, ani MOA v reportu s výsledky nevypisují na konci přesný seznam nejvýznamnějších slov bylo nutné provést výběr nejdůležitějších slov za pomocí regulárního výrazu.Ukázka stromové struktury viz příloha B. Dále se zaměříme na porovnání vygenerovaných významných slov sledovanými algoritmy při stejném počtu dokumentů. Algoritmus J48 poskytuje velice podrobnou stromovou strukturu, která není dobře zobrazitelná v plném rozsahu. S ohledem na rozsah stromové struktury je pro interpretaci výsledků zvolen výběr prvních 30 nejvýznamnějších slov, na kterých lze přehledně demonstrovat podobnost mezi danými algoritmy. Srovnání je provedeno na úrovni porovnání algoritmu J48 s obměnami Hoeffdingova stromu. Přehledně jsou podobnosti zaznamenány v Tab. 21, kde jsou vyznačena shodná slova.

Ve shodě s následující tabulkou významných slov lze dojít k vyvození významnosti slov. Algoritmus J48 se shoduje s obměnami HT ve slovech [**LOCATION, FRIENDLY, EXCELLENT, HELPFUL, SPACIOUS,DELICIOUS, NICE, CONVENIENT, EXCELENT, BEAUTIFUL, COMFORTABLE, CONVENIENCE, HELPFULL, EASY, ENJOYED**]. Hoeffdingův strom spolu s Hoeffdingovým alternativním stromem dokázali vygenerovat stejnou posloupnost slov. S algoritmem J48 se shodují v 10 slovech z 30, což vytváří podíl 33 %. Jedná se o slova [**HELPFUL, LOCATION, EXCELLENT, NICE, FRIENDLY, BEAUTIFUL, EASY, DELICIOUS, COMFORTABLE, CONVENIENT**]. Dá se konstatovat, že se jedná vesměs o kladně hodnotící slova. Částečně rozdílná slova dokázal při srovnání vygenerovat Hoeffdingův adaptivní strom. Jedná se o slova [**HELPFUL, LOCATION, FRIENDLY, EXCELLENT, NICE, HELPFULL, COMFORTABLE, EASY, CONVENIENT, ENJOYED**]. Shoda je opět 10 slov a s tím souvisí podíl 33 %. Mezi slovy, vygenerovanými danými algoritmy jsou zastoupena i jiná slova vesměs pozitivního hodnocení. Výjimku tvoří slova [**POOR, BED**], jenž lze považovat za negativní slova. Neutrální povahy poté nabývají slova jako [**NOT, ONLY**], které mohou nabývat jak pozitivního hodnocení, tak i negativního.

Pro váhu termu (TP) dochází k několika změnám ve shodě slov. Je docíleno vyšší shody mezi analyzovanými algoritmy. Jak je vidět z následující Tab. 22 algoritmus J48 se shoduje s obměnami HT ve slovech [**LOCATION, QUIET, HELPFUL, EXCELLENT, NICE, GOOD, GREAT, CLEAN, HAD, SOME, STAFF, BREAKFAST, VISIT, EASY**]. Pro algoritmus HT stejně jako jeho obměnu Hoeffdingův alternativní strom, je docílena shoda v celkem 12 slovech, což vytváří podíl 40 % na analyzovaném výběru nejvýznamnějších slov. Jedná se o slova [**HELPFUL, STAFF, CLOSE, LOCATION, GOOD, CLEAN, GREAT, EASY, NICE, VISIT, EXCELLENT, BREAKFAST**]. Hoeffdingův adaptivní strom opět vybral za významná některá jiná slova. Jedná se o slova [**HELPFUL, HAD, SOME, GOOD, NICE, LOCATION, EXCELLENT, CLEAN, QUIET, FRIENDLY, STAFF**].

Tab. 21: Srovnání 30 nejvýznamnějších slov (TF-IDF, 50 tis. dokumentů)

J48	HT	alternativní HT	adaptivní HT
POOR	HELPFUL	HELPFUL	HELPFUL
LOCATION	NOT	NOT	LOCATION
FRIENDLY	ONLY	ONLY	NOT
EXCELLENT	STAFF	STAFF	FRIENDLY
HELPFUL	BED	BED	CLOSE
WONDERFUL	HAVE	HAVE	HAVE
FRIENDLINESS	LOCATION	LOCATION	GREAT
SPACIOUS	CLEAN	CLEAN	EXCELLENT
DELICIOUS	WITH	WITH	CLEAN
NICE	CLOSE	CLOSE	GOOD
PROXIMITY	WELL	WELL	POOL
CONVENIENT	GOOD	GOOD	POLITE
OUTSTANDING	EXCELLENT	EXCELLENT	WELL
EXCELENT	RECOMMEND	RECOMMEND	NICE
BEAUTIFUL	NICE	NICE	WITH
AMAZING	SWIMMING	SWIMMING	MOST
CONVENIENTLY	FRIENDLY	FRIENDLY	CATERING
COURTEOUS	GREAT	GREAT	PASSPORT
CENTRALLY	LIKED	LIKED	SMALL
FANTASTIC	THERE	THERE	JUST
COMFORTABLE	BEAUTIFUL	BEAUTIFUL	FREE
HEART	EASY	EASY	OWNER
POSITION	AGAIN	AGAIN	EVEN
LOVELY	NEAR	NEAR	HELPFULL
SHOPPING	QUIET	QUIET	COMFORTABLE
PEACEFUL	DELICIOUS	DELICIOUS	STAFF
CONVENIENCE	LOVED	LOVED	EASY
HELPFULL	COMFORTABLE	COMFORTABLE	LIKE
EASY	MUCH	MUCH	CONVENIENT
ENJOYED	CONVENIENT	CONVENIENT	ENJOYED
SHODA	10	10	10
PODÍL	0,33	0,33	0,33

Tab. 22: Srovnání 30 nejvýznamnějších slov (TP, 50 tis. dokumentů)

J48	HT	alternativní HT	adaptivní HT
LOCATION	HELPFUL	HELPFUL	HELPFUL
FRIENDLY	NOT	NOT	HAD
QUIET	ONLY	ONLY	OTHER
HELPFUL	HELPFUL	STAFF	SOME
COULD	CLOSE	CLOSE	GOOD
EXCELLENT	DAY	DAY	NICE
NICE	LOCATION	LOCATION	STAY
GOOD	COMFORTABLE	COMFORTABLE	BUT
GREAT	GOOD	GOOD	LOCATION
THERE	CLEAN	CLEAN	HAVE
THEY	SMALL	SMALL	NOT
CLEAN	GREAT	GREAT	EXCELLENT
TOO	OTHER	OTHER	CLEAN
HAD	EASY	EASY	CONVENIENT
SOME	ROOM	ROOM	AIRPORT
QUITE	CITY	CITY	QUIET
STAFF	NICE	NICE	SHOPPING
ROOMS	SERVICE	SERVICE	CLOSE
BREAKFAST	VISIT	VISIT	WERE
AND	SETTING	SETTING	FRIENDLY
THE	JUST	JUST	THEATRE
CLOSE	ENJOYED	ENJOYED	CONVENIENCE
LIKE	FERRY	FERRY	WELL
VISIT	HAVE	HAVE	COMFORTABLE
YOUR	EXCELLENT	EXCELLENT	PERFECT
EASY	BREAKFAST	BREAKFAST	KITCHEN
TRAIN	LOVELY	LOVELY	STAFF
BED	FREE	FREE	CITY
BIT	WALKS	WALKS	VENUE
BATH	AIRPORT	AIRPORT	FAMILY
SHODA	12	12	12
PODÍL	0,4	0,4	0,4

Pro doplnění srovnání jsou dále určena slova pro množství 10 tis. a 2 tis. dokumentů, kde je zaměřeno již na výběr pouze 10 významných slov. Z následujících tabulek je vidět, že základ stromové struktury se při nižším množství dat nijak nemění. Dochází jen ke změnám v pořadí významných slov viz Tab. 23, 24, 25, 26.

Tab. 23: Srovnání 10 nejvýznamnějších slov (TF-IDF, 10 tis. dokumentů)

pozice	J48	HT	alternativní HT	adaptivní HT
1	LOCATION	HELPFUL	HELPFUL	HELPFUL
2	FRIENDLY	NOT	NOT	HAD
3	QUIET	ONLY	ONLY	OTHER
4	HELPFUL	STAFF	STAFF	SOME
5	COULD	CLOSE	CLOSE	GOOD
6	EXCELLENT	DAY	DAY	NICE
7	NICE	LOCATION	LOCATION	STAY
8	GOOD	COMFORTABLE	COMFORTABLE	BUT
9	GREAT	GOOD	GOOD	LOCATION
10	THERE	CLEAN	CLEAN	HAVE
	SUMA	3	3	4
	PODÍL	0,3	0,3	0,4

Tab. 24: Srovnání 10 nejvýznamnějších slov (TP, 10 tis. dokumentů)

pozice	J48	HT	alternativní HT	adaptivní HT
1	LOCATION	HELPFUL	HELPFUL	HELPFUL
2	FRIENDLY	NOT	NOT	HAD
3	QUIET	ONLY	ONLY	OTHER
4	HELPFUL	STAFF	STAFF	SOME
5	COULD	CLOSE	CLOSE	GOOD
6	EXCELLENT	DAY	DAY	NICE
7	NICE	LOCATION	LOCATION	STAY
8	GOOD	COMFORTABLE	COMFORTABLE	BUT
9	GREAT	GOOD	GOOD	LOCATION
10	THERE	CLEAN	CLEAN	HAVE
	SUMA	3	3	4
	PODÍL	0,3	0,3	0,4

Tab. 25: Srovnání 10 nejvýznamnějších slov (TF-IDF, 2 tis. dokumentů)

pozice	J48	HT	alternativní HT	adaptivní HT
1	LOCATION	HELPFUL	HELPFUL	HELPFUL
2	FRIENDLY	NOT	NOT	HAD
3	QUIET	ONLY	ONLY	OTHER
4	HELPFUL	STAFF	STAFF	SOME
5	COULD	CLOSE	CLOSE	GOOD
6	EXCELLENT	DAY	DAY	NICE
7	NICE	LOCATION	LOCATION	STAY
8	GOOD	COMFORTABLE	COMFORTABLE	BUT
9	GREAT	GOOD	GOOD	LOCATION
10	THERE	CLEAN	CLEAN	HAVE
	SUMA	3	3	4
	PODÍL	0,3	0,3	0,4

Tab. 26: Srovnání 10 nejvýznamnějších slov (TP, 2 tis. dokumentů)

pozice	J48	HT	alternativní HT	adaptivní HT
1	LOCATION	HELPFUL	HELPFUL	HELPFUL
2	FRIENDLY	NOT	NOT	HAD
3	QUIET	ONLY	ONLY	OTHER
4	HELPFUL	STAFF	STAFF	SOME
5	COULD	CLOSE	CLOSE	GOOD
6	EXCELLENT	DAY	DAY	NICE
7	NICE	LOCATION	LOCATION	STAY
8	GOOD	COMFORTABLE	COMFORTABLE	BUT
9	GREAT	GOOD	GOOD	LOCATION
10	THERE	CLEAN	CLEAN	HAVE
	SUMA	3	3	4
	PODÍL	0,3	0,3	0,4

5 Diskuze

V diplomové práci byly prozkoumány možnosti inkrementálního a dávkového strojového učení na rozsáhlých kolekcích textových dokumentů, které jsou pro své vlastnosti časově a prostorově velmi náročné na zpracování. V následujících sekcích se zaměříme na faktory, které měly podstatný vliv na získané výsledky.

5.1 Vliv metody výpočtu váhy termů

Pro algoritmus J48 se ukázalo, že na správnost a počet vygenerovaných uzlů stromu nemá výběr metody výpočtu váhy termu velký vliv (rozdíl je velice malý). Pro obměny HT tato skutečnost tak jednoznačná není. Pro nižší množství dat (2 a 5 tis. dokumentů) se ukázala metoda výpočtu váhy (TP) jako vhodnější. Bylo zde dosaženo vyšší správnosti a to ve všech obměnách. Pro vyšší množství dat se ukázalo, že použití metody výpočtu vah termů nemá již tak podstatný vliv – rozdíly správnosti jsou zanedbatelné.

5.2 Vliv velikosti objemu zpracovávaných dat

Pro objem zpracovávaných dat lze vyvodit obecný závěr, že u algoritmus J48 nedochází k výraznému růstu správnosti klasifikace – správnost roste pozvolně. Pro obměny HT lze odvodit lineární průběh růstu správnosti v závislosti na objemu dat. Pro nižší množství dat se správnost algoritmů pohybuje okolo 70 %. Pro vyšší objemy (25 a 50 tis. dokumentů) je pak správnost okolo 80 %, což lze považovat za přijatelnou úroveň správnosti. Z výsledků lze vyvodit závěr, že s objemem počtu instancí jak pro algoritmus J48, tak pro obměny HT, je patrný růst počtu uzlů/listů stromové struktury.

5.3 Změna parametrů zvolených algoritmů

Pro algoritmus J48 lze vyjít z ověřených experimentů, které ukazují, že při změně koeficientu prořezání (`C`) na hodnotu $\texttt{C} \geq \texttt{0.5}$ dochází k přílišnému prořezání stromu. Počet uzlů již pro algoritmus J48 neroste. Tento poznatek byl ověřen i následným experimentováním s tímto parametrem.

Pro Hoeffdingův strom byla nalezena závislost na třech významných parametrech. Na problematiku parametrizace se lze podívat z různých úhlů pohledu. Hoeffdingův strom přichází s novým konceptem učení z toku dat. Dle výsledků se ukazuje, že tento algoritmus je velice „citlivý“ na zvolené parametry. Z provedených experimentů lze konstatovat, že klasifikace je zde především ovlivněna trojicí parametrů *GracePeriod*, *TieThreshold*, volba klasifikátoru pro listy.

GracePeriod

Parametr **GracePeriod** poskytuje vyrovnané hodnoty správnosti pro nastavení parametru větší než `G200`. Nastavení hodnoty parametru na `G0` vede k nízké správnosti jak ukazují výsledky i k přeučení daného algoritmu. Pro různá množství dat se značně liší vhodné hodnoty parametrů, přičemž z naměřených výsledků lze vyvodit určitá doporučení.

Při použití váhy termu (TF-IDF) se z hlediska správnosti ukázalo pro 50. tis. recenzí nejvhodnější původní nastavení na hodnotu G200, pro nižší objem 2. tis. rozsah `<G100;G1000>` a pro 10 tis. `G20`.

Při použití váhy termu (TP) byl pro 50 tis. nejvhodnější rozsah `<G200;G1000>`, pro 2 tis. rozsah `<G80;G1000>`, pro 10 tis. `<G80;G100>`.

Změna daného parametru má zcela opodstatněný vliv i na dobu zpracování (budeme-li ignorovat hodnotu `G0`, jenž je irelevantní a nemá žádnou vypovídací hodnotu).

Pro dobu zpracování lze ohledně vhodného nastavení vyvodit následující závěry. Pro váhu termu (TF-IDF): pro 2 tis. a 10 tis. dokumentů rozsah `<G500;G1000>`, pro 50 tis. dokumentů hodnota `G10`. Pro váhu termu (TP): pro 2 tis. dokumentů: rozsah `<G200;G1000>`, pro 10 tis. dokumentů: rozsah `<G500;G1000>`; pro 50 tis. dokumentů: rozsah `<G500;G1000>`.

Nakonec bude srovnán normalizovaný poměr správnost/doba zpracování. Dle provedených měření lze vyvodit vhodné hodnoty parametru následovně: pro váhu termu (TF-IDF) pro 50 tis. dokumentů: `G10`, 2 a 10 tis. dokumentů: rozsah `<G500;G1000>`. Při použití váhy termu (TP): pro 50 tis. dokumentů (při opomenutí hodnoty `G0`) vychází nejlépe stejně jako pro 2 a 10 tis. rozsah: `<G500;G1000>`.

TieThreshold

Pro parametr TieThreshold lze vyvodit následující doporučení při použití váhy termu (TF-IDF) je pro 50 tis. dokumentů vhodná hodnota `H0.05`. Pro 2 a 10 tis. dokumentů je doba zpracování přibližně stejná. Stejné závěry platí pro váhu termu (TP).

Následně lze provést srovnání normalizovaného poměru správnost/doba zpracování. Pro váhu termu (TF-IDF) je nejvhodnější pro 50 tis. dokumentů nejvhodnější hodnota `H0.05` pro 2, 10 i 50 tis. dokumentů. Stejné výsledky jsou neměřeny pro váhu termu (TP).

Klasifikátor pro listy

Zhodnocení parametrů lze uzavřít hodnocením klasifikátoru pro listy. Naměřené výsledky ukazují, že z hlediska správnosti klasifikace nejsou rozdíly pro různé klasifikátory příliš patrné. Projevuje se zde však rozdíl z hlediska doby zpracování. Pro 50 tis. dokumentů je naměřena dvojnásobná doba zpracování při výběru Naivního Bayese oproti klasifikaci na základě Většinové třídy. Navíc Naivní Bayes zde dosahuje nejnižší správnosti oproti srovnávaným obměnám.

S ohledem na správnost se tak dle výsledků jako nejvhodnější ukazuje Adaptivní Naivní Bayes. V případě časového hlediska se jeví jako nejlepší řešení Většinová třída (*Majority class*).

5.4 Možnost využití v praxi

Experimenty s použitými daty mohou mít význam především pro majitelé hotelů. Lze velice dobře zjistit, co zákazníky hotelů nejvíce zajímá. Hoteliéři tak mohou dle přání či kritiky zákazníků učinit rozhodnutí pro případné změny. Ukazuje se, že pro získání potřebných znalostí dostačuje běžné PC. Výsledky jsou generovány v delším časovém úseku a mohou mít zcela jasný význam pro hoteliéra, který na základě zjištěných významných slov může provést patřičné vylepšení či zjištěné znalosti využít pro lepší marketingovou propagaci.

Bylo ukázáno, že správnost výsledků získaných pomocí Hoeffdigova stromu je porovnatelná s těmi z algoritmu J48. Lze tedy uvažovat o použití této inkrementální metody k analýze velkého množství dat a to s požadavkem na co nejkratší dobu učení a následného vyhodnocení.

Příkladem mohou být data z internetových sociálních sítí (Twitter, Facebook), která by tak bylo možné v (téměř) reálném čase sledovat a vyhodnocovat (nalézat v nich znalosti). Šlo by to využít ke sledování názorů a nálady veřejnosti, což může být užitečné pro firmy, vlády či výzkumné organizace.

5.5 Možnosti rozšíření

Správnost experimentů je do značné míry ovlivněna zvolenými daty. Pro větší zobecnění závěrů by zcela bylo nutné provést mnohem více experimentů – např. na datech, která berou v úvahu časovou dimenzi. Zde se tak otevírá možné rozšíření či pokračovaní v experimentální činnosti. Výsledky klasifikace jsou ovlivněny i volbou klasifikátoru pro listy.

Problém, se kterým bylo nutné se potýkat během celého experimentování, bylo především uchování a zpracování souborů ve formě ARFF a to z důvodu jejich značné velikosti. Převodem 50 tis. záznamů byl vygenerován soubor o velikosti kolem 1 GB. Práce s takto velkým souborem způsobovala často velké zatížení PC. Při rozsáhlejším experimentování by bylo vhodnější použití formátů využívajících pro reprezentaci dat řídké vektory. Tímto by se snížila doba zpracování a zjednodušila manipulace se soubory.

6 Závěr

Diplomová práce byla zaměřena na dnes významnou oblast zpracování masivního objemu textových dat. Cílem této práce bylo prozkoumat přístup inkrementálního zpracování a provést komparaci s dávkovým přístupem aplikovaným na zvolená testovací a trénovací data. Byla uskutečněna série experimentů na datech hotelových recenzí, a tyto byly detailně prozkoumány. Během testování byly provedeny experimenty s různými obměnami dat. Snahou bylo zjistit, co a do jaké míry ovlivňuje výsledky klasifikace vzhledem ke zvoleným kritériím.

V úvodní části práce je čtenář seznámen s problematikou dolování znalostí z dat. Tato část se zaměřuje především na problematiku dolování z textových dat a jsou zde podrobně rozebrány jednotlivé řešené úlohy a reprezentace textových dokumentů. Následně byly popsány metody předzpracování textových dat, mající bezesporu vliv na výsledek klasifikace. Dále byly realizovány kroky vedoucí k naplnění hlavního cíle, tedy srovnání vybraných klasifikačních algoritmů a vlivu hodnot parametrů na výsledky a průběh klasifikace. Bylo zde nutné pochopit principy daných algoritmů a nastudování programového vybavení, na kterém bylo realizováno provádění experimentů. Nejdříve bylo nutné vybrat vhodné algoritmy a programové vybavení a následně pochopit jejich princip resp. funkčnost.

Experimentální činností se podařilo porovnat vybrané algoritmy strojového učení – algoritmus J48 a Hoeffdingův strom vč. jeho obměn – z řady hledisek. Zaměřili jsme se především na srovnání správnosti klasifikace, dobu zpracování, změny v parametrizaci. Výsledky analýzy jsou statisticky zpracovány za pomocí aplikace MS Excel. Výstupy prvotního experimentování s rozsáhlými textovými daty vztahujícími se k hotelovým recenzím byly autorem práce úspěšně prezentovány na konferenci PEFnet 2015.

Provedenou experimentální činností lze dojít k názoru, že pro menší množství dat se Hoeffdingův strom nedokáže dostatečně natrénovat, proto jeho klasifikační výsledky dosahují nízké správnosti. Pro větší množství dat můžeme dojít k závěru, že Hoeffdingův strom dokáže vygenerovat rozhodovací strom, poskytující přibližně stejnou správnost jako srovnávaný algoritmus J48. Jak ukazují experimenty, algoritmus J48 potřebuje pro vyhodnocení takto velkého množství dat mnohem více času (konkrétně $4times$). Algoritmus Hoeffdingův alternativní strom zpracovával data vyšší dobu než ostatní srovnávané algoritmy, ale tyto výsledky jsou ovlivněny tím, že je nastavena základní volba $5\times$ znovu vybrat kořen pro rozhodování. Tím se časová složitost daného algoritmu zvyšuje.

7 Literatura

Abu-Mostafa, Y. S., Malik M.I., Lin H.T. *Learning from data: a short course.* AMLBook, 2012, xii, 201 s. ISBN 978-1600490064.

Acrea. *IBM Software Business Analytics.*[online]. 2016, [cit. 2016-04-15]. Dostupné na: <http://acrea.cz/assets/media/files/ibm-spss-modeler-17.pdf>.

Aguayo, R. *The Metaknowledge Advantage: The Key to Success in the New Economy.* 2. vyd. New York: Simon and Schuster, 2010, 304 s. ISBN 9781439138281.

Alpaydin, E. *Introduction to machine learning.* MIT press, 2014.

Atkinson, J. A. *Text Mining: Principles and applications.*Institute for Communicating and Collaborative Systems. School of Artifical Intelligence, University of Edinburgh, Edinburgh, Scotland, 2000.

Berka, P. *Dobývání znalostí z databází.*Praha, 2003. ISBN 80-200-1062-9.

Bhargava, N., Sharma, G., Bhargava, R., a Mathuria, M. *Decision tree analysis on j48 algorithm for data mining.* Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering, 2013.

Bifet, A. a Gavaldà, R. *Adaptive learning from evolving data streams.* In Advances in Intelligent Data Analysis VIII, Springer Berlin Heidelberg, 2009.

Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B. *Moa: Massive online analysis.* The Journal of Machine Learning Research, 2010.

Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B. *Data stream mining: Practical Approach.* The Journal of Machine Learning Research. [online]. 2011, [cit. 2016-04-12]. Dostupné na: <http://moa.cms.waikato.ac.nz>.

Brožová, H., Houška, M. *Modelování znalostí.* 1. vyd. Praha: Professional Publishing, 2011, s. 230-232, ISBN 978-80-7431-069-0.

Carela-Español, V., Barlet-Ros, P., Bifet, A., Fukuda, K. *A streaming flow-based technique for traffic classification applied to 12+ 1 years of Internet traffic.* Telecommunication Systems, 2015: 1-14.

Feldman, R., Sanger, J.*The text mining handbook: Advanced Approaches in Analyzing Unstructured Data.* New York: Cambridge, 2007. 423 s., ISBN 978-0-521-83657-9.

Gama, J., Sebastiano, R., Pereira, R. P. *Issues in evaluation of stream learning algorithms.* In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. s. 329-338.

Han, J., Micheline K. *Data mining: concepts and techniques.* Elsevier, 2011.

Hoeglinger, S., Pears, R. *Use of hoeffding trees in concept based data stream mining.* In: Information and Automation for Sustainability, 2007. ICIAFS 2007. Third International Conference on. IEEE, 2007. s. 57-62.

Holmes, G., Pfahringer, B., Kirkby, R. *New options for hoeffding trees.* In: AI 2007: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2007. s. 90-99.

Holmes, G., Pfahringer, B., Kirkby, R. B. *Mining Data Streams Using Option Trees.* Department of Computer Science, University of Waikato, s. 22-24, 2012.

Hulten G., et al. *Mining time-changing data streams. .* In: ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, s. 97-106, 2001.

Khan, S. S., Peer, M. A., Quadri, S. M. K. *Comparative study of streaming data mining techniques.* In Computing for Sustainable Global Development (INDIACom), 2014. s. 209-214. IEEE.

Koktan, M. *Automatické rozpoznávání (analýza) sentimentu.* Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

Kotiková, M. *Aplikace metod předzpracování při dolování znalostí z textových dat.* Brno, 2014. Diplomová práce. Mendelova univezita v Brně, Provozně ekonomická fakulta.

Krupník, J. *Automatizace generování stopslov.* Brno, 2014. Diplomová práce. Mendelova univezita v Brně, Provozně ekonomická fakulta.

Kullback, S. *Information theory and statistics.* Mineola, N.Y: Dover Publications, 2012, s. 416-418, ISBN 978-048-6142-043.

Mccallum, A., Nigam, K. *A Comparison of Event Models for Naive Bayes Text Classification.* In: Workshop on Learning for Text Categorization, 1998.

Miner, G. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications.* 1st ed. Waltham, MA: Academic Press, 2012, s. 1053-1055, ISSN 978-012-3869-791.

Netolický, P. *Massive Semantic Analysis of On-Line Text.* In PEFnet, 2015 European scientific conference of doctoral students, 47 s., 2015.

Novák, Z. *Aplikování paralelismu při dolování znalostí z textových dat.* Brno, 2013. Diplomová práce. Mendelova univezita v Brně, Provozně ekonomická fakulta.

Oshea, M. *Value of confidence and pruning in J48.* [online]. 2013, [cit. 2016-03-12]. Dostupné na: <http://weka.8497.n7.nabble.com/Value-of-confidence-and-pruning-in-J48-td30031.html>.

PARUCHURI, V. *Natural Language Processing Tutorial.* [online]. 2013, 26-06-2013 [cit. 2016-03-12]. Dostupné na: <http://vikparuchuri.com/blog/natural-language-processing-tutorial/>.

RASCHKA, S. *Naive Bayes and Text Classification – Introduction and Theory.* arXiv preprint arXiv:1410.5329, 2014.

SEBASTIANI, F. *Machine Learning in Automated Text categorization.* ACM Computing Surveys. New York: Association for Computing Machinery, 2002. s. 1-47.

SEDLÁČEK, P. *Text mining a jeho možnosti (aplikace).* Faculty of Informatics MU [online]. 2003, [cit. 2016-2-12]. Dostupné na: <http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xsedlac5.htm>.

SHMUELI, G., PATEL, N. R., BRUCE, P. C. *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Once Excel with XLMiner.* Hoboken, N.J.: Wiley, 2011. s. 404-405. ISBN 978-047-0526-828.

SRIVASTAVA, A., SAHAMI, M. *Text Mining: Classification, Clustering, and Applications.* HMinneapolis: CRC Press, 2009. ISBN 978-1-4200-5940-3.

ULDRICH, M. *Text mining aneb Kladivo na nestrukturovaná data.* IT Systems 12/2011: Text mining [online]. Časopis IT Systems, 2011, 12/2011 [cit. 2016-2-12].ISSN 1802-615X. Dostupné na: <http://www.systemonline.cz/clanky/text-mining-kladivona-nestrukturovana-data.htm>.

WEISS, S. M., INDURKHYA, N., ZHANG, T., DAMERAU, F. J. *Text Mining: Predictive Methods for Analyzing Unstructured Information.* Yorktown Heights: Springer, 2010, s. 237-239, ISBN 978-1-4419-2996-9.

WITTEN, I. H., FRANK, E., HALL, M. A. *Data Mining: Practical Machine Learning Tools and Techniques.* Burlington: Morgan Kaufmann, 2011, s. 629-630, Morgan Kaufman series in data management systems, ISBN 978-0-12-374856-0.

WU, W., XIONG, H., SHEKHAR, S. *Clustering and Information Retrieval.* Norwell, Mass.: Kluwer Academic Publishers, 2004, viii, 329 s. ISBN 9781402076824.

ZHANG, Y., JIN, R. *Understanding bag-of-words model: a statistical framework.* International Journal of Machine Learning and Cybernetics [online]. 2010, vol. 1, s. 43-52 [cit. 2016-02-12]. DOI: 10.1007/s13042-010-0001-0. Dostupné na: <http://link.springer.com/10.1007/ s13042-010-0001-0>.

ZHU, X., GOLDBERG, A. B. *Introduction to semi-supervised learning.* Synthesis lectures on artificial intelligence and machine learning 3.1, 2009, s. 1-130.

ŽIŽKA, J. *Business Intelligence.* [online]. Vysoká škola ekonomie a managementu, Praha, 2011. [cit. 2016-2-12]. Dostupné na: <www.vsem.cz/data/data/sis-texty/studijni-texty-bc/st_pis_bi_zizka.pdf>.

Žižka, J., Dařena, F. *Automatic Sentiment Analysis Using the Textual Pattern Content Similarity in Natural Language.* Lecture Notes in Computer Science. 2010. sv. 6231, č. 1, s. 224-231. ISSN 0302-9743.

Žižka, J., Dařena, F. *Mining Significant Words from Customer Opinions Written in Different Natural Languages.* Lecture Notes in Computer Science. 2011. sv. 6836, č. 1, s. 211-218. ISSN 0302-9743.

Žižka, J., Svoboda, A. *Customers' Opinion Mining from Extensive Amount of Textual Reviews in Relation to Induced Knowledge Growth.* Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis, 2015, 63.6: 2229-2237.

Přílohy

A Hodnocení úspěšnosti klasifikátoru

Tab. 27: Hodnocení úspěšnosti klasifikátoru pro GracePeriod (TF-IDF, 50 tis. dokumentů)

par_name	accuracy	tp_rate	fp_rate	precision	recall	f_measure
G0	61,92	0,62	0,62	0,72	0,62	0,47
G10	71,28	0,71	0,41	0,72	0,71	0,69
G20	73,11	0,73	0,21	0,79	0,73	0,73
G40	69,99	0,70	0,26	0,74	0,70	0,70
G80	66,71	0,67	0,23	0,78	0,67	0,66
G100	64,12	0,64	0,24	0,78	0,64	0,63
G200	81,03	0,81	0,21	0,81	0,81	0,81
G500	80,54	0,81	0,25	0,81	0,81	0,80
G750	74,54	0,75	0,38	0,77	0,75	0,72
G1000	80,27	0,80	0,26	0,80	0,80	0,80

Tab. 28: Hodnocení úspěšnosti klasifikátoru pro TieThreshold (TF-IDF, 50 tis. dokumentů)

par_name	accuracy	tp_rate	fp_rate	precision	recall	f_measure
H0	76,06	0,76	0,36	0,78	0,76	0,74
H0.05	81,03	0,81	0,21	0,81	0,81	0,81
H0.1	69,17	0,69	0,23	0,77	0,69	0,69
H0.2	73,55	0,74	0,20	0,80	0,74	0,74
H0.25	72,88	0,73	0,21	0,79	0,73	0,73
H0.3	72,88	0,73	0,21	0,79	0,73	0,73
H0.4	72,88	0,73	0,21	0,79	0,73	0,73
H0.5	72,88	0,73	0,21	0,79	0,73	0,73
H0.75	72,88	0,73	0,21	0,79	0,73	0,73
H1	72,88	0,73	0,21	0,79	0,73	0,73

Tab. 29: Hodnocení úspěšnosti klasifikátoru pro GracePeriod (TP, 50 tis. dokumentů)

par_name	accuracy	tp_rate	fp_rate	precision	recall	f_measure
G0	61,86	0,62	0,61	0,74	0,62	0,48
G10	73,89	0,74	0,31	0,74	0,74	0,74
G20	73,42	0,73	0,36	0,73	0,73	0,72
G40	71,54	0,72	0,20	0,81	0,72	0,72
G80	65,11	0,65	0,56	0,75	0,65	0,55
G100	68,54	0,69	0,50	0,76	0,69	0,62
G200	82,29	0,82	0,23	0,82	0,82	0,82
G500	81,87	0,82	0,24	0,82	0,82	0,81
G750	82,52	0,83	0,24	0,83	0,83	0,82
G1000	82,52	0,83	0,24	0,83	0,83	0,82

Tab. 30: Hodnocení úspěšnosti klasifikátoru pro TieThreshold (TP, 50 tis. dokumentů)

par_name	accuracy	tp_rate	fp_rate	precision	recall	f_measure
H0	67,53	0,68	0,52	0,75	0,68	0,60
H0.05	82,29	0,82	0,23	0,82	0,82	0,82
H0.1	72,57	0,73	0,20	0,79	0,73	0,73
H0.2	78,53	0,79	0,17	0,82	0,79	0,79
H0.25	73,37	0,73	0,21	0,78	0,73	0,74
H0.3	73,37	0,73	0,21	0,78	0,73	0,74
H0.4	73,37	0,73	0,21	0,78	0,73	0,74
H0.5	73,37	0,73	0,21	0,78	0,73	0,74
H0.75	73,37	0,73	0,21	0,78	0,73	0,74
H1	73,37	0,73	0,21	0,78	0,73	0,74

B Ukázka stromové struktury Hoeffdingův strom

```
Model description:
if [att 490:HELPFUL] <= 0.209:
  if [att 541:LOCATION] <= 0.14463636363636365:
    if [att 23:COULD] <= 0.265:
      if [att 431:THERE] <= 0.2127272727272727:
        if [att 685:CLOSE] <= 0.24336363636363637:
          if [att 692:POOR] <= 0.3649090909090909:
            if [att 726:YOUR] <= 0.335:
              if [att 782:PRICE] <= 0.26672727272727276:
                if [att 780:FRIENDLY] <= 0.1869090909090909:
                  if [att 798:TOO] <= 0.28063636363636363:
                    if [att 684:COMFORTABLE] <= 0.4475454545454545:
                      Leaf [class:_CLASS_] = <class 1:2> weights: {106|65,446}
                    if [att 684:COMFORTABLE] > 0.4475454545454545:
                      Leaf [class:_CLASS_] = <class 2:1> weights: {0|29,554}
                  if [att 798:TOO] > 0.28063636363636363:
                    Leaf [class:_CLASS_] = <class 1:2> weights: {30,211|0}
                if [att 780:FRIENDLY] > 0.1869090909090909:
                  Leaf [class:_CLASS_] = <class 2:1> weights: {1|58,87}
              if [att 782:PRICE] > 0.26672727272727276:
                Leaf [class:_CLASS_] = <class 2:1> weights: {11|47,04}
            if [att 726:YOUR] > 0.335:
              Leaf [class:_CLASS_] = <class 1:2> weights: {29,598|7}
          if [att 692:POOR] > 0.3649090909090909:
            Leaf [class:_CLASS_] = <class 1:2> weights: {34,437|3}
        if [att 685:CLOSE] > 0.24336363636363637:
          Leaf [class:_CLASS_] = <class 2:1> weights: {6|69,868}
      if [att 431:THERE] > 0.2127272727272727:
        if [att 685:CLOSE] <= 0.24336363636363637:
          Leaf [class:_CLASS_] = <class 1:2> weights: {69|14,319}
        if [att 685:CLOSE] > 0.24336363636363637:
          Leaf [class:_CLASS_] = <class 2:1> weights: {0|18,681}
    if [att 23:COULD] > 0.265:
      Leaf [class:_CLASS_] = <class 1:2> weights: {87,537|16}
  if [att 541:LOCATION] > 0.14463636363636365:
    if [att 666:FAR] <= 0.3487272727272727:
      if [att 896:BETTER] <= 0.3877272727272727:
        Leaf [class:_CLASS_] = <class 2:1> weights: {9,414|127}
      if [att 896:BETTER] > 0.3877272727272727:
        Leaf [class:_CLASS_] = <class 1:2> weights: {4,586|0}
    if [att 666:FAR] > 0.3487272727272727:
      Leaf [class:_CLASS_] = <class 1:2> weights: {11,059|2}
if [att 490:HELPFUL] > 0.209:
  if [att 653:NOT] <= 1.5632727272727274:
    if [att 19:WOULD] <= 2.4890909090909092:
      Leaf [class:_CLASS_] = <class 2:1> weights: {0|70}
    if [att 19:WOULD] > 2.4890909090909092:
      Leaf [class:_CLASS_] = <class 2:1> weights: {2|5}
```