VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Radek Stříteský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SÉMANTICKÉ ROZPOZNÁVÁNÍ KOMENTÁŘŮ NA WEBU
SEMANTIC RECOGNITION OF COMMENTS ON THE WEB

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Radek Stříteský

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Lukáš Povoda

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Radek Stříteský ***ID:*** 154876
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Sémantické rozpoznávání komentářů na webu

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte algoritmus, který pomůže s automatickým rozpoznávaním komentářů na webových stránkách. Algoritmus bude postaven na klasifikaci pomocí strojového učení. Součásti práce je tvorba databáze pro trénování a testování klasifikátorů, ověření přesnosti, vhodná prezentace výsledků a zhodnocení dosažených výsledků. Algoritmus připravte pro použití na serveru pro automatický sběr informací.

DOPORUČENÁ LITERATURA:

[1] DUCKETT, Jon. HTML and CSS: design and build websites. John Wiley & Sons, 2011.

[2] MARSLAND, Stephen. Machine learning: an algorithmic perspective. Boca Raton: CRC Press, c2009. Chapman & Hall/CRC machine learning & pattern recognition series. ISBN 978-1-4200-6718-7.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Lukáš Povoda
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Hlavním cílem diplomové práce je sémantické rozpoznávání komentářů na webových stránkách s použitím strojového učení. Teoretická část je zaměřena na umělou inteligenci, zejména se zde popisují metody strojového učení. Jsou zde podrobně popsány metody SVM, k-NN, rozhodovací strom a náhodný les. Dále se v teoretické části nachází popis HTML DOM dokumentu a CSS selektorů. V druhé části diplomové opráce je popsán postup pro automatické vyhledávání komentářů na webových stránkách. Nachází se zde generování trénovací databáze pomocí generátorů příznaků. Pro predikci třídy na trénovacích datech byl využit program RapidMiner. Výsledkem je program pro automatické vyhledávání komentářů z webové stránky vytvořený v programovacím jazyku JAVA. Součástí práce je tvorba trénovací databáze pro trénování a testování klasifikátorů, ověření přesnosti a vyhodnocení úspěšnosti jednotlivých použitých klasifikátorů.

KLÍČOVÁ SLOVA

RapidMiner, Java, strojové učení, SVM, rozhodovací strom, náhodný les, k-NN

ABSTRACT

The main goal of the diploma thesis is Semantic recognition of comments on websites using machine learning. The theoretical part is focused on the artificial intelligence, especially the methods of machine learning are describing here. In this part we can also find detailed explanation of methods like SVM, k-NN, decision tree or random forest. Further HTML DOM document and CSS selectors are discussed here. In the second part of this work is the practical procedure for automatic searching for comments on websites. The generating of training databases using feature generator is also described in this part of diploma thesis. Software RapidMiner was used for the class prediction based on training data. The result is program for automatic recognition of comments on websites in programming language JAVA. The important part of thesis is creation of training database for training and also testing of classificators, verification of accuracy and evaluation of success of each used classifier.

KEYWORDS

RapidMiner, Java, machine learning, SVM, decision tree, random forest, k-NN

STŘÍTESKÝ, Radek *Sémantické rozpoznávání komentářů na webu*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 50 s. Vedoucí práce byl Ing. Lukáš Povoda

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Sémantické rozpoznávání komentářů na webu“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce panu Ing. Lukášovi Povodovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Sémantické vyhledávání komentářů a jiných struktur z webových stránek na základě kontextu je v dnešní době, kdy se co nejvíce informací přesouvá na webové servery propojené sítí Internet, velmi perspektivní obor. Historie webových stránek začíná v roce 1990, kdy Berners-Lee vyvinul první verzi protokolu pro výměnu hypertextových dokumentů, značkovací jazyk a první prohlížeč webových stránek. První myšlenka webových stránek byla vyměňovat si vědecké informace napříč celým světem. Dnes jsou webové stránky dostupné všem občanům z civilizovaných zemí na světě a jejich funkčnost a velikost je úplně v jiné dimenzi. Je nevyhnutelné velké množství dat třídit do tříd a vyhledávat v nich informace. Tato diplomová práce by měla přispět k automatickému vyhledávání komentářů a jiných částí webových stránek.

Hlavním cílem diplomové práce je sémantické rozpoznávání komentářů na webových stránkách s použitím strojového učení s učitelem. Součástí práce je tvorba trénovací databáze pro trénování modelu a testování klasifikátorů, ověření přesnosti a vyhodnocení úspěšnosti jednotlivých použitých klasifikátorů. Výsledkem praktické části je program pro automatický sběr informací z webových serverů.

Teoretická část je zaměřena na umělou inteligenci, zejména se zde popisují metody strojového učení s učitelem. Jsou zde podrobně popsány metody SVM, k-NN, rozhodovací strom a náhodný les. Dále se v teoretické části nachází popis struktury dokumentu webové stránky a CSS selektorů. V druhé části diplomové opráce je popsán postup pro automatické vyhledávání komentářů na webových stránkách. Nachází se zde generování trénovací databáze pomocí generátorů příznaků. Pro predikci třídy na trénovacích datech byl využit program RapidMiner. Výsledkem je program pro automatické vyhledávání komentářů z webové stránky vytvořený v programovacím jazyku JAVA. Součástí práce je tvorba trénovací databáze pro trénování a testování klasifikátorů, ověření přesnosti a vyhodnocení úspěšnosti jednotlivých použitých klasifikátorů.

1 WEBOVÉ TECHNOLOGIE

WWW (World Wide Web) je označení pro systém prohlížení, ukládání a odkazování dokumentů nacházející se v celosvětové síti Internet. Autorem webu je Tim Berners-Lee, který navrh jazyk HTML (HyperText Markup Language), protokol HTTP (Hypertext Transfer Protocol) a napsal první webový prohlížeč. Na konci roku 1990 spustil první webový server na světě. Od vzniku se web závratným tempem vyvíjí a dnes je nepostradatelnou součástí běžného života. Velké množství informací nacházejících se na webových serverech je nutné filtrovat a vyhledávat v nich důležité informace. [1]

1.1 Protokol HTTP

Protokol HTTP je aplikační bezstavový protokol, který je určen pro výměnu hypertextových dokumentů. Nejznámějším a nejpoužívanějším hypertextovým dokumentem je WWW, který je označován pro systém prohlížení, ukládání a odkazování dokumentů nacházejících se v síti internetu. HTTP spolu s WWW se používá od roku 1990.

1.1.1 Verze HTTP protokolu

HTTP 0.9 1990

První verze protokolu HTTP obsahuje pouze jednu metodu GET s jedním parametrem, a to názvem požadovaného dokumentu. Server posílá jako odpověď přímo požadovaný dokument bez hlavičky. Chybová hlášení se vrací rovněž ve formě dokumentu. [2]

HTTP 1.0 1996

Tato verze přináší do HTTP hlavičky a nové metody HEAD a POST. Pro identifikaci dokumentů používá MIME (Multipurpose Internet Mail Extensions). [2] [1]

HTTP 1.1 1997

Tato verze umožnuje udržovat TCP spojení a dále přidává další metody OPTIONS, PUT, DELETE, TRACE a CONNECT. [3]

[1] MIME (Multipurpose Internet Mail Extensions) - víceúčelová rozšíření internetové pošty, využívá se například pro identifikaci formátu souboru

HTTP 2.0 2015

Nová verze přináší několik dalších vylepšení HTTP protokolu. Jedná se o binární protokol. Do textové verze se překládá v debugerech. Dále podporuje Multiplexing - umožňuje v jednom spojení poslat více požadavků, kde nezáleží na pořadí odpovědí. [4]

1.1.2 Dotazovací metody

HTTP definuje několik metod, které se mají provést nad daným objektem. Metody HEAD, GET, OPTIONS a TRACE jsou definované jako bezpečné, slouží pouze pro získávání informací. Metody POST, PUT a DELETE jsou definované jako nebezpečné. Tyto metody obvykle mění objekt na serveru. [3]

Metody PUT a DELETE jsou definované jako idempotentní, což znamená, že více totožných požadavků by mělo mít stejný účinek jako jeden požadavek.

- GET
  Metoda GET znamená získat všechna data pomocí identifikátoru URI (Uniform Resource Identifier). Jestliže požadavek URI odkazuje na zpracování dat, jsou vrácena výsledná data z procesu.
- HEAD
  Metoda HEAD je identická s metodou GET, ale v odpovědi nevrací tělo zprávy. Využívá se pro získávaní metadat a testování hypertextových odkazů.
- POST
  Metoda POST slouží pro odesílání dat na server pomocí identifikátoru URI. Používá se například pro odesílání formulářů.
- PUT
  Metoda slouží pro editaci zdroje. V případě, že identifikátor URI neodkazuje na existující zdroj, může být vytvořený nový.
- DELETE
  Metoda DELETE slouží pro odstranění zdroje ze serveru pomocí identifikátoru URI.
- TRACE
  Metoda TRACE se používá ke zjištění cíle požadavku v aplikační vrstvě. Cílový příjemce požadavku by měl zprávu odeslat jako tělo entity s odpovědí 200 (OK).
- OPTIONS
  Metoda OPTIONS reprezentuje požadavek na informace o možnostech komunikace kanálu identifikovaného URI. Tato metoda umožňuje klientovi identifikovat možnosti a požadavky na zdroj.

- CONNECT
  Spojí se s uvedeným objektem přes uvedený port. Používá se při průchodu skrze proxy pro ustanovení kanálu SSL (Secure Sockets Layer).

1.1.3 Stavové kódy HTTP

Stavový kód informuje, jak byla odpověď serverem zpracována. [3]

- 1xx - Informační
  Zprávy definované konkrétní aplikací.
- 2xx - Úspěch
  Tato třída kódu označuje, že požadavek byl úspěšně přijat.
- 3xx - Přesměrování
  Klient musí pro konečné zpracování požadavků vykonat určitou další činnost.
- 4xx - Chyba klienta
  Tato třída kódu informuje, že nastala chyba na straně klienta.
- 5xx - Chyba serveru
  Tato třída kódu informuje, že nastala chyba na straně serveru.

1.2 Objektový model dokumentu

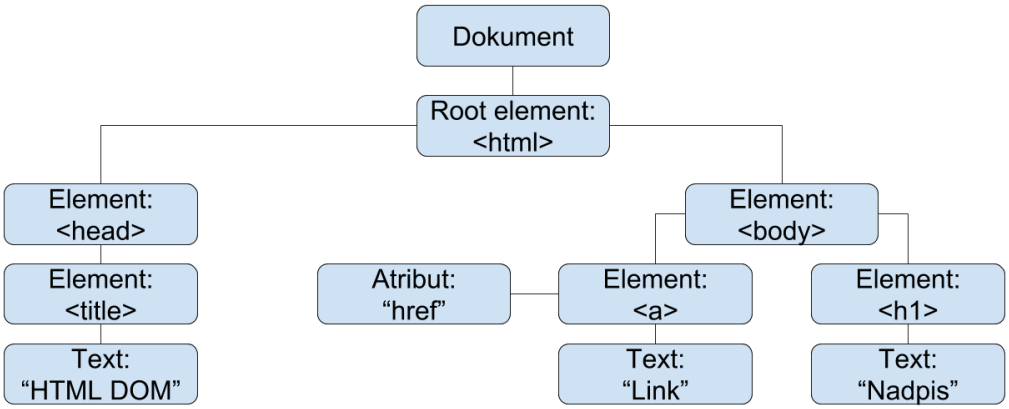
DOM (Document Object Model) je objektově orientovaná reprezentace XML (eXtensible Markup Language) nebo HTML dokumentu. DOM je API (Application Programming Interface) umožňující přístup a modifikaci obsahu, struktury nebo stylu dokumentu. DOM struktura dokumentu je implementována ve všech moderních webových prohlížečích a lze ji obsluhovat pomocí JavaScriptu.

DOM umožňuje přístup k dokumentu jako ke stromu, což je také datová struktura využívaná v parserech dokumentů. To umožňuje metoda nazývaná grove (Graph Representation Of property ValuEs), která vyžaduje nahrání celého parsovaného dokumentu do paměti.

Objektový model dokumentu vznikl pro potřeby přístupu k některým typům elementů na webové stránce pomocí JavaScriptu. [5]

1.3 CSS selektor

Kaskádové styly (Cascading Style Sheets) jsou jazykem pro popis způsobu zobrazení elementů ve webovém prohlížeči. Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury. Pomocí CSS selektorů lze vybrat z webové stránky HTML elementy. Základní rozdělení CSS selektorů je následující:



Obr. 1.1: Ukázka struktury HTML dokumentu

- Základní - Do této skupiny spadají selektory, které vybírají elementy podle třídy, ID a atributu.
- Kombinované - Kombinované selektory zahrnují element dítěte, sourozence nebo potomků.
- Pseudoelementy - Jsou speciální elementy, které prohlížeč vloží do existujícího elementu na základě nějakého stavu. Příklad pseudoelementu je první odstavec nebo první písmeno.
- Pseudotřídy - Pseudotřídy jsou zvláštním typem tříd, které jsou aktivovány událostí nebo stavem. Jsou vždy aplikovány na celý element. Příklad pseudotřídy je navštívený odkaz, najetí myši na element nebo kliknutí na element. [6]

2 UMĚLÁ INTELIGENCE

Umělá inteligence je vědní obor zabývající se stroji, které vykazují inteligenci. Přesná definice umělé inteligence neexistuje, existují pouze obecné definice. Jednou z nejlépe přijatých je definice podle Marvina Minského:„Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence“. [7]

2.1 Strojové učení

Strojové učení (Machine learning) je podoblastí umělé inteligence zabývající se algoritmy a technikami, které umožňují strojům proces učení. Strojové učení se prolíná se statistikou a dolováním znalostí a má široké uplatnění. Jeho využití pokrývá celou řadu možností, od rozpoznání ručně psaných textů přes rozpoznání řeči až po nelegální užití kreditní karty. V současné době se potýkají se zvýšeným zájmem v souvislosti s dolováním dat z nepřeberného množství databází.

Algoritmy strojového učení určují příznaky zkoumaných dat a vyhledávájí souvislosti mezi nimi. Výsledkem je natrénovaný model, který umožuje predikci zařazení do třídy.

Základní rozdělení algoritmů podle způsobů učení je následující:

- Učení s učitelem - trénovací databáze obsahuje správné zařazení vzorků do tříd nebo spojitou hodnotu při regresi.
- Učení bez učitele - ke vstupním datům není známý výstup.
- Kombinovaná - část vstupních dat je se známým výstupem (zařazení do třídy) a další data, typicky větší množina, jsou bez známého výstupu.

Rozdělení podle reprezentace příkladů použitých v procesu učení:

- Atributy - vlastnosti objektů reprezentované řádky v datové tabulce. Základní rozdělení atributů je *kategoriální (diskrétní)* a *numerické (spojité)*. Toto základní rozdělení platí pro většinu algoritmů strojového učení. Kategoriální lze dále rozdělit na *binární* (nabývají pouze dvou hodnot), *nominální* (nabývající jedné z konečného počtu hodnot, které nejsou navzájem uspořádány) a *ordinální* (nabývající jedné z konečného počtu navzájem uspořádaných hodnot).
- Relace - řada navzájem provázaných vztahů mezi objekty a atributy. [8] [9]

2.1.1 Algoritmy podpůrných vektorů (SVM)

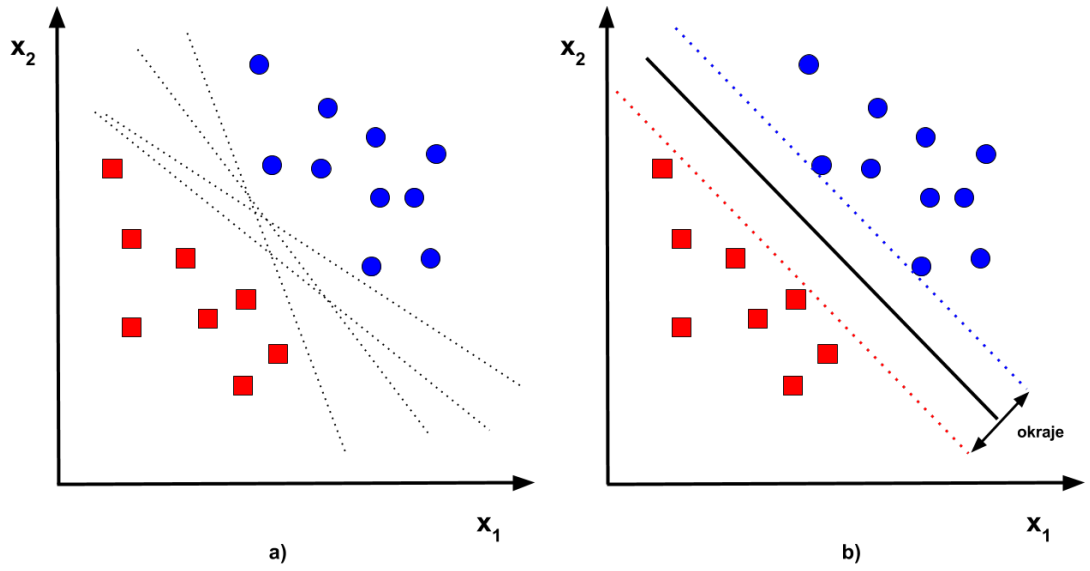
SVM (Support vector machine) je metoda strojového učení určená pro klasifikaci úlohy, regresivní analýzu nebo neparametrický odhad hustoty. Tato metoda strojo-

vého učení patří do kategorie jádrových algoritmů (kernel machines). Metodu rozpracoval v roce 1963 Vladimir Naumovič Vapnik a Alexej Jakovlevič Červoněnkis.

SVM se nejčastěji používá k rozdělení dat do dvou skupin. Metoda hledá pro trénovací množinu (vektory) nadrovinu (hyperplane), která data rozděluje do dvou poloprostorů. Výsledná nadrovina by měla splňovat jednu základní podmínku, hodnota minima vzdáleností bodů od nadroviny by měla být co největší. Pokud není trénovací množina příznaků lineárně separovatelná, převede se původní prostor do vícerozměrného, kde lze nalézt lineární nadrovinu rozdělující data do dvou tříd.

Lineární SVM

Jedná se o základní variantu SVM klasifikátoru, kde zůstáváme v původním prostoru příznaků a nedochází k žádné jádrové transformaci. Trénovací data lze vždy rozdělit několika separačními liniemi, což je zobrazeno na levé části obr. 2.1. Úkolem je najít optimální nadrovinu z hlediska kategorizace budoucích dat.



Obr. 2.1: a) Možné oddělovací nadroviny. b) Optimální oddělovací nadrovina

Příklad vhodně zvolené nadroviny je zobrazen v pravé části obr. 2.1, principem je nalezení nejširšího rozdělovacího pásma. Body ležící na hranici tohoto pásma se nazývají podpůrné vektory (support vectors). Odebráním jednoho z nich by se změnila poloha optimální nalezené nadroviny. U výsledné nadroviny musí platit následující

vztahy:

$$\begin{aligned} \vec{w} \cdot \vec{x} + b &= 0 \quad (na\ přímce\ A1) \\ \vec{w} \cdot \vec{x} + b &= -1 \quad (na\ podpůrném\ vektoru\ a_{12}) \\ \vec{w} \cdot \vec{x} + b &= +1 \quad (na\ podpůrném\ vektoru\ a_{22}). \end{aligned} \tag{2.1}$$

Pro natrénovaný klasifikátor musí platit následující rovnice:

$$f(\vec{x}) = \begin{cases} 1, & \vec{w}, \cdot\vec{x} + b \geq 1 \\ -1, & \vec{w} \cdot \vec{x} + b \leq -1 \end{cases} \tag{2.2}$$

Šířku okrajů (margin) lze vyjádřit vztahem:

$$Margin = \frac{2}{\|\vec{w}\|^2}. \tag{2.3}$$

Často se při výpočtech zavádí tzv. laxní proměnné, které porušují podmínku správného zařazení do tříd. Velkou výhodou těchto proměnných je získání širšího pásma nebo možnost separovat nelineární prostor. Může se jednat o zašuměná data, kde se jednotlivé třídy překrývají a není možné najít lineární nadrovinu prostoru. Obr. 2.2 ilustruje získání větší šířky okrajů optimální nadroviny. Toto řešení by mělo zvyšovat úspěšnost budoucí kategorizace. Upravená rovnice s přidanou proměnou ξ vypadá následovně:

$$f(\vec{x}) = \begin{cases} 1, & \vec{w}, \cdot\vec{x} + b \geq 1 - \xi \\ -1, & \vec{w} \cdot \vec{x} + b \leq -1 - \xi \end{cases} \tag{2.4}$$

Jádrová transformace SVM

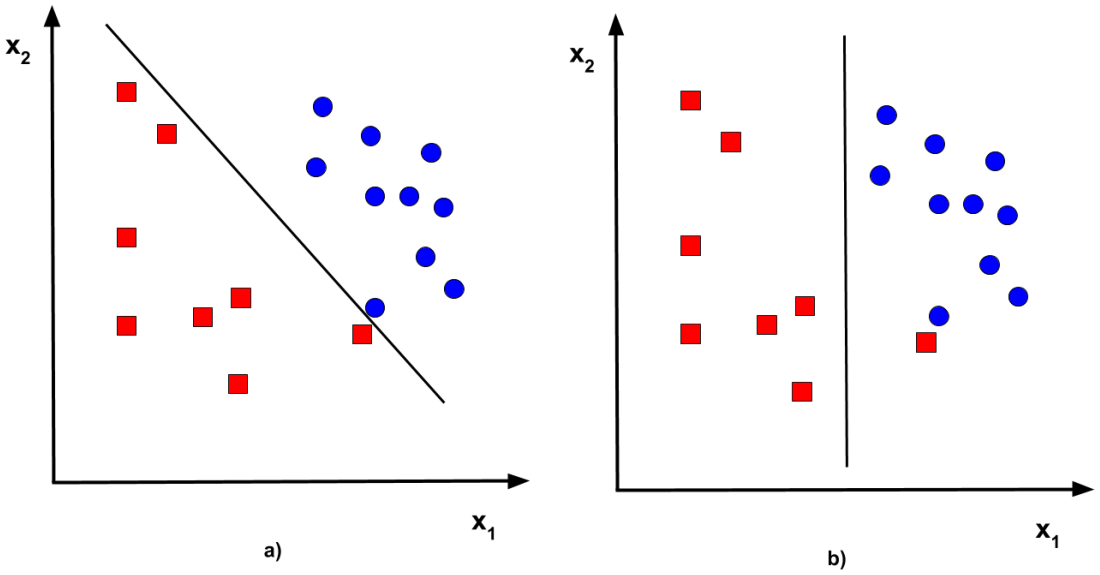
Největším přínosem klasifikátoru SVM je transformace l-rozměrného prostoru x na prostor definovaný systémem nelineárních funkcí $\varphi(x)$, jehož dimenze nabývá vyšších hodnot a může být až nekonečná. Transformace vstupních lineárně neseparovatelných dat do lineárně separovatelného prostoru s vyšší dimenzí je zobrazena na obr. 2.3. [10] [11] [12] [9]

Varianty jádrové transformace jsou:

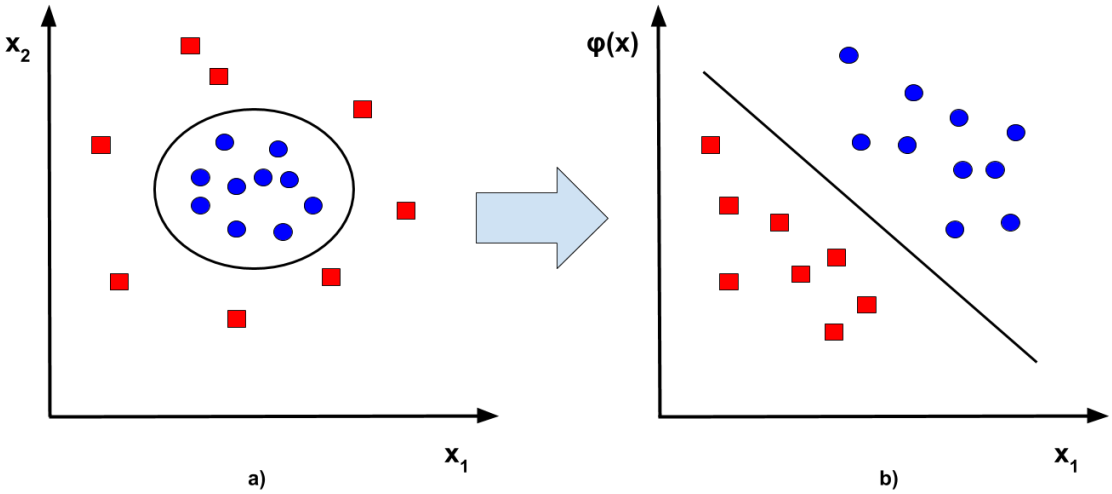
- Lineární
- Polznomická řada
- RBF (Radial Base Function)
- Sigmoida

2.1.2 Rozhodovací strom (Decision Tree)

Jedná se o hierarchický, nelineární systém umožňující uložení znalostí. Typickým příkladem uložení informace je *botanický klíč*, ve kterém lze postupným procházením



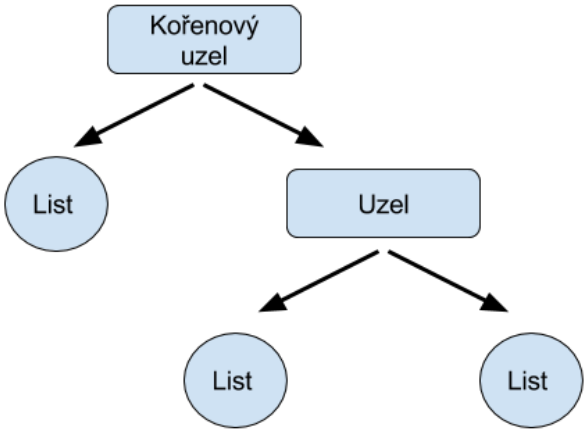
Obr. 2.2: a) Lineárně separovatelná data, ale za cenu užší šířky okrajů. b) Větší šířka okrajů za cenu porušení podmínky správné klasifikace.



Obr. 2.3: a) Vstupní lineárně neseparovatelná data. b) L-rozměrný prostor, kde lze data lineárně rozdělit do tříd.

atributů rostliny určit její název. Hlavním znakem rozhodovacího stromu je hierarchická struktura. Po rozdělení uzlu na základě hodnoty atributu vnikají nové uzly, každý nový uzel má parametry nezávislého podstromu. Další významnou vlastností rozhodovacího stromu je jeho flexibilita. V uzlu může být rozhodováno na základě vícerozměrné funkce tvořené kombinací vybraných nezávislých atributů. Další vý-

hodou je čitelnost rozhodovacích stromů. Z hlediska bezpečnosti je dobré rozumět systému, který klasifikuje data do tříd. Složitost je nevýhodou neuronových sítí, kde jejich složitá struktura neumožňuje dostatečně porozumět výslednému modelu. Znalosti z vytvořeného modelu lze extrahovat nebo použít k predikci nových dat.



Obr. 2.4: Struktura rozhodovacího stromu.

Popis modelu

Klasifikační strom je model pro data, kde pozorování patří do některé z tříd $C_1, ..., C_k$, přičemž $k = 2$. Pozorování je současně charakterizované vektorem $x = (x_1, ..., x_m)$, kde hodnoty prediktorů jsou $X_1, ..., X_m$.

Model klasifikátoru se popisuje stromovým grafem složeným z uzlů a orientovaných hran (orientace hran shora dolů).

V každém nedeterministickým uzlu se strom větví do dvou nebo více dceřiných uzlů. Nejběžnější je binární větvení, kde se porovná hodnota prediktoru. Tomu odpovídá kladná a záporná orientovaná hrana vedoucí k dceřinému uzlu.

Pozorování dat podle hodnot prediktorů prochází od kořenového uzlu přes větvení v neterminálních uzlech do konečného terminálního uzlu (listu). Terminálnímu uzlu je přiřazena některá z tříd $\{C_1, ..., C_k\}$. Strom určuje klasifikační funkci d_T definovanou na X s hodnotami v množině $\{C_1, ..., C_k\}$.

Algoritmus pro tvorbu stromu

Při vytváření rozhodovacích stromů se postupuje metodou „rozděl a panuj“ (divide and oncquer). Trénovací data se rozdělují na podmnožiny (uzly stromu) tak, aby v podmnožinách převládala jedna ze tříd. Tedy entropie uzlu se blíží k nule. Tento

postup bývá označován jako TDIDT (Top Down Induction of Decision Trees). Cílem je nalézt strom konzistentní s trénovacími daty, přitom se dává přednost jednodušším strukturám stromu.

Obecný postup algoritmu TDIDT:

1. Zvol jeden atribut jako kořen dílčího stromu.
2. Rozděl data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidej uzel pro každou podmnožinu.
3. Existuje-li uzel, pro který nepatří všechna data do téže třídy, pro tento uzel opakuj postup od bodu 1, jinak skonči.

Zvolení vhodného atributu pro větvení stromu je hlavní předpoklad pro daný algoritmus. Cílem je vybrat atribut, který od sebe nejlépe odliší příklady různých tříd. Výběr atributu je převzán z teorie informace a pravděpodobnosti. Mezi nejčastěji používaná rozdělovací kritéria patří: entropie, informační zisk, poměrný informační zisk a gini index.

Entropie je základním pojmem teorie informace, která se zabývá měřením, přenosem, kódováním, ukládáním a následným zpracováním informací z kvantitativního hlediska. Entropie znamená míru neurčitosti, s narůstající informací klesá a naopak. Entropii (střední hodnota informace na jeden symbol zprávy) definioval v roce 1948 Claude Elwood Shannon. Entropiie uzlu je definována následující rovnicí

$$H(S) = -\sum_{i=1}^{n} P_i \, log_2 \, P(_i), \tag{2.5}$$

kde n je počet navzájem se vylučujících stavů x a pravděpodobnost stavu je P_i. Znaménko zde značí, že entropie je záporná veličina. Entropie je maximální při rovnoměrém rozložení pravděpodobnosti všech jevů. Naopak při pravděpodobnosti p = 1 (všechnny prvky náleží do této třídy) nebo p = 0 (žádný prvek nenáleží do této třídy) je entropie maximální. Pro větvení rozhodovacího stromu je vybrán atribut s nejmenší entropií.

Informační zisk (Information Gain) a *poměrný informační zisk (information gain ratio)* jsou kritéria odvozená z entropie. Informační zisk se počítá jako rozdíl entropie pro celá data a pro uvažovaný atribut. Tím dosáhneme redukce entropie, která je zapříčiněná volbou uvažovacího atributu. Zatímco v případě entropie hledáme atribut s minimální hodnotou, v případě informačního zisku hledáme atribut s maximální hodnotou. První člen rozdílu je konstantní (entropie pro celá data), takže rozdíl bude maximální v případě, že druhý člen rozdílu bude minimální. Toto řešení nebere v potaz počet hodnot zvoleného atributu, uvažuje pouze jak tento atribut odlišuje příklady různých tříd. Volba nesprávného atributu umožní bezchybně klasifikovat trénovací data, ale je nepoužitelný pro predikci nových dat. Z tohoto důvodu se častěji používá *poměrný informační zisk (information gain ratio)*, který bere v úvahu

i počet hodnot atributů.

Gini index vyjadřuje číselný odklon Lorenzovy křivky od křivky dokonalého rozdělení. Atribut, který nejvíce redukuje špatná data, je vybrán jako rozdělovací atribut. Hodnota Gini indexu je minimální, pokud je v konečném uzlu zastoupena pouze jedna kategorie a maximální, pokud jsou v konečném uzlu proměnné zastoupeny rovnoměrně.

Algoritmů pro tvorbu rozhodovacích stromů je velká řada, nejpoužívanější jsou:

- Huntův algoritmus
- CART
- ID3
- C4.5
- SLIQ
- SPRINT

[13] [14] [17] [18] [15] [16]

2.1.3 Náhodný les (Random Forest)

Random forest jey metoda pro klasifikaci a regresi pomocí rozhodovacích stromů. Tuto metodu představil ve své práci Breiman v roce 2001.

Klasifikátor obsahuje soubor binárních rozhodovacích lesů. Myšlenkou náhodného lesa je natrénovat každý rozhodovací strom nezávisle na ostatních tak, že se budou navzájem náhodně lišit. Náhodné lesy do procesu zavádějí náhodu.

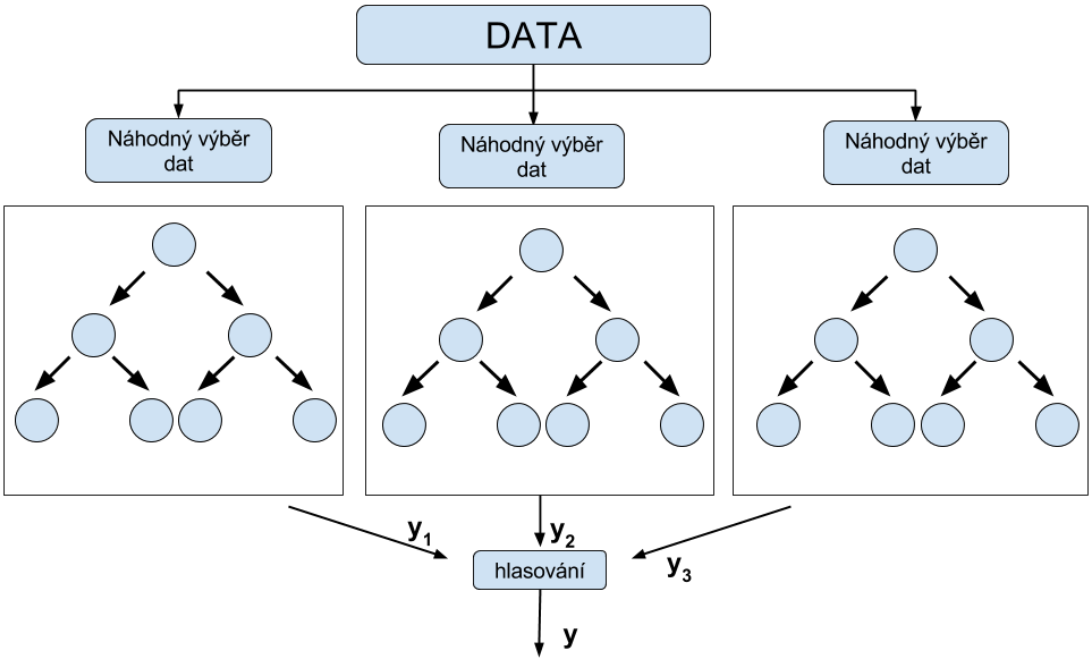
Problém nastává při vytváření stromů. Cílem je vytvořit stromy unikátní, protože kombinováním stejných stromů přidanou hodnotu nezískáme. Jedním ze způsobů vytváření stromů je metoda náhodný les.

Princip metody náhodný les:

- Z trénovacího datového souboru $L = \{(x_1, y_1), ..., (x_n, y_n)\}$ se náhodným výběrem vytvoří L souborů $L_1, ..., L_L$ velikosti n.
- Při volbě větvení pro daný uzel se z m prediktorů $X_1, ..., X_m$, které jsou k dispozici, nejdříve náhodně vybere některých m_0, načež se nejlepší větvení hledá již klasicky, ale jen mezi těmi větveními, která jsou založena na vybraných m_0 veličinách.
- Výsledkem jsou vytvořené velké stromy (neprořezávají se). [19]

2.1.4 K-nejbližších sousedů

Metoda k-NN (k - nearest - neighbor) je jedna z nejjednoduších metod strojového učení. Metoda je velmi přímočará, proto její popis bude podstatně kratší v porovnání s předchozími. Klasifikace je prováděna na základě podobnosti. Známe trénovací



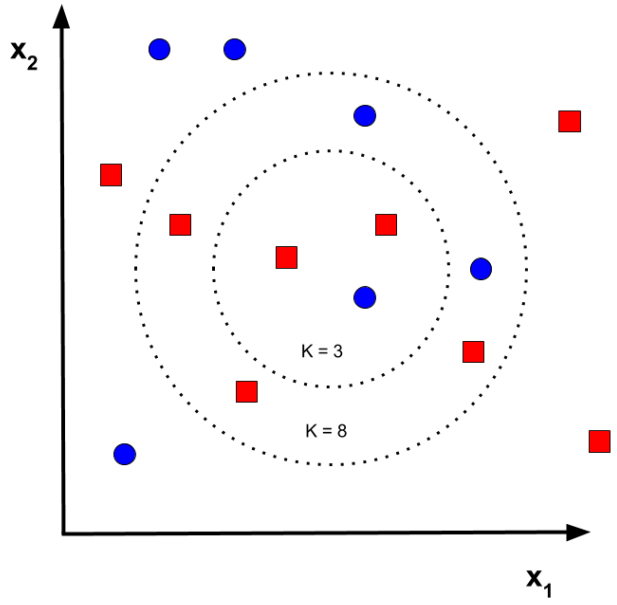
Obr. 2.5: Klasifikátor náhodný les

množinu $(x_i, \omega_i)_{i=1,..,K}$, kde x_i je vzorek, kterému je přiřazena třída ω_i a k je velikost trénovací množiny. Pro neznámý prvek x hledáme $x_k^{'}$ takové, že $||x_k^{'} - x|| = min||x_i^{'} - x||_{i=1,...,k}$. Prvek zařadíme do stejné třídy, do jaké náleží x_k.

Na obr. 2.6 vidíme příklad použití klasifikace nejbližších sousedů typu 3-NN a 6-NN. Kolem neznámého prvku je vytvořena hyperkoule, která obsahuje právě tři nebo šest nejbližších prvků z trénovací množiny. Neznámý prvek klasifikujeme do třídy, která je v hyperkouli zastoupena největším počtem. Při volbě parametru $k = 1$, se jedná o klasifikaci nejbližšího souseda. Neznámý prvek klasifikujeme do třídy, do které spadá nejbližší prvek.

Při použití metod k-NN je velmi důležitá volba parametru k. Například pro dvě třídy volíme lichý parametr k, který vede k jednoznačnému rozhodnutí pro klasifikaci neznámého prvku. Pro více tříd mohou nastat situace, kdy nelze jednoznačně rozhodnout.

Metoda k-nejbližších sousedů disponuje snadnou implementací a relativně vysokou účinností. Často se tato metoda v praxi používá jako referenční. Je vhodná především pro predikci do nižšího počtu tříd. Pro vyšší požadavky klasifikace existují modifikace této metody. [20] [21]



Obr. 2.6: Metoda k-nejbližších sousedů (k-NN)

2.2 Rapid Miner

Jedná se o nejrozšířenější a nejvíce používaný nástroj pro hloubkovou analýzu dat. Projekt vznikl na univerzitě v Dortmundu v roce 2001 pod původním názvem Yale. Celý program je napsaný v programovacím jazyce JAVA pod licencí GNU AGPL (Affero General Public License). Jedná se tedy o svobodný software s rozšířením použití softwaru přes síť.

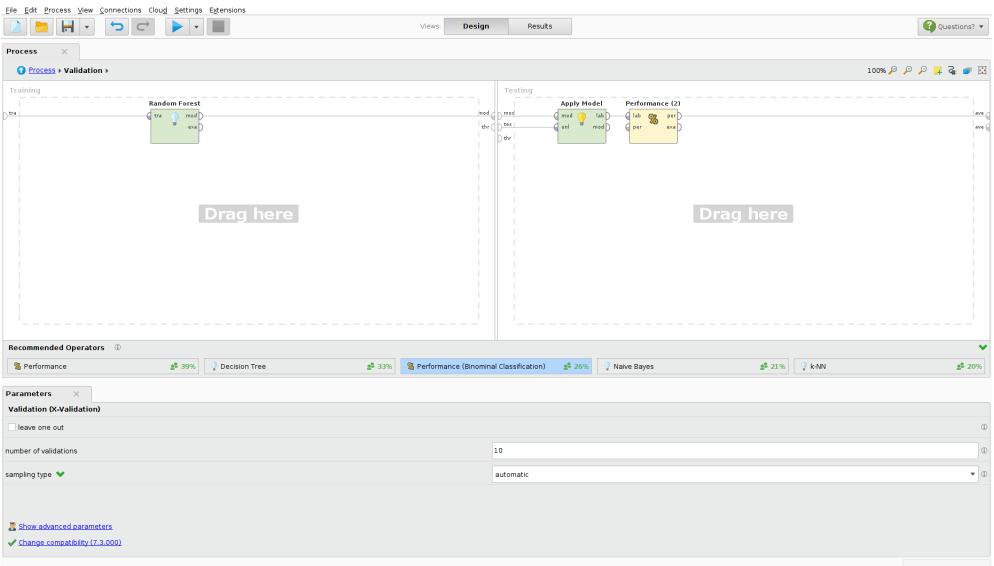
Program je využívaný velkým množstvím lidí z komerční sféry, univerzitami i výzkumnými středisky. Velkou výhodou tohoto programu je jednoduchá implementace nových operátorů a možnost jejich propojení s už existujícími.

RapidMiner disponuje velmi příjemným grafickým uživatelským rozhraním, které může být pro začínající uživatele příliš složité. Na druhou stranu je vhodné pro zkušené uživatele, kteří dokáží během pár minut vytvořit složitý proces z více jak 400 možných operátorů.

Proces se modeluje pomocí operátorů, které se navzájem spojují v jeden funkční celek. Operátor je objekt, který má vstupní a výstupní porty. Tyto porty umožňují propojení operátorů v jeden celek. Nejtěžší částí je správné nastavení parametrů u všech operátorů. Každý operátor má své vlastní specifické nastavení, které se vztahuje k jeho funkčnosti. Dále RapidMiner umožňuje hierarchické uspořádání operátorů. Zde například můžeme uvést operátor *X-Validation*, který ve své vnitřní struktuře obsahuje další operátory. Vnitřně je proces definovaný pomocí XML sou-

boru. Například při nevhodném definování parametrů u bloku *Read CSV* (blok určený ke čtení vstupních dat ve formátu CSV), proces spadne hned na následujícím bloku z důvodu nevalidních vstupních dat. Při návrhu procesu je nutné přihlížet k výkonostním parametrům počítače, zejména na velikost operační paměti. Při nedostaku operační paměti se celý proces ukončí.

Další významnou částí programu RapidMiner je načítání a ukládání dat pro další zpracování. Základním konceptem pro práci s daty je takzvaný repozitář. Jedná se o abstrakci souborového systému v operačním systému, která ulehčuje přenositelnost procesů. Všechny cesty, se kterými RapidMiner pracuje, jsou relativní vůči tomuto repozitáři. Pro načítání a ukládání dat do repozitáře má RapidMiner dva základní operátory. Prvním je operátor *Store*, který umožňuje ukládat výstup libovolného operátoru do repozitáře. Naopak operátor *Retrieve* umožňuje data z repozitáře načítat a přivádět na vstupní porty operátorů. Pomocí těchto operátorů lze ukládat natrénovaný model klasifikátoru a načítat v jiném procesu. Dále program umožňuje načítat data ze standardních zdrojů, například z CSV souborů a mnoha dalších. [22]



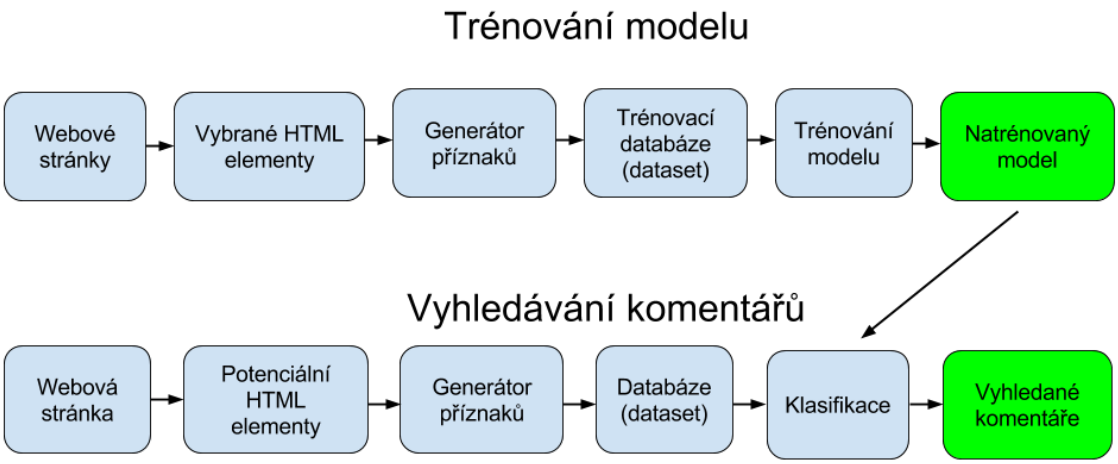
Obr. 2.7: Rapid Miner studio

3 PRAKTICKÁ ČÁST

Praktická část se zábývá sémantickým rozpoznáváním komentářů na webových stránkách. Sémantické vyhledávání funguje pomocí algoritmu strojového učení s učitelem. Praktická část se dělí na natrénování modelu a následnou klasifikaci potenciálních HTML elementů s využitím natrénovaného modelu. Na obr. 3.1 je znázorněno celé blokové schéma postupu vyhledávání komentářů na webových stránkách.

První částí je vytvoření algoritmu, který prochází předdefinované webové stránky a podle zadaných CSS selektorů vyhledává HTML elementy s komentáři. Pomocí generátorů příznaků a DOM struktury webové stránky se vytváří trénovací databáze. Spolu s pozitivními vzorky (komentáři) se vytvářejí také negativní vzorky. Jedná se o vzorky získané z HTML elementů, které se nacházejí na HTML stránce, ale nejedná se o komentáře. Následuje výběr vhodného algoritmu strojového učení a natrénování modelu v programu RapidMiner. První část praktické části diplomové práce slouží pouze pro natrénování modelu. Součástí této práce je i vytvoření trénovací databáze, tudiž natrénovaný model je již vytvořený a může se začít rovnou následujícím krokem.

Druhou částí je vyhledávání komentářů na webové stránce s využitím natrénovaného modelu. Nejprve dojde k vyhledání potencíalních HTML elementů s komentáři. Na tuto množinu HTML elementů se aplikuje generátor příznaků. Úplně nakonec je pro vyhledávání komentářů nutné klasifikovat vzorky získané z potenciálních HTML elementů do dvou tříd pomocí natrénovaného modelu. Výstupem celého programu je textový soubor s nalezenými komentáři ze zadané webové stránky.

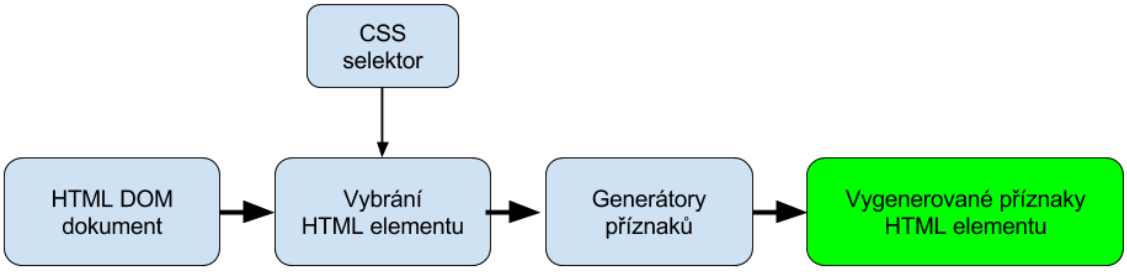


Obr. 3.1: Blokové schéma vyhledávání komentářů

3.1 Generátor příznaků

Generátor příznaků generuje atributy jednoho HTML elementu z dokumentu. Výsledné příznaky popisují zanoření elementu v dokumentu spolu s atributy elementů.

Na obr. 3.2 je znázorněn postup při generování příznaků. Na vstupu je vždy HTML stránka a CSS selektor, který vybírá jeden nebo více HTML elementů z dané stránky. Pro každý vybraný element se vygenerují příznaky pomocí předdefinovaných generátorů příznaků popsaných v další kapitole. Generátor příznaků se používá při vytváření klasifikačního modelu i při následném sémantickém vyhledávání komentářů na webové stránce.



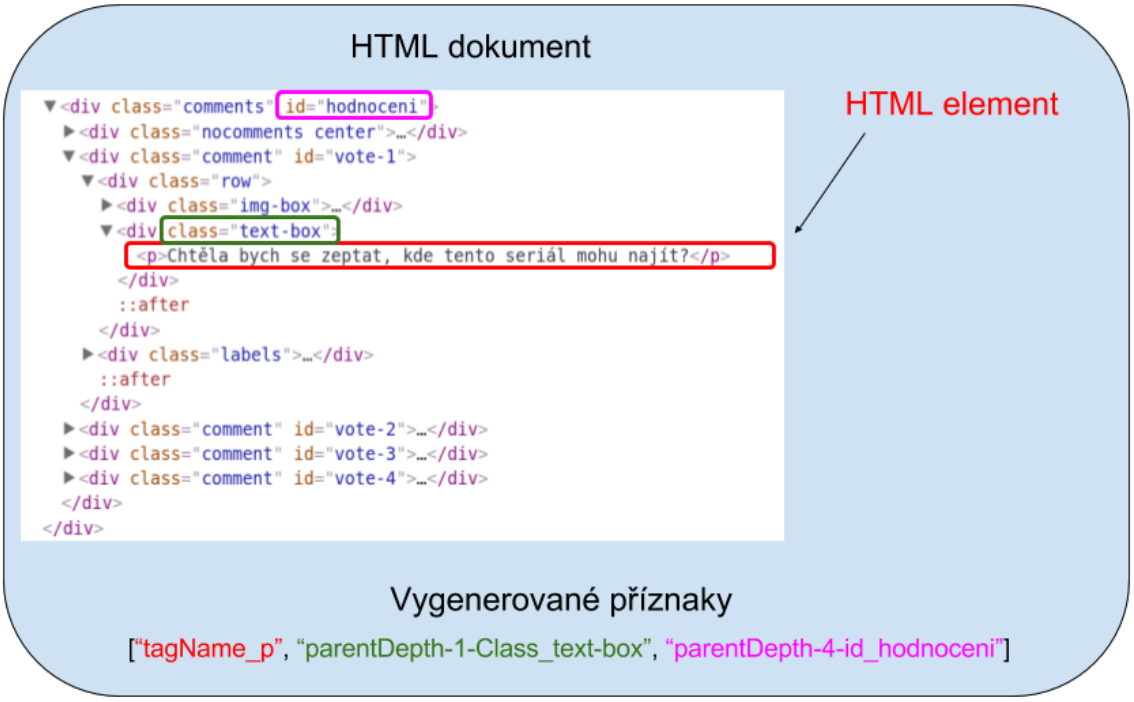
Obr. 3.2: Blokové schéma generování příznaků

Princip fungování generátoru příznaků je zobrazen na obr. 3.3. V ukázce je vygenerované jméno tagu HTML elementu a název tříd dvou nadřazených elementů.

Pomocí knihovny *Jsoup* je prováděno parsování HTML DOM dokumentu do JAVA objektů. Metody pro generování příznaků májí vstupní paramet HTML element. Výstupem generátorů příznaků jsou všechny vygenerované příznaky.

3.1.1 Generování názvů tagů rodičovských elementů

Prvním generátorem je generování jména tagů rodičovských elementů. Pseudokód algoritmu je zobrazen na výpisu algoritmu číslo 1. Vstupem algoritmu je HTML element a výstupem jsou všechny vygenerované příznaky. Generátor prochází strukturu HTML DOM dokumentu do přednastavené hloubky pomocí konstatnty *MAX_DEPTH*. Hloubka zanoření je nastavena na deset, tato hodnota je kompromisem mezi velikostí výsledné databáze a dostatečným počtem příznaků pro trénování klasifikačního modelu a následnou klasifikaci. Algoritmus postupně prochází rodičovské elementy a přidává příznak do množiny všech příznaků k danému vzorku. Příznak se skládá z prefixu, hloubky zanoření a názvu tagu. Při neexistujícím rodičovském elementu se algoritmus ukončí. Nejčastější výskyt komentářů je v tagu *p*, *div* nebo *span*.



Obr. 3.3: Princip generátoru příznaků

3.1.2 Generování atributů rodičovských elementů

Velice zásadním algoritmem je generátor atributů rodičovských elementů. Generátor generuje příznaky pomocí atributů nadřazených tagů. Z každého atributu se stane jeden příznak. Pseudokód algoritmu je zobrazen na výpisu algoritmu číslo 2. Algoritmy všech čtyř generátorů jsou velmi podobné, liší se pouze blok příkazů ve *While* smyčce. Tento algoritmus se odlišuje od předchozího přidáním smyčky *foreach*, která prochází všechny atributy z HTML elementu. Příznak se v tomto generátoru skládá z prefixu, hloubky zanoření, jména atributu a hodnoty atributu. Z jednoho atributu může být vytvořeno i více příznaků, například element obsahuje více CSS tříd, kde z každé třídy se vygeneruje jeden příznak. Atributem může být *třída*, *ID*, *style*, *value* a dalších mnoho atributů. Komentář se často vyskytuje v tagu s třídou „comment“, „comment-body“ a s mnoha dalšími specifickými *třídami* a *ID* pro komentáře.

3.1.3 Generování atributů sousedících elementů

Tento generátor generuje příznaky pomocí atributů sousedních HTML elementů, tedy elementů ve stejném zanoření v HTML DOM dokumentu. Pseudokód algoritmu je zobrazen na výpisu algoritmu číslo 3. Algoritmus je podobný jako generátor atributů rodičovských elementů. Je zde pouze přidaná smyčka *foreach*, která

Input: HTML *element*
Output: vygenerované *příznaky*

$MAX\_DEPTH \leftarrow 10$;
$i \leftarrow 0$;
$prefix \leftarrow$ "$parrentTagName - depth -$ ";

while $i < MAX\_DEPTH$ **do** `// procházení DOM dokumentu`
`do přednastavené hloubky`
    příznak $\leftarrow prefix + i + element.getTagName()$;
    přidání příznaku do množiny všech příznaků;
    if *neexistuje-li rodičovský element* **then**
        break;
    end
    element $\leftarrow$ nadřazený element;
    $i++$;
end

Algoritmus 1: Pseudokód generátoru názvu tagů rodičovských elementů

Input: HTML *element*
Output: vygenerované *příznaky*

$MAX\_DEPTH \leftarrow 10$;
$i \leftarrow 0$;
$prefix \leftarrow$ "$parrentAttribute - depth -$ ";

while $i < MAX\_DEPTH$ **do** `// procházení DOM dokumentu`
`do přednastavené hloubky`
    foreach *procházení všech atributů elementu* **do**
        příznak $\leftarrow prefix + i + attribut.getName() + attribut.getValue()$;
        přidání příznaku do množiny všech příznaků;
    end
    if *neexistuje-li rodičovský element* **then**
        break;
    end
    element $\leftarrow$ nadřazený element;
    $i++$;
end

Algoritmus 2: Pseudokód generátoru atributů rodičovských elementů

prochází všechny sousední HTML elementy. Sousední atributy mohou mít například třídu „comment-title“, „comment-autor“ a dalších mnoho atributů spojených s komentáři.

Input: HTML *element*
Output: vygenerované *příznaky*

$MAX\_DEPTH \leftarrow 10$;
$i \leftarrow 0$;
$prefix \leftarrow "siblingAttribute - depth - "$;

while $i < MAX\_DEPTH$ **do** `// procházení DOM dokumentu do přednastavené hloubky`
  foreach *procházení všech sousedních elementů* **do**
    foreach *procházení všech atributů elementu* **do**
      $příznak \leftarrow prefix + i + attribut.getName() + attribut.getValue()$;
      přidání příznaku do množiny všech příznaků;
    end
  end
  if *neexistuje-li rodičovský element* **then**
    break;
  end
  element $\leftarrow$ nadřazený element;
  $i++$;
end

Algoritmus 3: Pseudokód generátoru atributů tagů sousedících elementů

3.1.4 Generování názvu tagu u sousedících elementů

Generátor generuje název tagu sousedících elementů. Pseudokód algoritmu je zobrazen na výpisu algoritmu číslo 4. Tento algoritmus prochází všechny sousední elementy a generuje příznak pomocí prefixu a hloubky zanoření a jména daného sousedního tagu.

3.2 Vytvoření trénovací databáze

Prvním krokem při sémantickém vyhledávání komentářů na webu s pomocí klasifikátorů je vytvořit trénovací databázi. Tato datábáze je potřeba pro natrénování

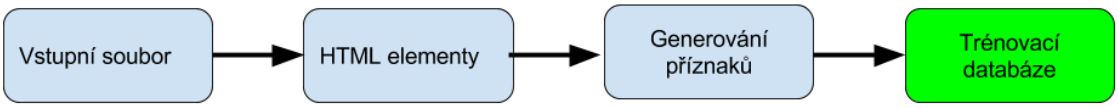
Input: HTML *element*
Output: vygenerované *příznaky*

$MAX\_DEPTH \leftarrow 10$;
$i \leftarrow 0$;
$prefix \leftarrow$ "$siblingTagName - depth -$ ";

while $i < MAX\_DEPTH$ **do** `// procházení DOM dokumentu do přednastavené hloubky`
  foreach *procházení všech sousedních elementů* **do**
    příznak $\leftarrow prefix + i + element.getTagName()$;
    přidání příznaku do množiny všech příznaků;
  end
  if *neexistuje-li rodičovský element* **then**
    konec;
  end
  element $\leftarrow$ nadřazený element;
  $i++$;
end

Algoritmus 4: Pseudokód generátoru názvu tagů sousedních elementů

klasifikačního modelu. Princip tvoření trénovací databáze spočívá v manuálním vyhledávání webových stránek obsahující komentáře. Vytvoření kvalitní trénovací databáze zabere hodně úsilí a času a má zásadní vliv na úspešnost klasifikace.



Obr. 3.4: Blokové schéma pro vytvoření trénovací databáze

3.2.1 Vstupní data aplikace

Vstupní data aplikace jsou uložena v datovém souboru JSON (JavaScript Object Notation), což je formát určený pro výměnu informací. Tento formát je snadno čitelný a zapisovatelný člověkem a zároveň jednoduše strojově analyzovatelný i generovatelný. Pro práci se souborem JSON je využita knihovna *jackson*. Příklad jednoho vstupního linku je ukázán ve výpisu kódu 3.1. Každý link obsahuje URL adresu

webové stránky, pozitivní CSS selektor (všechny komentáře) a negativní CSS selektor (HTML element neobsahující komentář). Pomocí CSS selektorů se vybírají HTML elementy.

Výpis 3.1: Příklad vstupních dat ve formátu JSON

```
{
  "links":[
    {
      "link": "http://stalkerky.cz/agata-neco-chystaaa/",
      "positive": [
        "article.comment-body␣div␣p"
      ],
      "negative": [
        "li.menu-item␣a"
      ]
    }
  ]
}
```

3.2.2 Načtení webové stránky

Pro vygenerování příznaků si potřebujeme nejprve načíst celou HTML stránku do aplikace. Stránka se načítá s využitím knihovny *Jsoup*, která zašle požadavek na server pomocí HTTP metody GET. Přístup k webové stránce je velmi rychlý (pohybujeme se v řádech stovek ms), ale neprovede se vykonávání javaScript funkcí. JavaScript se provádí na straně klienta (webový prohlížeč). V mnoha případech se načítají komentáře pomocí javaScriptu, tedy javaScript se dodatečně dotazuje serveru na další komponenty webu. Komentáře se mohou načítat ze služeb třetích stran, zde lze uvést například *facebook plugin*. Řešením by mohlo být použití Selenium serveru s prohlížečem, který reálně simuluje prohlížení webových stránek, tedy včetně vykonávání javaScript kódu na straně klienta. Jako prohlížeč lze použít *PhantomJS*. Jedná se o takzvaný „prohlížeč bez displeje“, který má zabudované klasické jádro WebKit. WebKit je rendrovací jádro prohlížeče. Pro bezproblémový běh aplikace na různých počítačích a platformách byl Selenium server z programu odstraněn. Selenium server se používá například pro automatizované testování webových aplikací, což je simulování procházení webových stránek uživateli.

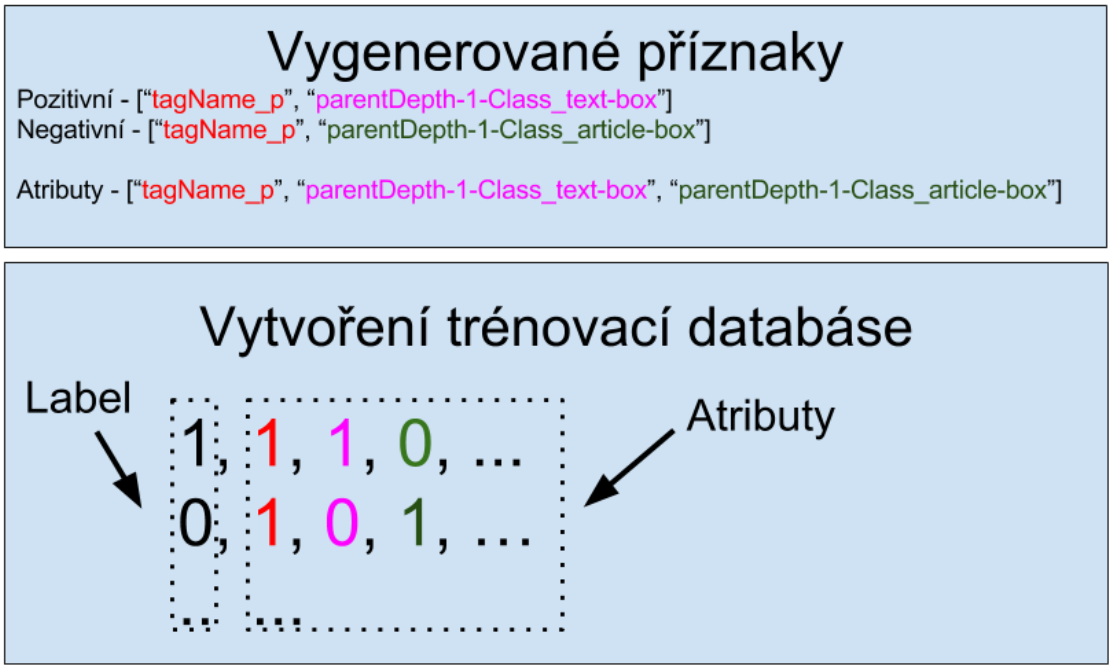
3.2.3 Aplikace generátoru příznaků

Z každé načtené webové stránky se vyberou pomocí CSS selektoru pozitivní (komentáře) a negativní HTML elementy (HTML element, který neobsahuje komentář). Počet pozitivních a negativních dat v trénovací množině by měl být přibližně stejný. Tento poměr je důležitý pro co nejlepší klasifikaci a celkovou úspěšnost sémantického vyhledávání komentářů na webových stránkách. Následně se na každý vybraný HTML element aplikují generátory příznaků. Výsledek z generátoru se ukládá do vnitřní datové struktury, ze které se následně vygeneruje trénovací databáze.

3.2.4 Výsledná trénovací databáze

Způsob vytvoření a formát trénovací databáze byl přizpůsoben programu RapidMiner, ve kterém se pomocí připraveného procesu vytváří klasifikační model. Jako výstupní formát souboru byl vybrán formát CSV (Comna separated values). CSV je jednoduchý souborový formát určený pro výměnu tabulkových dat. Soubor ve formátu CSV se skládá z řádků, ve kterých jsou hodnoty odděleny čárkou. Čárku jako oddělovač můžeme nahradit i jinými znaky. To je hojně využíváno například v České republice, jelikož čárka zde může značit oddělovač desetinných míst čísla. Díky jednoduchosti a čitelnosti i bez speciálního softwaru se tento formát používá pro výměnu informací mezi různými systémy a aplikacemi. Pro složitější strukturu informací se používájí modernější formáty souborů, nejznámějšími a nejvíce používanými jsou JSON (JavaScript Object Notation), XML (eXtensible Markup Language) a YAML (Ain't Markup Language). Tyto formáty se používají především pro serializaci strukturovaných dat. Serializace je proces převedení libovolného objektu do sériové (jednorozměrné struktury).

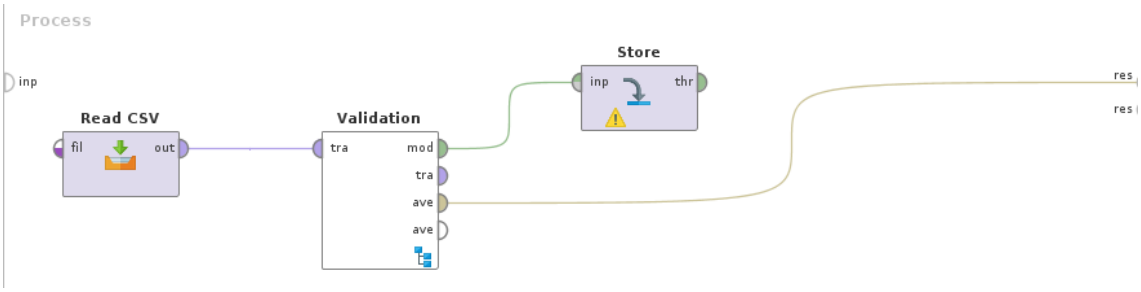
Příklad generování trénovací databáze je vysvětlen na obr 3.5. Jeden řádek CSV souboru reprezentuje informace o jednom HTML elementu. První sloupec je vždy „label“, který udává zda se jedná o pozitivní (komentář) nebo negativní (jiný HTML element) záznam v trénovací databázi. Dalším krokem je vytvořit množinu všech možných vygenerovaných příznaků ze všech HTML elementů. Další sloupce (na obr. 3.5 označené atributy) udávají výskyt konkrétních příznaků u daného elementu. Pokud element obsahuje příznak, nastaví se do souboru logická jednička, v opačném případě logická 0. Výsledný soubor obsahuje přibližně 2000 řádků (různých HTML elementů) a 20 000 sloupců (unikátních příznaků ze všech elementů). Velikost trénovací databáze je přibližne 120 MB.



Obr. 3.5: Generování trénovací databáze

3.3 Trénování modelu pomocí klasifikátorů

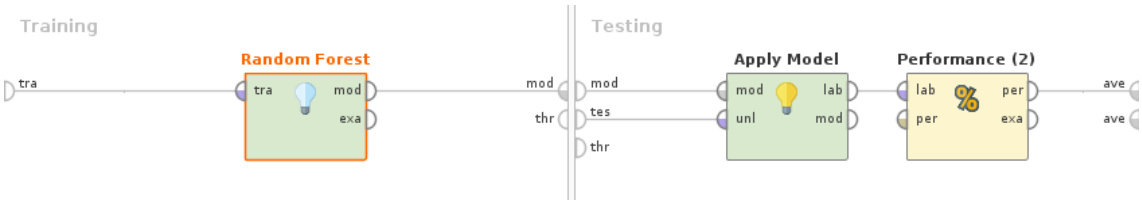
K trénování modelu a klasifikaci dat je využit už dříve zmíněný program RapidMiner, kde se pomocí navrhnutého procesu trénuje klasifikační model. V této kapitole budou popsány a porovnány čtyři různé klasifikátory pro klasifikaci dat. U jednotlivých klasifikátorů budou rozebrány a vysvětleny zejména jejich nastavovací parametry. Po otestování všech čtyř klasifikátorů bude vybrán klasifikátor s nejvyšší úspěšností zařazovaní HTML elementů do správných tříd, tedy zda se jedná o komentář nebo ne. Na obr. 3.6 je zobrazen vytvořený proces pro klasifikaci trénovací databáze.



Obr. 3.6: Navržení procesu v Rapid Miner

Proces začíná blokem *Read CSV*, ve kterém se načte trénovací databáze. Je důležité nastavit některé důležité parametry tohoto bloku. Pomocí parametru *data set meta data information* se nastavují typy atributů. První atribut se musí nastavit jako *label* (označuje třídu dat) a datový typ *binominal*. *Binominal* je datový typ umožňující přesně dvě hodnoty, například *true / false* nebo *1 / 0*.

Načtená data pokračují do bloku *Validation*, kde se rozdělují na data trénovací a testovací. Rozdělení dat je důležité pro vyhodnocení úspěšnosti jednotlivých klasifikátorů. Parametrem *number of validations* se nastavuje počet podskupin pro křížovou validaci. Křížová validace je metoda určená ke zjištění úspěšnosti vytvořeného klasifikačního modelu. Principem je rozdělit trénovací data na několik množin. Jedna množina slouží jako testovací, zbylé podmnožiny slouží jako trénovací. Tento proces se několikrát opakuje, pokaždé s jinou testovací množinou. Druhým nastavovacím parametrem je *sampling type*, kterým nastavujeme způsob rozdělení trénovací databáze do podmnožin. Pro naši trénovací databázi je nejvhodnější nastavit *linear sampling*, což nám rozdělí trénovací data po blocích. Tímto nastavením zamezíme výskytu dvou podobných pozitivních HTML elementů v obou množinách (trénovací i testovací) a tím i zkreslení úspěšností klasifikátorů. Vnitřní architektura bloku je zobrazena na 3.7, kde vidíme rozdělení na trénovací a testovací část. V trénovací části se postupně vyměňují klasifikátory, čímž se postupně zjišťuje úspěšnost každého z nich. V testovací čási se klasifikují data s využitím natrénovaného modelu pomocí bloku *Apply Model* a pomocí bloku *Performance* je zjišťována daná úspěšnost.



Obr. 3.7: Navržení procesu v Rapid Miner

Posledním blokem je *Store*, který ukládá natrénovaný model pro následné použití v algoritmu pro sémantické vyhledávání komentářů na webových stránkách.

3.3.1 Klasifikátory

S využitím trénovací databáze jsou natrénovány čtyři různé klasifikátory a je vyhodnocena jejich úspěšnost.

Tab. 3.1: Výsledky klasifikace pomocí SVM

typ jádra	**úspěšnost** [%]
rbf	69,84
linear	66,70

Tab. 3.2: Výsledky klasifikace pomocí rozhodovacího stromu

criterion	**maximal depth**	**úspěšnost** [%]
gan_ratio	20	76,04
information_gain	20	77,68
gini_index	20	78,34
gan_ratio	-1	76.04

SVM

Nastavovací parametry jsou:

- *Typ jádra (kernel type)* - Slouží pro nastavení jádrové transformace. Pro naši úlohu je nejlepší zvolit parametr RBF (Radial Base Functions).
- *C* - Jedná se o konstantu, která udává toleranci pro chybovou klasifikaci. Čím vyšší hodnota C, tím „měkčí“ hranice mezi prvky.
- *epsilon* - Tento parametr určuje necitlivost konstant.
  V tabulce 3.1 vidíme výsledek úspěšnosti klasifikátoru SVM.

Rozhodovací strom (Decision tree)

Rozhodovací strom, jehož základní nastavovací parametry jsou:

- *Kritérium (criterion)* - Zde se nastavuje typ rozhodovacího parametru, který je více popsaný v teoretické části.
- *Maximální hloubka (maximal depth)* - Nastavuje se maximální hloubka stromu.
- *Spolehlivost (confidence)* - Tento parametr určuje úroveň spolehlivosti použitou pro výpočet pesimistické chybovosti prořezávání.
- *Minimální zisk (minimal gain)* - Jedná se o zisk uzlu, který se počítá před rozdělením. Uzel je rozvětven, pokud je zesílení větší než minimální zisk.
- *Minimální velikost listu (minimal leaf of size)* - Parametr určuje velikost koncového uzlu a je udáván jako celé číslo.

U rozhodovacího stromu jsem měnil parametr *criterion*, který nastavuje typ rozhodovací podmínky. Výsledky úspěšnosti se v závislosti na měnícím se parametru nemění. Klasifikace dosahuje vysoké úspěšnosti okolo 80 %. Výsledky jsou v tabulce 3.2.

Tab. 3.3: Výsledky klasifikace pomocí K-NN

parametr K	**úspěšnost** [%]
1	69,65
2	70,16
3	70,03
4	68,25
10	67,2

Tab. 3.4: Výsledky úspěšnosti klasifikátoru náhodný les

počet stromů	**úspěšnost** [%]
2	68,74
5	67,87

K-NN

Jedná se o algoritmus k-nejbližších sousedů. Základní nastavovací parametry v programu RapidMiner jsou:

- *k* - Parametr udává počet nejblíže vyhledávaných prvků z trénovací množiny.
- *Typ měření (measure types)* - Zde nastavujeme typ, podle kterého se vyhlédávají nejbližší prvky.

V tabulce 3.3 vidíme úspěšnost klasifikace v závislosti na parametru *k*. Parametr *typ měření* neměl vliv na úspěšnost klasifikace.

Náhodný les (Random Forest)

Tento klasifikátor obsahuje stejné parametry jako *rozhodovací strom*, ale navíc přidává některé další parametry:

- *Počet stromů (number of tree)* - Parametr udává počet generovaných stromů. Nejčastěji se nastavuje v rozmezí 0 - 50 stromů.

U klasifikace pomocí *náhodného lesa* jsem nastavoval jako parametr počet vytvářených stromů, který neměl na výsledek úspěšnosti vliv. Výsledek klasifikace je v tabulce 3.4.

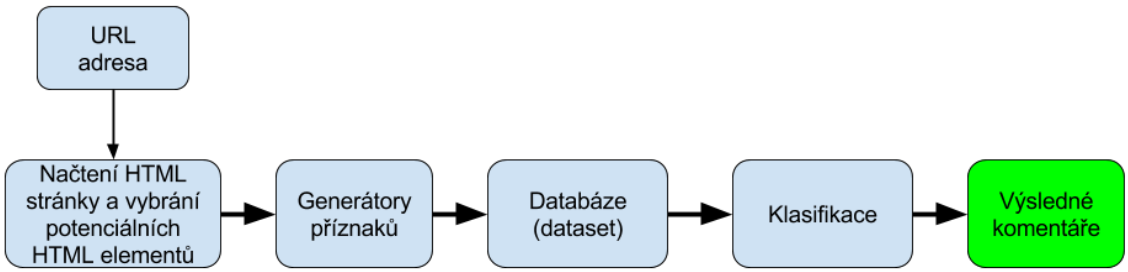
3.3.2 Srovnání úspěšnosti klasifikátorů

Nejvyšší úspěšnost při strojovém učení byla dosáhnuta u klasifikátoru rozhodovací strom. Úspěšnost se zde pohybovala mezi 75 a 80 %. Jako druhý nejlepší klasifikátor

se osvědčil k-NN, kde se úspěšnost pohybovala okolo 70 %. U posledních dvou klasifikátorů se úspěšnost klasifikace pohybovala lehce nad 50 %, což pro sémantické vyhledávání komentářu je velmi málo.

3.4 Vyhledávání komentářů

Samotné sémantické vyhledávání komentářů je poskládání předešlých kroků a přidání procesu pro klasifikaci dat (datasetu). Konkrétně se v této části používá z předešlých částí generátor příznaků, generování trénovací databáze a klasifikační model. Na obr. 3.8 je zobrazeno blokové schéma postupu při vyhledávání komentářů.



Obr. 3.8: Blokové schéma vyhledávání komentářů na webové stránce

Vstupním parametrem programu je URL adresa webové stránky s komentáři. Ze znalosti vstupního parametru se načte HTML stránka do paměti programu. Následně se pomocí CSS selektoru vyberou potenciální HTML elementy s komentáři. Na všechny vybrané elementy se aplikuje generátor příznaků a vygeneruje CSV soubor podobným způsobem jako při generování trénovací databáze popsané v kapitole 3.2.4. Následně se klasifikují data s využitím natrénovaného modelu. Výsledkem celého programu je výstupní soubor s komentáři.

Další podkapitoly vyhledávání komentářů budou obsahovat detailnější popis jednotlivých bloků z obr. 3.8. Nejvíce bude rozebrán blok klasifikace dat, která využíván natrénovaný model.

3.4.1 Vybrání potenciálních HTML elementů

Z URL adresy webové stránky předané do programu jako vstupní parametr se načte HTML stránka do paměti programu. Nejdůležitějším krokem v tomto bloku je vybrat správné potenciální HTML elementy, které by mohly obsahovat komentáře z webové stránky. Jde o vybrání rozumné množiny elementů, ve které se nacházejí všechny komentáře a co nejméně ostatních HTML elementů z webové stránky.

Ze získaných znalostí při vytváření trénovací databáze, které spočívalo pomocí CSS selektorů vybrat HTML elementy s komentáři z webové stránky, jsem zvolil následující tři filtry.

- Jméno tagu HTML elementu - Ze získaných informací z tvorby trénovací databáze bylo zjištěno, že HTML elementy s komentáři se nacházejí pouze v tagech *p*, *div* nebo *span*. První filtr slouží k vybrání pomocí CSS selektoru pouze elementů s názvy těchto tagů.
- Počet vnitřních elementů - Dalším potřebným filtrem je filtrace podle počtu vnitřních tagů. Například HTML element s 200 vnitřními tagy nebude komentář. Většina elemetů s tagy obsahuje maximálně několik tagů, může se jednat například o odkaz, obrázek nebo zvýraznění písma uvnitř komentáře.
- Délka vlastního textu v elementu - Posledním filtrem je omezení podle délky vlastního textu uvnitř elementu. V tomto případě je odfiltrována spodní hranice. Jedná se zejména o prázdné HTML elementy nebo krátké doplňující informace o komentáři (datum přidání, jméno autora a mnoho dalších).

3.4.2 Aplikace Generátoru příznaků

Na vybrané potenciální HTML elementy se aplikuje stejný generátor jako při generování trénovací databáze. Z výsledné množiny příznaků jednoho vzorku jsou důležité pouze příznaky, které se nacházely v trénovací databázi a pomocí kterých byl natrénován klasifikační model. Generátor příznaků je detailně popsán v kapitole 3.1.

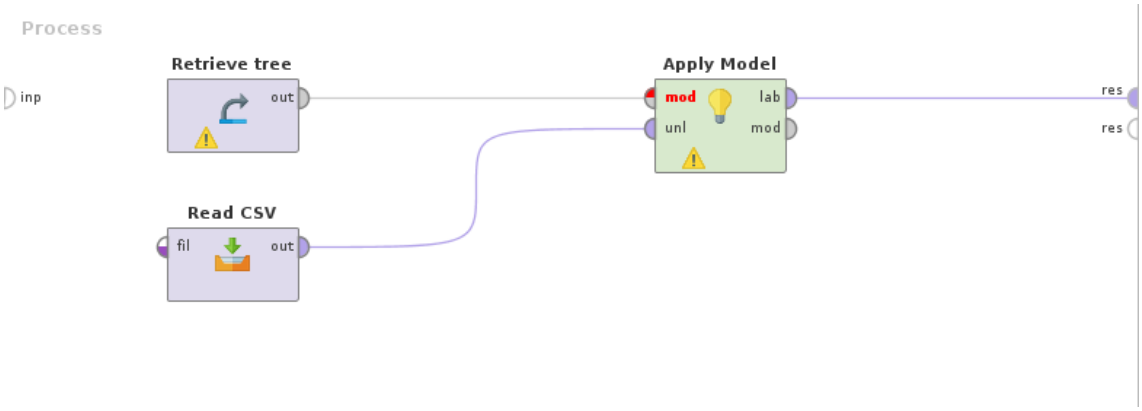
3.4.3 Vygenerování databáze (dataset)

Vytvoření CSV souboru pro predikci pomocí natrénovaného modelu je podobné jako při generování trénovací databáze popsané v kapitole 3.2.4. Jediný rozdíl je načtení všech atributů z fáze generování trénovací databáze. Tímto zajistíme stejný počet sloupců CSV souboru během obou fází běhu programu, jak při trénování klasifikačního modelu, tak i při konečném vyhledávání komentářů. Velikost souboru záleží na velikosti a specifikaci webové stránky. Čím je větší webová stránka, tím se vybere více potenciálních HTML elementů k predikci pomocí umělé inteligence.

3.4.4 Klasifikace vzorků z potenciálních HTML elementů

Na vstup klasifikace dle natrénovaného modelu vstupují data ve formátu CSV. Klasifikace probíhá na základě vytvořeného procesu v programu RapidMiner, který je zobrazen na obr. 3.9. Proces se skládá pouze ze tří bloků. Hlavním blokem je *Apply Model*, který za použití natrénovaného modelu a databáze (dataset) klasifikuje

vzorky HTML elementů do tříd. Databáze v souborovém formátu CSV se načítá pomocí bloku *Read CSV* a již natrénovaný klasifikační model se načítá pomocí bloku *Retrieve*. Pro implementaci procesu do programu JAVA je nutné proces vyexportovat do XML souboru. Proces je v RapidMineru uložen primárně v XML souboru a při exportu proto nedochází k žádné transformaci mezi datovými strukturami. Celá klasifikace probíhá automaticky a všechny parametry bloků jsou přednastaveny a uloženy v procesu, v našem případě tedy v XML souboru. Tato fáze je v porovnání s trénováním klasifikačního modelu relativné rychlá (v řádu několika jednotek sekund). Záleží zde na počtu potenciálních HTML elementů a počtu všech atributů z fáze generování trénovací databáze.



Obr. 3.9: Schéma procesu pro vyhledávání komentářů

3.4.5 Výsledné komentáře

Výsledkem sémantického rozpoznání komentářů na webových stránkách je vygenerovaný textový soubor s výslednými komentáři. Textový souboru je vygenerovaný ve formátu JSON, který je vhodný pro jednuduché počítačové zpracování. Tento formát je také dobře čitelný člověkem.

3.5 Použité programy a nástroje

Hlavním cílem bylo navrhování a vytváření algoritmu pro sémantické rozpoznávání komentářů na webových stránkách. K vytvoření výsledného programu bylo použito několik technologií a nástrojů, které musejí fungovat jako jeden funkční celek. V této kapitole se nachází popis základních použitých technologií. Nejvíce přínosný byl program RapidMiner, ze kterého se využívají algoritmy strojového učení.

3.5.1 Programovací jazyk a prostředí

Při navrhování programu byl vybrán programovací jazyk JAVA. Hlavním důvodem byla dostupná knihovna RapidMiner pro využití umělé inteligence a klasifikátorů v programu a také výborná knihovna *Jsoup* pro prácí s HTML DOM dokumentem a HTML elementy. Dalším velkým přínosem je na platformě nezávislý přenos výsledného JAR souboru.

Pro správu a sestavení aplikace byl použit systém Maven. Jeho využitím odpadá závislost na použitém IDE (Integrated Development Environment), jelikož veškeré informace ke kompilaci a sestavení programu jsou uloženy v souboru POM (Project Object Model). Hlavní funkcí systému Maven je správa závislostí mezi knihovnami. Knihovny se stahují z centrálního repozitáře umístěného na adrese `http://search.maven.org/`, ale lze nastavit také firemní nebo týmový repozitář. Pomocí pluginu *maven-assembly-plugin* lze automaticky vytvořit balíček se všemi závislostmi. Výsledný soubor je ve formátu JAR (Java Archive), jedná se o kompresní souborový formát k seskupení *Java Class*, souvisejících metadat a zdrojů do jednoho souboru. JAR soubory jsou přenositelné na různé operační systémy. Vývoj celého programu byl verzován pomocí verzovacího nástroje GIT. [1] Veřejný repozitář je umístěný na GitHubu na adrese `https://github.com/stritesky/DP`.

3.5.2 RapidMiner

K trénování klasifikačního modelu a následné klasifikaci vzorků do tříd byl využit program RapidMiner. Použití tohoto programu bylo velmi jednoduché a velmi urychlilo práci při využívání algoritmů strojového učení. Velkým záporem je nedostatečná dokumentace pro implementování procesu do programovacího jazyka JAVA.

[1] GIT - distribuovaný systém pro správu verzí

4 ZÁVĚR

Teoretická část této diplomové práce obsahuje popisy algoritmů metod strojového učení s učitelem. Jsou zde popsány klasifikátory rozhodovací strom, SVM, náhodný les a k-NN. Dále jsou v teoretické části zmíněny webové technologie a popsaný program RapidMiner, který je využíván v praktické části pro metody strojového učení.

Cílem projektu bylo vytvořit algoritmus, který pomůže s automatickým rozpoznáváním komentářů na webových stránkách. Součástí práce je vytvoření databáze pro trénování a testování klasifikátorů, ověření přesnosti a prezentace výsledků.

Sémantické vyhledávání je řešené pomocí umělé inteligence a využitím různých klasifikátorů na predikci. V práci byly porovnávány celkem čtyři klasifikátory, jmenovitě SVM, rozhodovací strom, k-NN, a náhodný les. Nejvyšší úspěšnost byla s použitím klasifikátoru rozhodovací strom. Úspěšnost se zde pohybovala okolo 80 %. Pro práci s klasifikátory byl vybrán program RapidMiner, ve kterém se navrhovaly procesy pro trénování modelu a následnou klasifikaci vzorků HTML elementů. Výsledný proces lze vyexportovat do XML souboru a dále s ním pracovat pomocí programovacího jazyka JAVA. Běh programu je rozdělen do tří fází. První fází je vygenerování trénovací databáze pomocí množiny webových stránek a CSS selektorů pro vyhledání komentářů na webové stránce. Další fází je natrénování co nejlepšího klasifikačního modelu pomocí trénovací databáze. Tento proces je manuální a je prováděn v grafickém prostředí programu RapidMiner. Výsledný program pro vyhledávání komentářů na webových stránkách je plně automatický, programu stačí předat pouze URL adresu webové stránky, ze které se mají vyhledat komentáře. Výstupem programu je soubor ve formátu JSON obsahující vyhledané komentáře. Výsledný program je ve formátu JAR, který je přenositelný na několik operačních platforem.

Hlavním přínosem práce je sémantické vyhledávání komentářů z webových stránek. Toto řešení může následně posloužit k nasazení na server a sloužit ke generování objemných databází komentářů k následnému zpracování. Zpracováním je myšleno vyhledávat v komentářích různé užitečné informace s spojovat závislosti mezi nimi.

Algoritmus pro generování trénovací databáze pro následnou klasifikaci vzorků HTML elementů je navržen univerzálně a může se použít pro sémantické vyhledávání HTML elementů s nově definovaným kontextem.

LITERATURA

[1] BERNERS-LEE, Tim J.*The world-wide web*. Computer networks and ISDN systems, 1992, 25.4-5: 454-459.

[2] *Hypertext Transfer Protocol – HTTP/1.0* [online]. 1996 [cit. 2016-12-12]. Dostupné z: `https://tools.ietf.org/html/rfc1945`

[3] Hypertext Transfer Protocol – HTTP/1.1. *The Internet Engineering Task Force* [online]. 1999 [cit. 2016-12-12]. Dostupné z: `https://tools.ietf.org/html/rfc2616`

[4] Hypertext Transfer Protocol Version 2 (HTTP/2). *The Internet Engineering Task Force* [online]. 2015 [cit. 2016-12-12]. Dostupné z: `https://tools.ietf.org/html/rfc7540`

[5] What is the Document Object Model? *The World Wide Web Consortium* [online]. [cit. 2016-12-12]. Dostupné z: `https://www.w3.org/TR/WD-DOM/introduction.html`

[6] *Mozilla Developer Network* [online]. [cit. 2017-04-22]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors`

[7] BERKA, Petr. Současné trendy um2lé inteligence. *Acta Oeconomica Pragensia*. 2003, **8**(11), 41-46.

[8] HONZÍK, Petr. *Strojové učení*. Brno: FEKT Vysokého učení technického v Brně, 2006.

[9] BURGET, Radim. *Teoretická informatika*. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Purkyňova 118, 612 00 Brno, 2013. ISBN 978-80-214-4897-1.

[10] Tong, Simon, and Daphne Koller. *Support vector machine active learning with applications to text classification*. Journal of machine learning research 2.Nov (2001): 45-66. APA

[11] CHAPELLE, Olivier; HAFFNER, Patrick; VAPNIK, Vladimir N. *Support vector machines for histogram-based image classification*. IEEE transactions on Neural Networks, 1999, 10.5: 1055-1064.

[12] CORTES, Corinna; VAPNIK, Vladimir. *Support-vector networks*. Machine learning, 1995, 20.3: 273-297.

[13] KLASCHKA, Jan; KOTRČ, Emil. Klasifikační a regresní lesy. ROBUST 2004. Sborník prací 13. letní školy JČMF, 2004.

[14] AMIT, Yali; GEMAN, Donald. *Shape quantization and recognition with randomized trees*. Neural computation, 1997, 9.7: 1545-1588.

[15] BREIMAN, Leo, et al. *Classification and regression trees*. CRC press, 1984.

[16] QUINLAN, J.. Ross . *Induction of decision trees*. Machine learning, 1986, 1.1: 81-106.

[17] BREIMAN, Leo. *Bagging predictors*. Machine learning, 1996, 24.2: 123-140.

[18] FREUND, Yoav; SCHAPIRE, Robert; ABE, N. *A short introduction to boosting*. Journal-Japanese Society For Artificial Intelligence, 1999, 14.771-780: 1612.

[19] BREIMAN, Leo. *Random forests*. Machine learning, 2001, 45.1: 5-32.

[20] ELKAN, Charles. Nearest neighbor classification. elkan@ cs. ucsd. edu‖,‖ January, 2011, 11: 3.

[21] ALTMAN, Naomi S. *An introduction to kernel and nearest-neighbor nonparametric regression*. The American Statistician, 1992, 46.3: 175-185.

[22] Rapid Miner [online]. [cit. 2016-12-14]. Dostupné z: https://rapidminer.com/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
SSL	Secure Sockets Layer
DOM	Document Object Model
CSS	Cascading Style Sheets
SVM	Support vector machine
TDIDT	Top Down Induction of Decision Trees
k-NN	k-Nearest Neighborss
AGPL	Affero General Public License
XML	eXtensible Markup Language
YAML	Ain't Markup Language
JAR	Java ARchive
POM	Project Object Model
JSON	JavaScript Object Notation
CSV	Comma-separated values
TCP	Transmission Control Protocol
RBF	Radial Base Functions

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

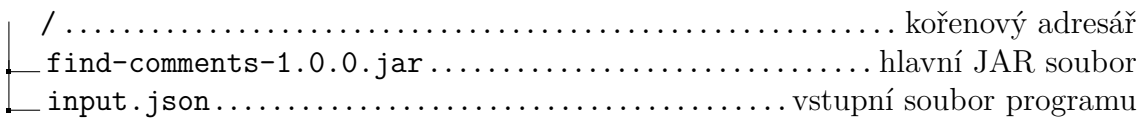
- `/` ..... kořenový adresář přiloženého CD
  - `stritesky_radek.pdf` ..... diplomová práce
  - `DP-master.zip` ..... zdrojové kódy z GitHubu
  - `trainModel.rmp` ..... proces pro trénování modelu pomocí trénovací databáze
  - `program` ..... složka s JAR souborem a zdroji
    - `find-comments-1.0.0.jar` ..... hlavní JAR soubor
    - `repository` ..... složka se zdroji, které jsou potřebné ke klasifikaci
      - `dataset` ..... složka pro uložení datasetu - automaticky ukládá program
      - `model` ..... složka obsahuje natrénovaný model
        - `model.ioo` ..... natrénovaný model
      - `attributes.json` ... soubor obsahuje včechny atributy z fáze generování trénovací databáze
      - `process.rmp` ..... proces pro klasifikaci - program spouští automaticky
    - `input.json` ..... vstupní soubor programu
    - `data-example` ..... složka s výstupními soubory z fáze generování trénovací databáze
      - `attributes.json` ..... soubor obsahuje množinu všech vygenerovaných atributů
      - `result.csv` ..... vygenerová trénovací databáze (dataset)
    - `resultComments-2017.05.23.20.15.45.json` ..... výstupní soubor při vyhledávání komentářů

B INSTRUKCE PRO PRÁCI S PROGRAMEM

V této kapitole se nachází detailní instrukce pro spuštění programu. Program se spouští pomocí souboru JAR. Zdroje k běhu programu nejsou umístěné v JAR souboru z důvodů obměny natrénovaného modelu a souvisejících souborů. Přikládat k spustitelnému souboru další zdroje se může zdát nadbytečné, ale výsledný program má mnohem větší potenciál využití. Potenciálem není myšleno jenom zkvalitňování a rozšiřování trénovací databáze, ale i možnost naučit program klasifikovat úplně jiné HTML elementy (jiný kontext z webové stránky).

B.1 Vygenerování trénovací databáze

První fází programu je vytvoření trénovací databáze. Jako zdroj je povinný vstupní soubor *input.json*, který obsahuje informace o všech webových stránkách zahrnutých v trénovací databázi. Vstupní parametr JAR souboru je *learn*, který definuje mód programu pro generování trénovací databáze. Výstupem je trénovací databáze ve formátu CSV a všechny atributy použité v trénovací databázi ve formátu JSON. Na obr. B.1 je zobrazena adresářová struktura spuštění programu v módu generování trénovací databáze.



Obr. B.1: Adresářová struktura při spuštění programu v módu generování trénovací databáze

Ve výpisu kódu B.1 je příkaz pro spuštění souboru JAR pro vygenerování trénovací databáze.

Výpis B.1: Příkaz pro spuštění jar programu v módu generování trénovací databáze

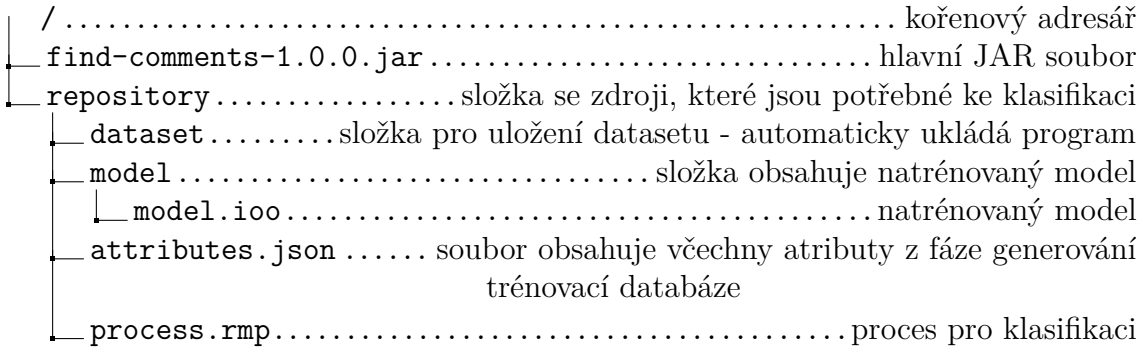
```
java -jar find-comments-1.0.0.jar learn
```

B.2 Natrénování klasifikačního modelu

Fáze mezi generováním trénovací databáze a vyhledávání komentářů na webových stránkách je řešena manuálně v programu RapidMiner. Toto řešení je zvoleno z důvodu lepšího nastavení parametrů klasifikátorů a dosáhnutí vyšší úspešnosti následné klasifikace. Je zde možné klasifikátory vyměňovat a vybrat ten s nejvyšší úspěšností predikce. Tato manuální práce se prování pouze při nové trénovací databázi.

B.3 Vyhledávání komentářů

Sémantické vyhledávání komentářů je hlavní částí a cílem diplomové práce. Tato část je plně automatická a využívá zdroje získané z předešlých kroků (natrénovaný model a seznam všech atributů). Dále je ve zdrojích uložený proces pro klasifikaci. Tento proces se při změně natrénovaného modelu nemění. Na obr. B.2 je zobrazena adresářová struktura při spouštení programu v módu vyhledávání komentářů. Strukturu je nutné dodržet, včetně pojmenování souborů. Výstupem programu je soubor ve formátu CSV, který obsahuje všechny vyhledané komentáře z dané webové stránky.



Obr. B.2: Adresářová struktura při spuštění programu v módu vyhledávání komentářů

Ve výpisu B.2 je uveden příklad spuštění JAR programu pro vyhledávání komentářů. Vstupní parametr je URL adresa webové stránky ze které chceme získat komentáře. Tímto parametrem zajistíme také běh programu v módu vyhledávání komentářů. Výstupem programu je JSON soubor s výslednými komentáři. JSON soubor je vhodný pro následné strojové , stejně tak je jednoduše čitelný člověkem.

Výpis B.2: Příkaz pro spuštění jar programu v módu vyhledávání komentářů

```
java -jar find-comments-1.0.0.jar www.example.com/article01
```