Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Optimální metody dataminingu pro zpracování semistrukturovaných medicínských dat

Plzeň 2015 Mario Kamburov

Prohlášení

Prohlašuji, že jsem diplomovou práci na téma

“Optimální metody datamining pro zpracování semistrukturovaných medicínských dat”

pod vedením vedoucího diplomové práce vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4 května 2015 Mario Kamburov

Poděkování

Chtěl bych poděkovat paní Doc. Dr. Ing. Janě Klečkové za vstřícný postoj, motivaci a cenný čas věnovaný vedení mojí diplomové práci. Dále bych chtěl poděkovat své rodině za trpělivosti a podpoře, zejména svému bratrovi, který svými znalostmi v oboru medicíny přispěl k tomuto projektu.

Abstrakt

Mario Kamburov, Optimální metody dataminingu pro zpracování semistrukturovaných medicínských dat

Diplomová práce, Plzeň 2015

Cílem mé diplomové práce bylo navrhnout řešení a vytvořit program, který by umožňoval korekce lékařských textů na základě velmi rozsáhlých a různorodých semistrukturovaných dat z lékařských zpráv. V práci teoreticky popisuji možnosti zpracování přirozeného jazyka a několik již implementovaných datamining algoritmů pro klasifikace textů. Je zde popsán princip mnou navrženého řešení, který využívá databáze pro ukládání trénovacích dat. Dále je podrobně popsána implementace v jazyce Java s napojením na databázi MySQL, PostgreSQL a IBM DB2 a provedeno ověření na vybrané kolekci medicínských dat. Na konci jsou pak předloženy obsáhlé statistiky průběhu zpracování a porovnávání získaných výsledků. Závěr obsahuje celkové hodnocení práce s doporučením možných budoucích vylepšení.

Abstract

Mario Kamburov, Optimal datamining methods for processing semi-structured medical data

Diploma thesis, Pilsen 2015

The aim of my thesis was to propose solution and to create a program that would allow correction of medical texts on the basis of a very large and diverse semistructured data from medical reports. The work describes the theoretical possibilities of natural language processing, and several already implemented datamining algorithms for text classification. There is described the principle of my proposed solution, which uses a database to store the training data. The implementation of Java program is also described in detail using MySQL, PostgreSQL, and IBM DB2 databases. The verification was applied to a selected collection of medical data. At the end there are comprehensive statistics of the data processing and comparing the obtained results. Conclusion contains an overall assessment of the work with recommendations for possible future improvements.

Obsah

1 Úvod

V dnešní době existuje řada informačních systémů a je potřeba dbát na celkovou architekturu, integrovatelnost a rozšiřitelnost či nadstavby existujících systémů. Z toho vyplývá, že je důležité přemýšlet globálně při návrhu a implementaci jakéhokoliv softwarového produktu či komponentu. V rámci této diplomové práci jsem se zabýval návrhem řešení pro korekci lékařských textů.

Problematika lékařských informačních systémů je velmi rozšířená a komplexní. V tomto projektu jsem se podílel na práci s výzkumným týmem MRE KIV[1]. Úkolem bylo automaticky korigovat nebo opravovat medicínské odborné termíny a lékařské texty v českém jazyce do takového tvaru, ve kterém by jim po prvním přečtení porozuměli nejen lékaři specialisté, ale i například ambulantní lékaři, akreditovaní Českou Lékařskou Komorou. Tento projekt slouží jako podklad k integraci pro aktuální systém běžící ve FN Plzeň ve formě komponentu pro korekci lékařských textů.

S pokrokem Internetu a moderných technologií narůstají počet dokumentů v elektronické podobě a potřeby dolování znalostí z nich. Metody, zabývající se klasifikací dokumentů, souhrnně nazýváme metody pro dolování či dobývání znalosti z dat. Metody kategorizace jsou založeny na principech pravděpodobnosti, umělé inteligence či rozhodovacích stromech atd. Tyto metody využívají hlavně pozitivní vzorky.

Výsledkem této práce je aplikace pro automatickou korekci medicínských semistrukturovaných dat. Na základě podrobného porovnávání jednotlivých algoritmů umělé inteligence bylo vybráno několik vhodných a optimálních metod pro rychlou a maximálně bezchybnou korekci medicínských textů. V závěru této práce je vidět porovnávání a zhodnocení dosažených výsledků.

[1]MRE KIV - Medical Research and Education Information Systems:
`http://mre.kiv.zcu.cz` součást Katedry Informatiky a Výpočetní techniky v Plzni

2 Analýza problému

Zadání vzniklo na popud výzkumné skupiny MRE KIV a Fakultní Nemocnice Plzeň. Při pomalém a manuálním opravování lékařských textů docházelo ke stagnaci efektivity výzkumných aktivit. To bylo hlavním důvodem tohoto projektu a naprogramovaného řešení, kde se kladl hlavní důraz na plnou automatizaci při korekci lékařských zpráv. Důvodem automatického dataminingu je využití následných výsledků tohoto řešení pro další zpracování a text-mining analyzu medicínských textů ze strany výzkumného týmu. Jinými slovy tento projekt slouží jako podpůrný systém pro korekce medicínských textů.

Hlavním problémem byl oprava zkrácených lékařských textů a medicínských slov v lékařských zprávách do neuniverzálních a někdy nepochopitelných tvarů. Jednalo se o agregace lékařských zpráv z více zdrojů (nemocnic a lékařských stanic), kde bylo potřeba, na základě těchto zpráv pracovat s informacemi o pacientovi dál v rámci výzkumu. Jelikož se jedná o odborné texty, zabývající se lidském zdravím je jistota a přesnost výsledků kritická. Toto je hlavní důvod a důležitost konzervativní korekce těchto medicínských textů. Pod pojmem konzervativní lze rozumět, že v tomto projektu není velký prostor pro inovace a experimentální odzkoušení ruzných metodik dataminingu, protože se klade důraz na přesnost a úplnost výsledků.

Obecným problémem této textové analýzy jsou neexistující standardy v latině a obzvlášť v českém jazyce, které by lékaři mohli používat a řídit se jimi jako oficiální psaní zkratek daných lékařských termínů. Existuje slovník medicínských abreviatur, ale samotné zkratky v českém jazyce mohou nabývat mnohdy zcela jiný význam, v závislosti na kontextu, ve kterém jsou použité. Proto je potřeba tyto zkratky zkorigovat či rozepsat do původního tvaru.

Cílem této práce byl textová analýza lékařských pojmů v lékařských zprávách, nikoli samotná oprava textu jako součást oboru strojové učení pro práci s přirozenou řečí. Jinými slovy v tomto projektu se neklade důraz na lemmatizaci[2]. či češtinu samotnou, ale pouze na korekci medicínských textů a zpráv ve tvaru takovém, ve kterém by porozuměl každý jiný lékař, pracující na území ČR, akreditován Českou lékařskou komorou. Následně po provedení analýzy byly data predzpracované a odfiltrované. Přípravou dat zabralo zhruba 50% času v tomto projektu. Posléze byly tyto data připravené pro použití do ostrého odskoušení v běžící aplikaci nad vybraných algoritmech.

Nejdůležitější část diplomové práci je zpracování nestukturovaných a semistrukturovaných dat. Dále při samotné analýze bylo zjištěno, že poskytnutý dataset je rozmanitý a obsahuje velké množství různorodých zkratek jak z pohledu lexikální semantiky českého jazyka, tak z pohledu latinského jazyka pro medicínské termíny. Nicméně kombinace českých a latinských zkratek komplikuje celkový smysl a korekce textů. Za tímto účelem jsem se zaměřil přimárně na opravu medicínských zkratek ve spolupráci s odborníkem, který z pohledu strojového učení měl roli supervizora (učitel). Ten se staral o porozumění zpráv v daném kontextu.

Jako podstatným krokem a výstupem analýzy bylo to, že je důležité nezačínat velkém řešením hned ze začátku, ale postupně inkrementálním způsobem přidávat další funkcionality a vylepšování trénovacích lékařských dat, nad kterými následně testovat algoritmy. To se osvědčilo hned na začátku tohoto projektu při předzpracováním semistrukturovaných medicínských dat a implementace proudu dat z uživatelského rozhraní do samotného algoritmu.

[2]Lemmatizace je proces, kdy je slovo převedeno do základního tvaru - tzv. lemma. Například slovo "počítačích" je převedeno na slovo "počítač". Umožňuje lepšímu strojovému porozumění textu a používá se pro vyhledávání fulltextem

3 Současný stav

V rámci tohoto projektu byla poskytnutá anonymizovaná množina reálných lékařských zpráv, anamnéz a medicínských textů, které byly využity jako podklad k realizaci tohoto projektu. Tato kapitola uvádí jednotlivé případy typů zkratek, které jsem měl za úkol opravit či rozepsat do základního tvaru.

3.1 Popis dat

Vstupní data byla dodána Plzeňskou fakultní nemocnicí, konkrétně rentgenovým oddělením. Jednalo se o anonymizovaná data. Celkem 375 vzorků. Šlo o vzorky zpětně dohledatelné podle URI a čistě textové .csv soubory. Aby se jednalo o kvalitní porovnání různých metod dataminingu byly použity stejné vzorky na všech algoritmech. [6]

3.2 Klasické zkratky ukončené tečkou

Lékařské termíny potřebující korekci mohou být v různém tvaru. Jde primárně o klasické zkratky ukončené tečkou, například:

1. "Na mozku je patrná hyperdenzita v počátečním úseku a. cerebri media vlevo"
   Kde zkratka "a." znamená arteria. V rozepsaném tvaru:
   "Na mozku je patrná hyperdenzita v počátečním úseku arteria cerebri media vlevo"

2. "Alterace perfúzních parametrů v povodí ACM dx. s pouze drobným jádrem v bílé hmotě." ⇒ "Alterace perfúzních parametrů v povodí ACM dextra s pouze drobným jádrem v bílé hmotě."

Tyto zkratky bylo potřeba ručně dodefinovat ve všech možných podobách, které můžou nabývat. Například zkratka a. by mohla být použita také jako ar., art., arter., artr., Aa., A. apod. Pro efektivnější natrenování modelu byly zkratky tohoto typu definovány a sepisovány ručně s použitím stejného kontextu, za pomocí kterého lze dedukovat též stejný význam. Tím jsem se snažil dosáhnout úplnost modelu slov.

3.3 Abreviatúry

Jedná se též o abreviatury, jako například:

"CTAG:

Odstupy krčních tepen z oblouku aorty jsou volné, v oblasti jugula jsou patrny dislokační změny při zvětšené ŠŽ a uzlovité strumě vycházející z dol. pólu levého laloku, která zasahuje mírně retrosternálně. Oboustranně jsou patrny poměrně masivní kalcifikace v plátech v oblasti větvení ACC, není však patrna významnější stenoza. Intrakraniálně typické uspořádání řečiště s embolem v M1 úseku pravostranné ACM."

Kde:

- CTAG znamená počítačová tomografická angiografie,
- ŠŽ je štítná žláza,
- ACC je arteria carotis communis,
- M1 je pars sfenoidalis,
- ACM je arteria cerebri media

3.4 Složitější zkratky

Další komplikovanější případy jsou neobvyklé zkratky, kombinace velkých a malých písmen, několik teček mezi písmeny nebo v nejhorším případě, když autor lékařské zprávy zapomene přidat tečku ke zkratce, pak lze poměrně složitě naučit algoritmus na danou zkratku. Příklady:

1. "Aa. vertebrales volné. ⇒ Arteria vertebrales volné"

2. "vyš. provedeno po apl. KL i. v. dvoufázově
   ⇒ Vyšetření provedeno po aplikaci kontrastních látek intravenozně dvoufázově"

3. "CT mozku nativně: Vyjádřená ischemie levého F, T a P laloku, bez zn. krvácení. Diskr. tlak. změny na F roh levé postr. komory, střed. struktury bez lateralizace. Prosáknutí měkkých poktývek hlavy vpravo TP a v obl. pravé tváře.
   postkontrastně
   CT perfuze:
   Výpadek perfuze s minimální penumbrou FTP vlevo, zachován pruhovitý okrsek perfuze okolo centrálního sulcu vlevo."

Na příkladu 3 vidíme ukázka medicínské zkratky stejného typu ve 2 neuniverzálních podob. Jednak lze vyjádřit ischemie laloku rozděleně pomocí popisujících písmen F - Frontální, T - Temporální, P - Parietální nebo je to možné vyjádřit přimo zkratkou FTP. Tento příklad je komplikovaný v tom, že ne vždy lze konkrétně klasifikovat danou zkratku v závislosti na jejím použití. Lékař by mohl někdy potřebovat popsat v lékařské zprávě jednu z nich vícekrát a tímto se narušuje možnost naučit algoritmus na přesnou korekci. Nicméně při nalezení sloučené a zkrácené verze FTP klasifikační algoritmus zvládá rozhodování úspěšně.

3.5 Další typy zkratek

Další typy zkratek obsahují například číslíce:

1. "Uzávěr ACM dx v úrovni A1/M1. Aplazie P1 sin - plní se cestou zadní komunikanty.bez dalších patol. změn Willisova okruhu." ⇒ "Uzávěr arteria cerebri media dextra v úrovni A1/M1. Aplazie P1 sin - plní se cestou zadní komunikanty. Bez dalších patologických změn Willisova okruhu."

2. "MR pánve a horních stehen: nativně a postkontrastně, 3T, sekvence T2 TSE, T2 TIRM, T1 TSE FS, a postkontrastně T1 TSE +FS"

   Tyto zkratky jsou specifické pro rentgenologické oddělení a vyžadují porozumění odborníkem z rentgenologického oddělení, který je se zkratky týkající se specifických zákroku při provádění rentgenové vyšetření seznámený a zkušený.

3.6 Problémy medicínských dat

Na základě příkladech dat v předchozích kapitolách lze odvodit, že problematika medicínských dat je poměrně komplexní a nejednoznačná. Obecně je možné data rozdělit do dvou dílčích skupin podproblému - heterogenita dat a právní problémy dat

3.6.1 Heterogenita

Základním problémem medicínských dat je nejednotný formát a složení dat při zpracování lékařských zpráv a jejích ukládání do ruznch struktur - relačních či NoSQL databází. Dostupné data byly dodány v .csv formátu po anonymizaci, nicméně ukládání a transformace skutečných neanonymizovaných dat může mít různou formu a podobu. To je hlavním problémem při jakékoliv další zpracování

dat takovéhoto typu. Jako další vlastnost lze zdůraznit problém BigData[3], neboli flexibilně narůstající objem dat. Známé vlastnosti jsou tzv. 4V:

- volume (objem) Objem dat narůstá exponenciálně.
- velocity (rychlost) Objevují se úlohy vyžadující okamžité zpracování velkého objemu průběžně vznikajících dat. Vhodným příkladem může být zpracování dat produkovaných kamerou.
- variety (různorodost, variabilnost) Kromě obvyklých strukturovaných dat jde o úlohy pro zpracování nestrukturovaných textů, ale i různých typů multimediálních dat.
- veracity (věrohodnost) Nejistá věrohodnost dat v důsledku jejich nekonzistence, neúplnosti, nejasnosti a podobně. Vhodným příkladem mohou být údaje čerpané z komunikace na sociálních sítích.

3.6.2 Etnické nebo právní problémy

Z pohledu vlastnictví lze říct, že je důležité mít na vědomí, že medicínské data se týkají osobních informací jednotlivých pacientů. Jinými slovy se bavíme o privatní a důverihodná data registrovaných pacientů. Jedná-li se o privátní data, pak je třeba, s ukládáním a transformaci či transport dat, zacházet velice opatrně a řídit se specifickými zákony a pravidel, stanovené v právních vnitrostátních a mezinárodních pramenech práv, jako například zákony a normy týkající se ochraně osobných údajů pacientů.

[3]firma Gartner za big data označuje soubory dat, jejichž velikost je mimo schopnosti zachycovat, spravovat a zpracovávat data běžně používanými softwarovými prostředky v rozumném čase.

4 Zpracování a příprava dat

4.1 Semistrukturovaná data

Semistrukturovaná data jsou definována jako data, která jsou neuspořádaná či neúplná, jejich struktura se může měnit, dokonce nepredikovatelným způsobem.

Semi-strukturovaná data je forma strukturovaných údajů, které nejsou v souladu s formální strukturou datových modelů spojených s relační databází nebo jiných forem datových tabulek, ale přesto obsahují značky nebo jiné oddělující sémantické prvky a utváří hierarchii v rámci dat. Z tohoto důvodu, jsou také známé jako samopopisující se struktury dat. [1] [4]

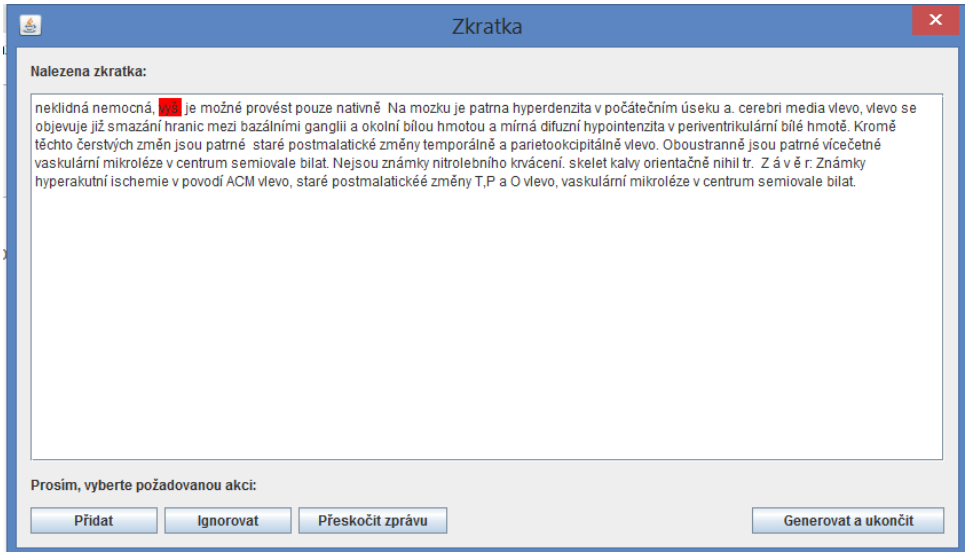
V semi-strukturovaných datech, subjekty patřící do stejné třídy můžou mít různé atributy, i když jsou seskupeny, dále pořadí atributů není důležité.

Semi-strukturovaná data stále častěji narůstají. S příchodem internetu full-textové dokumenty a databáze už nejsou jedinou formou údajů. Různé aplikace potřebují zdroj pro výměnu informací. [1]

- Výhody: [1] [4]
  - Není třeba se starat o objektově-relační nesoulad entit, místo toho se mohou serializovat objekty pomocí lehkých knihoven.
  - Podpora vnořených nebo hierarchických dat.
  - Podpora seznamů objektů.
- Nevýhody: [1] [4]
  - Tradiční relační datový model má oproti semistrukturovaným datům populární jazyk dotazu SQL.
  - Nestabilita dat bez integritních omezení.

4.2 Příprava dat

Ke zpracování textu zpráv byl použit externí program pro přípravu kontextů k jednotlivým zkratkám, tak aby algoritmy byly natrénované v potřebném množství a kvalitě dat. Program pro zpracování kontextů a předpřípravu dat byl určený lékaři, který lékařské zprávy opravoval a program automaticky opravené zprávy rovnou ukládál do trénovacího datasetu. Tyto datasety byly následně použité pro trenování jednotlivých zkratek při celkové opravě LZ. Při zpracování textů a práce s programem pomohl můj bratr MUDr. Boyko Kamburov.



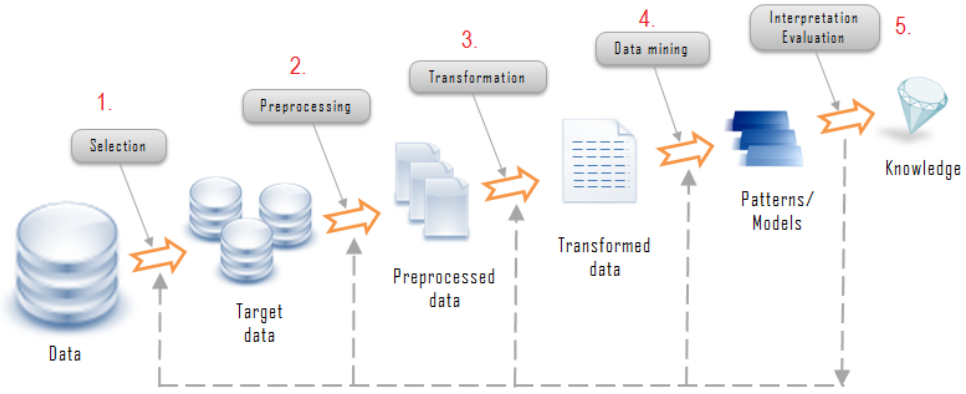
Obrázek 1: Ukázka rozhraní programu pro předzpracování dat

Program prochází jednotlivé zkratky a lékař má možnost je programově rozepsat a automaticky uložit do požadovaného tvaru pro správné natrenování vůči kontextu dané zkratky. Program byl vyvinut jiným studentem FAV KIV (Bc. Petrem Žákem), spolupracující nad výzkumem MRE jako podklad k této práci. Tento program vyřešil automatizaci a rychlejší zpracování, ale ne úplnost rozepsaných zkratek. Za tímto účelem jsem po osobních schůzkách s lékařem vždy doplňoval trénovací data a kontroloval jejich správnost ručně.

4.3 Manuální příprava

K připravení trénovacích množin do velké míry napomohla spolupráce reálného lékaře. Před samotným použití datamining algoritmů, bylo potřeba porozumět problematice medicínských textů a celkově konkrétním lékařským zprávám. Za tímto účelem MUDr. Boyko Kamburov měl za úkol také lékařské texty manuálně projít a přečít. Celková časová náročnost vyšla na 3-4 týdny průběžné práci, kdy pečlivě a postupně byly lékařské zprávy analýzované a opravované ručně.

V důsledku bylo zjištěno, že v závislosti na specializaci konkrétního lékaře, který lékařskou zprávu napsal, pak lze tvrdit, že každý jiný specialista v tomto oboru by měl této zprávě jednoznačně porozumět též. Tím lze odvodit to, že aplikace mé diplomové práce má sloužit jako nápomocný nástroj pro rozepisování lékařských zpráv do podoby, do které by jim mohl porozumět i lékař, který není specialista v daném oboru, ve kterém jsou lékařské zprávy. Klasicky se může jednat o medicínské texty v semistrukturovaném formátu z rentgenologického oddělení, radiologického oddělení či jiného zdravotnického pracoviště.



Obrázek 2: Ukázka procesu předzpracování dat za účely dataminingu

Tato část diplomové práce slouží jako záchytný bod pro další zpracování jaké-

koliv automatizovaného programu. Bylo zjištěno, že ne každý specialista umí porozumět konkrétním lékařským textům se specifickým zaměření. Tím je úkol o to těžší a závislý na trénovacím korpusu dat.

Na obrázku 2 je vidět celkový proces dobývání znalostí z dostupných dat v rámci diplomové práci. V prvním kroku (Selection) se data z různých zdrojů vybíralá do tzv. target data, neboli smyslúplná a vhodná data pro účely medicínských textů. Jednalo se o tzv. čištění dat. V dalším kroku č.2 (Preprocessing) byly tyto data předzpracovaná a transformovaná do standardizované podoby. V rámci tohoto projektu se zde nacházel proces ruční a manuální korekce lékařských zkratek, které byly rozepisováné ve dvou krocích. První za pomocí automatizovaného programu popsaný v předchozí kapitole a druhý proces, zabývající se ruční korekce reálným lékařem. V kroku č.3 (Transformation) byly připravené data převedené znovu do elektronické podoby a snadno transformované do vhodného datového modelu. Zde jsem použil datový sklad jako úložiště dat pro účely otestování celkového konceptu. V kroku č.4 (Datamining) byly aplikované algoritmy data-miningu nad předzpracovanými daty a byly vytvořeny tzv. vzory (patterns), kterými se algoritmus řídil při svém rozhodování. V kroku č.5 (Evaluation) se následovně algoritmus rozhodoval sám a vyhodnocoval celkové výsledky medicínských dat. Na základě těchto modelů bylo umožňěno samotné dobývání znalosti z dat. Jinými slovy výstupem byly opravené lékařské zprávy. Tyto medicínské texty byly v iteračním cyklu postupně opravovány a procházely kroky č. 4 a č.5 inkrementálně na základě již natrénovaného korpusu dat v datovém skladu, použitý pro účely tohoto projektu.

Nejnáročnější část projektu byla fáze č. 1 a č.2, týkající se samotném předzpracování a čištění dat. Posléze dalším důležitým krokem bylo zvolit vhodný optimální algoritmus, který by fungoval stabilně nad dostupnými daty.

5 Úloha kategorizace textů

Pro účely této diplomové práce jsem analyzoval a zabýval se problematikou kategorizace textů do konkrétních tříd. Úloha automatické korekce medicínských semistrukturovaných dat jsem determinoval na klasifikační úlohu.

Kategorizace (klasikace) textu je úloha, která medicínské zkratce (dokumentu) přiřazuje jednu nebo více z předem daných tématických kategorií(tříd). Slovem třídu lze rozumět jako rozepsanou lékařskou zkratku. Modul vykonávající klasifikaci se nazývá klasifikátor. Klasifikaci tvoří dvě fáze- trénovací a klasifikační. V průběhu trénovací fáze (učení) klasifikátor analyzuje množinu lékařských zkratek (dokumentů), u kterých známe jejich zařazení do kategorií, a získává z ní určité informace, například vztah všech slov, obsažených v textu ke kategoriím (rozepsaným zkratkám). Zařazení těchto dokumentů do kategorií provádí expert. V případě tohoto projektu medicínské texty byly algoritmem trénováné reálným lékařem MUDr. Boykem Kamburovem. Klasifikátor je na základě získaných informací učitelem(lékařem) schopen vykonat v klasifikační fázi vlastní klasifikaci, samostatně a uměle rozhodovat. Známe-li, do kterých kategorií klasifikované dokumenty patří, můžeme spočítat hodnocení klasifikátoru. [5]

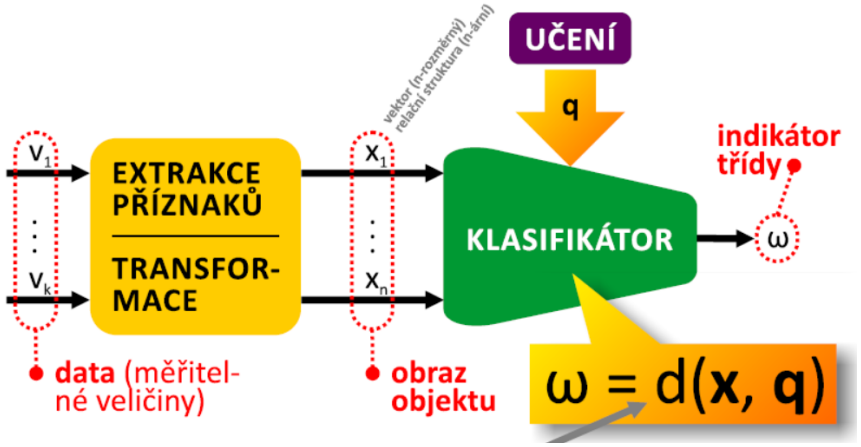
Klasifikace medicínských textů lze rozdělit do několik kroků:

1. Lékař manuálně zařazoval určitý počet dokumentů (trénovací data) do stanovených kategorií (rozepsaných zkratek)

2. Provádělo se automatické natrénování klasifikátoru podle vybraného algoritmu. Klasicky se jednalo například o výpočet četnosti slov pro konkrétní kategorii i výpočtu pravděpodobností

3. Po vytvoření modelu bylo možné kvalitu modelu zhodnotit na testovacích datech

Způsob, jakým klasifikátor vytváří trénovací model, ukládá a používá informace, je závislý na konkrétní dataminingové metodě kategorizace. Pro účely této práce jsem použil několik metod. Kapitola 7 uvádí použité metody klasifikace medicínského textu.



Obrázek 3: Ukázka tvorby klasifikátoru na základě extrakce příznaků z trénovacích dat. Funkce d(x, q) je tzv. rozhodovací pravidlo, které za pomoci nastavení q rozhoduje samostatně do které třídy zařadí vzorek. [2]

5.1 Lexikální analýza medicínských textů

Lexikální analýza je první nutný, nikoli postačující krok při indexaci [4]. Stará se o nalezení hranic mezi slovy ve vstupních datech. Klíčovým problémem lexikální analýzy je stanovení oddělovačů slov. V případě lékařských textů největším problémem je určení zkratek ke korekci. Velmi často se stává, že zkratky jsou umístěny na konci věty a jsou ukončeny tečkou.[7]

[4]Indexace je proces vyjádření obsahu dokumentu pomocí prvků selekčního jazyka, obvykle s cílem umožnit zpětné vyhledávání

Klasicky se jako oddělovače slov volí interpunkční znaménka, mezery a znaky konce řádku. Mezi oddělovače je dále možné započítat další nepísmenné znaky. U některých symbolů však nastává dilema, zda je považovat za oddělovače slov či nikoli, případně za jakých okolností tak máme činit. U medicínských zkratek obvlášť. Interpunkční znaménka způsobují problémy většinou v medicínských zprávách, kde se vyskytují jako součást jmen (např. MR pánve a horních stehen: nativně a postkontrastně, 3T, sekvence T2 TSE, T2 TIRM v úrovní A1/M1, T1 TSE FS, a postkontrastně T1 TSE +FS). Navíc slova se spojovníky vyvolávají další otázky - má-li být slovo se spojovníkem považováno za slovo jediné nebo rozděleno na slova samostatná, či zda spojovník neoznačuje jen dělení slov na konci řádků.

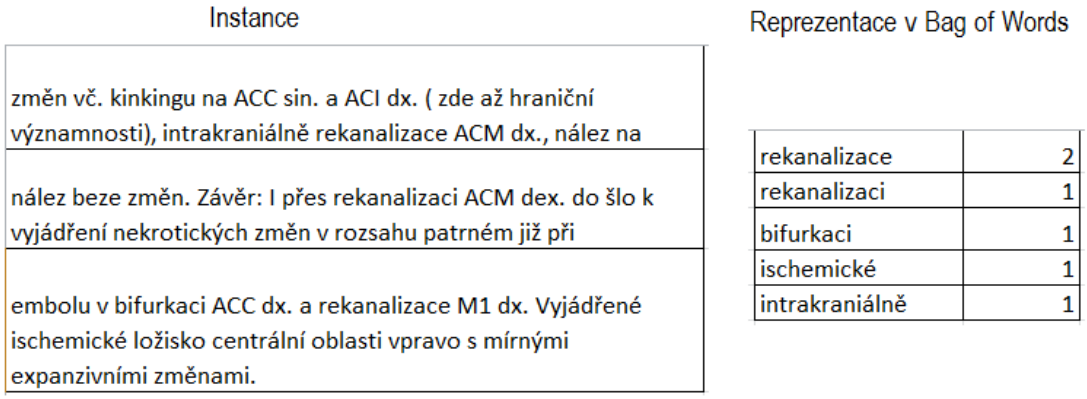
Zejména v medicínských textech se objevují problémy s intepretací číslic – ve většině případů sice netvoří samostatná slova, mohou se však vyskytovat jako součást identifikujících slov, které mohou mít velký vliv na konečný výsledek klasifikace . Obvyklým řešením bývá respektovat pouze ta slova s číslicemi, která číslicí nezačínají. Dalším problémem může být výskyt spojovníků – obvykle jej chápeme jako oddělovač různých slov, přestože někdy může být součástí názvů , nebo jen indikuje dělení slova na konci řádku. [8] [7]

Dalším problémem může být rozlišování velkých a malých písmen. V praxi řešení tohoto problému spočívá v převodu všech písmen na stejnou velikost. To ovšem nepředstavuje problém s anglickým textem, nicméně v případě češtiny je potřeba věnovat pozornost způsobu kódování. [7]

Největším problémem lekařských textů a medicínských dat byly nekorektně zapsaná slova a zkratky, které nebyly ukončeny interpuknctí. Takové zkratky lze velmi snadno splést se spojovníkem, čárkou nebo přinejhorším úplně ignorovat při predzpracováním.

5.2 Nevýznamné slova

V procesu indexace bývá vhodné odstranit ze vstupních dat slova, jež mají zpravidla pouze gramatický význam a nenesou žádnou informační hodnotu pro účely klasifikace. Seznam takových slov se označuje jako slovník nevýznamových slov (tzv. stop-list nebo stop words [5]). Slovník nevýznamových slov můžeme vytvářet ručně nebo použitím frekvenčního slovníku, jenž obsahuje určité procento nejčastěji se vyskytujících slov (tj. slova, objevující se ve většině zpracovávaných dokumentů). Výhodná strategie spočívá ve spojení obou způsobů – vyjdeme z automaticky vytvořeného frekvenčního slovníku, ze kterého některá slova odebereme a zefektivníme rozhodování klasifikátoru lékařských zpráv. Ignorování nevýznamových slov nejen urychluje zpracování a snižuje paměťové nároky. Celkem tato slova tvoří zhruba 20% textu a výsledky jsou přesnější. [7]



Obrázek 4: Příklad tvorby bag of words po filtraci nepotřebných slov v modelu obsahující trénovací instance lékařských zpráv z rentgenologii FN Plzeň

[5]Jako stopslova se při počítačovém zpracování přirozeného jazyka označují slova, která se v daném jazyce vyskytují často, ale nenesou žádnou významovou informaci, mají zpravidla pouze syntaktický význam. Typicky se jedná o spojky, předložky atd. Jsou též označovany jako negativní slovník.

6 Datamining

6.1 Teorie

Datamining (Získávání znalostí z databází nebo KDD - Knowledge Discovery in Databases) [9], interdisciplinární podoblast počítačové vědy, [10], je výpočetní proces objevování vzorů ve velkých datových sadách, zahrnující metody na pomezí umělé inteligence, strojového učení, statistiky a databázových systémů.[10] Celkovým cílem procesu dolování dat je získat informace z datových sadách a transformovat je do srozumitelné struktury pro další použití. Zahrnuje také aspekty databázi a správu dat, predzpracování dat, model úvahy, zhodnocení metrik, složitost úvahy, vizualizace atd. Používají se techniky jako rozhodovací stromy, asociační pravidla, regresní, logistická analýza, neuronové sítě či shluková analýza (clustering) pro segmentaci skupin podle společných vlastnosti.

Existuje obecný postup kroků všech datamining metodologií:

1. Inicializační – formulace úlohy a porozumění problému. Často automatické vyhledávání znalostí nelze provádět zcela naslepo.

2. Datový – vyhledání a příprava dat pro analýzu. Statistické algoritmy potřebují data připravená v určité podobě, proto není možné použít přímo surových semistrukturovaných dat z operačních databází.

3. Analytický – hledání informace v datech, vytváření statistických modelů. Nejčastěji používanými metodami však jsou logistická regrese s automatickým výběrem proměnných, rozhodovací stromy a neuronové sítě.

4. Aplikační – zjištěné poznatky a modely je třeba uvést do praxe, například korekce lékařských zkratek.

5. Řízený – je třeba zajistit zpětnou vazbu (jak efektivní byl model) a v případě

dlouhodobě nasazovaných modelů i kontrolovat, zda model příliš nezestárl a zachovává si svoji efektivitu.

6.2 Datamining v medicíně

Jak jsem již zmíňoval v kapitole 3.6, největším problémem je samotné předzpracování dat, zahrnující filtrace, transformace a čištění dat od nestrukturované a semistrukturované podoby do jasně určené podoby, vhodné pro trénování datamining algoritmů. Posléze lze analyzovat dosažené výsledky a hledat vhodné korelace a úvahy pro zhodnocení. Záznamy pacientů se skládájí z klinických, laboratorních parametrů, výsledků jednotlivých vyšetření, které jsou specifické pro různá odvětví a specializace. Tyto data mají většinou následující vlastnosti:

- Neúplnost: Chybí hodnoty atributů, chybí některé atributy zájmu nebo obsahují pouze souhrnná data
- Šum: Obsahují chyby nebo odlehlé hodnoty
- Nekonzistentní: Obsahují rozpory v kódech nebo názvech
- Temporální: Parametry chronických onemocnění v čase

Neexistuji-li kvalitní údaje, lze tvrdit,že neexistuji nýbrž kvalitní výsledky. Datový sklad pro dolování medicínských dat potřebuje důslednou integraci kvalitních údajů. Řešením je vytvoření rozsáhlého slovníku pojmů, jednotného rozhraní pro integrace více datových zdrojů a předávání elektronických záznamu o pacientech na úrovní mezi jednotlivými nemocnicemi celosvětově. Je dále potřeba porozumění tzv. Medical Domain, neboli v IT je nedostatek lidi se znalostní domény v oboru medicíny.

7 Výběr datamining algoritmů

Tato sekce popisuje výběr jednotlivých algoritmů API WEKA pro klasifikaci medicínského textu. Algoritmy byly použité z vývojařské knihovny tohoto dataminingového nástroje. Bylo potřeba se s jejich implementací velmi podrobně seznámit.

7.1 Naivní bayes

7.1.1 Teoretické základy

Samotný algoritmus, který jsem vybral ve své práci se zakládá na klasické bayesové větě, která je založená na pravděpodobnostním vzorečku, kterým se po celou dobu algoritmus řídí a rozhoduje, v závislosti na pravděpodobnosti výskytu daných slov, zda zařadí do konkrétní kategorii danou lékařskou zkratku (resp. testovací vzorek). Pravděpodobnostní vzoreček vypadá takto:

$$P(k|doc) = \frac{P(doc|k)P(k)}{P(doc)} \tag{1}$$

Kde $k \in K$ je rozepsaná lékařská zkratka z množiny všech možných kategorií do které lze zařadit danou nalezenou neopravenou zkratku a doc je samotná zkratka, kterou potřebujeme klasifikovat. Pravděpodobnost hypotézy $k \in K$, podmíněna pozorováním medicínské zkratky doc lze tedy vyjádřit jako poměr pravděpodobností, že lékařská zkratka doc patří do dané kategorie k (rozepsaná zkratka), krát apriorní pravděpodobnost kategorie k, vůči evidenci, což je apriorní pravděpodobnost trénovacích dat (rozepsaných zkratek). Jinými slovy algoritmus je naivní ve svém přístupu, tím že spoléha na to, že v závislosti na hodnotě pravděpodobnostního výskytu v trénovacím modelu bude zkratka v testovací množině patřit do konkrétní kategorie k (konkrétní rozepsaná zkratka). Jinak řečeno, algoritmus spoléhá na to, že existuje rovnoměrná distribuce.[11] [12]

7.1.2 Naivní Bayes a klasifikace textů

Algoritmus Naivní Bayes je velmi rozšířený mezi klasifikačními algoritmy pro práci s textem. Je to jeden z nejpoužívanějších a nejefektivnějších algoritmů strojového učení pro práci s textem. Praxe ukazuje, že algoritmus pracuje skvěle jak s malým tak i s velkým množstvím trénovacích dat. Toto bylo osvědčeno v rámci této diplomové práci. Důležité však je kvalitní natrénování modelu. Ohodnocení pak bude prokazovat mnohem méně chyb.

7.1.3 Naivní Bayes Multinomial

Vylepšení původního algoritmu Naivní Bayes, kterého jsem použil ve své práci zejména z důvodu, že jsem potřeboval jemnější výsledky, je algoritmus Multinomiální Naivní Bayes. Ten se liší oproti původnímu pouze v tom, že používá Multinomcké rozdělení. Klasický Naivní Bayes používá rovnoměrné rozdělení.[11] [12] [13]

7.1.4 Detailní příklad

Podívejme se na detaily, jakým způsobem Multinomiální Bayes klasifikuje své vzorky do odpovídajících tříd. Za prvé je potřeba nadefinovat apriorní pravděpodobnost dané třídy[12]:

$$P(k) = \frac{Nc}{N} \tag{2}$$

Kde Nc je počet vzorků trénovacího modelu, popisujících třídu k a N je počet všech vzorků trénovací množiny. Vezmeme-li trénovací dataset pro rozepsanou medicínskou zkratku "dextra":

```
@relation 'dx'

@attribute text string
@attribute class {dx, dex, dextra, ?}

@data
'změn vč. kinkingu na ACC sin. a ACI dx. ( zde až hraniční významnosti), intrakraniálně rekanalizace ACM dx., nález na',dextra
'nález beze změn. Závěr: I přes rekanalizaci ACM dex. do šlo k vyjádření nekrotických změn v rozsahu patrném již při',dextra
'embolu v bifurkaci ACC dx. a rekanalizace M1 dx. Vyjádřené ischemické ložisko centrální oblasti vpravo s mírnými expanzivními změnami.',dextra
```

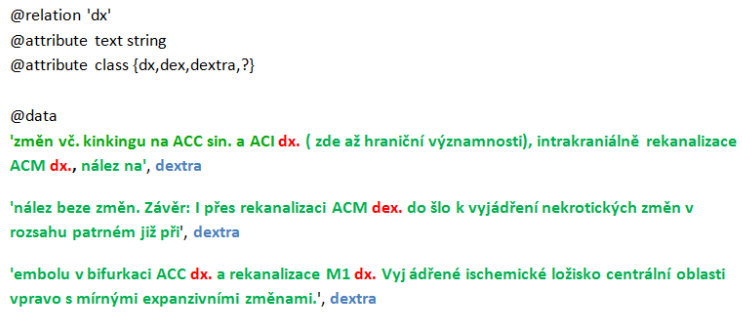
Obrázek 5: Ukázka trénovacích dat ve formátu .arff

pak naše apriorní znalost P(k) pro rozepsanou zkratku dextra je $P('dextra') = \frac{3}{3} = 1$. Model je minimálně natrenovaný různými zkratkami stejného typu. S ohledem na to, že v trénovacím modelu mám třídu '?', ke které nepatří žádný trénovací vzorek, tak tyto pravděpodobnosti algoritmus přepočítává a přiřazuje tzv. m-odhad třídy '?'. Proto v celkovém výsledku je tato výsledná pravděpodobnost o něco málo menší. Poté se vypočítávají jednotlivé podmíněné pravděpodobnosti. Pro každé slovo trénovací množiny se vypočte podmíněná pravděpodobnost s jakou může patřit do dané třídy. Existuje na to následující vzoreček [12] :

$$P(doc|k) = \frac{count(doc, k) + 1}{count(k) + |V|} \tag{3}$$

Kde $P(doc|k)$ udává pravděpodobnost dat, za podmínky, že patří do třídy k. Počítá se to snadno a to tak, že count(doc, k) vyjadřuje četnost slov testovacího vzorku, obsažené v trénovací množině. Z důvodu normalizace se přičítá jednička. Ve jmenovateli count(k) je počet všech slov, týkajících se naší konkrétní třídy k (rozepsaná lékařská zkratka dextra). $|V|$ je tzv. vocabulary neboli slovník všech slov trénovací množiny (známo také jako bag of words - ukázka na straně 22). Ve výsledku, například podle obrázku 6 na straně 28, je podmíněna pravděpodobnost lékařské zkratky 'dx' patřící do třídy 'dextra' následující:

$$P('dx'|dextra) = \frac{count(doc, k) + 1}{count(k) + |V|} = \frac{4 + 1}{57 + 57} = \frac{5}{114} = 0,04385$$



Obrázek 6: Ukázka výpočtu četnosti trénovacího modelu

Počet všech slov třídy dextra je celkem 57. Počet zkratek 'dx' jsou přesně 4. Počet všech slov trénovací množiny (bag of words) je též 57, protože tato množina je specifická pro tento projekt a neobsahuje další třídy.

7.1.5 Optimální vylepšení algoritmu

Z důvodu použití knihovny WEKA, bylo potřeba seznámit se podrobněji s implementací používaného algoritmu tohoto open-source produktu. Byly zjištěny malé změny v algoritmu, oproti klasickému učebnicovému vzorečku. Weka používá normalizaci za pomocí logaritmování a odlogaritmování jednotlivých pravděpodobností. Domnívám se, že důvod této implementace je rychlost ve zpracování výsledků. Normalizace vypadá takto:

$$P(k|doc) = \frac{WP(k)}{P(doc)} \tag{4}$$

Kde W = $e^{(log(x)-log(y))}$. Argument x je tedy $P(doc|k)$, což už víme, že je celková podmíněná pravděpodobnost daného testovacího vzorku, patřící do konkrétní třídy k a y je vždy maximální hodnotou ze všech vypočtených x. V podstatě hodnota W se vždy rovná 1 v nejlepším případě, kdy je největší pravděpodobnost, že zkratka patří do kategorie k. Pravda je taková, že z normalizace vyplývá, že e^0=1. Jinými slovy, je-li hodnota W $<$ 1, tak bude v každém případě menší pravděpodobnost,

že patří do této kategorie. Je-li W = 1 je nejlepší pravděpodobnost, že patří do dané kategorie. Ovšem k jiným výsledkům je možno se dopracovat v závislosti na apriorních znalostí P(k), což může být způsobeno specifickým natrénováním dat. Například v trénovací množině bude více rozepsaných zkratek v kategorii arteria cerebri media pro zkratku "a.", tím algoritmus bude spíše směřovat numericky k třídě arteria cerebri media v případech kdy bude váhat kterou třídu vybrat nebo jsou-li si pravděpodobnosti velmi blízké.

7.2 SMO

7.2.1 Vznik

SMO (zkratka ze Sequential Minimal Optimization) je algoritmus pro řešení problému kvadratického programování (QP), který vzniká při trénování algoritmu SVM (Support Vector Machines). Byl vynalezen Johnem Plattem v roce 1998 ve společnosti Microsoft Research. SMO je široce používán pro trénování SVM a je implementován populární knihovnou LIBSVM. Zveřejnění algoritmu SMO v roce 1998 vyvolal hodně vzrušení v komunitě SVM vývojařů, protože dříve dostupné metody pro trénování SVM byly mnohem složitější a výpočetně náročnější. [14]

7.2.2 Optimalizační problém SVM

Uvažujme podle binární klasifikace problému s datovymi sady (x_1, y_1),...,(x_n, y_n), kde x je vstupní vektor a $y_i \in$ (-1, 1) je binární název odpovídající k němu. Jemné rozpětí SVM je natrénováno k řešení problému kvadratického programování, kde je problém vyjádřen ve tvaru:

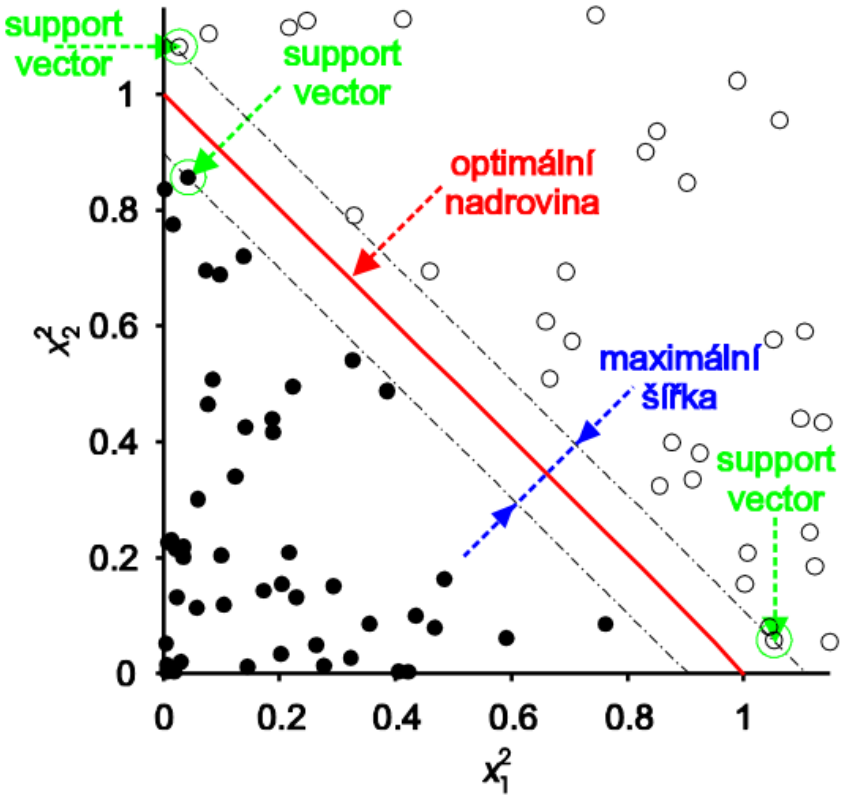
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j, \tag{5}$$

kde platí:

$$0 \leq \alpha_i \leq C, \quad \text{pro } i = 1, 2, \ldots, n,$$

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

kde C je SVM hyperparameter a K(x_i, x_j) je funkce jádra, oba dodávané uživatelem. Proměnné α_i jsou Lagrangeovy multiplikátory. [15] [16] [18]



Obrázek 7: Rozhodovací hranice (nadrovina) a stanovení podpůrných vektorů [17]

Optimální lineární oddělovač se v algoritmu support vector machines hledá pomocí metody kvadratického programování. Zde je jistá obdoba s hledáním maxima jako u lineárního programování, problém je však složitější. Velkou výhodou

je skutečnost, že těchto podpůrných vektorů je obvykle mnohem méně než datových bodů, takže efektivní počet parametrů definujících optimální oddělovač je pak mnohem menší než N.

7.2.3 Algoritmus SMO

SMO je iterativní algoritmus pro řešení problému optimalizace popsaný výše. SMO rozděluje tento problém do série nejmenších možných dílčích problémů, které jsou pak řešitelné analyticky. Vzhledem k lineárnímu omezení rovnosti, která zahrnuje Lagrangeové multiplikátory α_i, nejjednodušší možný problém se týká dvou takovýchto multiplikátorů. Poté, pro libovolné dva multiplikátory α_1 a α_2, pak platí:

$$0 \leq \alpha_1, \alpha_2 \leq C,$$

$$y_1\alpha_1 + y_2\alpha_2 = K$$

a takto zredukovaný problém lze vyřešit analyticky. Je potřeba najít minimum jednorozměrné kvadratické funkce. K je negativní součet rovnice, který v každé iteraci klesá.

Algoritmus probíhá následujícím způsobem [18]:

1. Nalezne Lagrangeovy multiplikátory α_1, které porušují Karush Kuhn Tuckerovo, KKT[6] podmínky pro optimalizační úlohy.

2. Vybere si druhý násobitel α_2 a optimalizuje dvojici (α_1, α_2).

[6]KKT podmínky jsou nutné podmínky pro hledání optimálního řešení úlohy nelineárního programování, za předpokladu, že i některé další podmínky jsou splněny. Je to zobecnění metody Lagrangeových multiplikátorů na omezující podmínky neobsahující rovnost (může tedy obsahovat nerovnosti)

3. Opakuje kroky 1 a 2, dokud nedokonverguje.

4. Když všechny násobky Lagrange splňují podmínky KKT (v rámci tolerance uživatelem definované), problém je vyřešen. Ačkoli tento algoritmus zaručeně vždy dokonverguje, se používají heuristiky pro výběr páru multiplikátorů tak, aby se urychlil postup celého algoritmu.

V nejhorším případě dosahuje asymptotyckou složitost $O(n^3)$.

7.3 J48

J48 je open source Java implementace C4.5 algoritmu, generující rozhodovací strom. C4.5 staví rozhodovací stromy z trénovacích dat stejným způsobem jako algoritmus ID3 [7], pomocí metody informační entropie. Je vylepšený o tzv. "pruning" (prořezávání stromu) a optimalizovaný proti přeučení (over-fitting). Trénovací data jsou vyjádřena množinou $S = s_1, s_2, ...$ již klasifikovaných vzorků. Každý vzorek s_i se skládá z n-rozměrného vektoru $(x_{1,i}, x_{2,i}, ..., x_{n,i})$, kde x představují atributy nebo vlastnosti vzorku, jakož i jako třídu, v níž s_i spadá. [19]

V každém uzlu stromu, C4.5 vybírá atribut, který nejúčinněji rozděluje trénovací sadu vzorků do podskupin posilujících jednu nebo durhou třídu. Kritériem rozdělení (prořezávání stromu) je normalizovaná informace zisku (rozdíl entropie). Atribut s nejvyšším normalizovaným informačním ziskem je vybrán do role rozhodujícího. C4.5 algoritmus se pak opakuje na poduzlech.

7.3.1 Rozdělení C4.5

Tento algoritmus má několik základních případů. [21]

[7] ID3 - Iterative Dichotomiser 3 je algoritmus generující rozhodovací strom, vynalezeny Rossem Quinlanem

1. Všechny vzorky v seznamu patří do stejné třídy. Když toto nastane, algoritmus vytváří listový uzel, který při rozhodování klasifikuje texty vždy do stejné třídy.

2. Žádný z atributů nepřináší žádný informační zisk. V tomto případě, C4.5 vytváří rozhodovací uzel abstraktně výš od kořene a používá očekávanou hodnotu třídy.

3. Nalezne třídu se kterou se nesetkal. Opět platí, že C4.5 vytváří rozhodovací uzel výše stromu pomocí očekávané hodnoty.

7.3.2 Příklad fungování algoritmu

V pseudokódu, obecný algoritmus pro vytváření rozhodovacích stromů funguje následovně: [19] [20]

1. Kontroluje pro základní případy

2. Pro každý atribut a

   (a) Vypočte jednotlivé informační zisky

   (b) Vyhledá normalizovaný podíl získané informace z prořezání stromu v a

3. Nechť a_{best} atribut je nejlepší normalizovaný informační zisk

4. Vytvoří rozhodovací uzel, který rozděluje v a_{best}

5. Opakuje na poduzlech získaných rozdělením v a_{best}, posléze přidává tyto uzly jako potomky uzlu

Ukázka vygenerovaného stromu pro rozhodování zda pacientka má rakovinu prsou, na základě několik atributů, které nejúčinněji rozdělují trénovací sadu na podskupiny lze vidět v příloze A, na obrázku 22. Jedná se o atributy velikost uzlu, velikost nádoru či poloha uzlu (nahoře, dole atd.). [26]

7.4 IBk

Algoritmus IBk implementuje metodu k-nejblížších sousedů. Ve strojovém učení, algoritmus k-nejbližších sousedů (nebo k-NN v krátkosti, zkrácené z k-Nearest Neighbours) spadá mezi neparametrické[8] metody klasifikace. [23]

Při k-NN klasifikaci, výstupem je příslušná třída. Vstupní vzorek je klasifikován na základě hlasování svých sousedů. Testovací vzorek je přiřazován k příslušné třídě, jejíž vzorky jsou nejběžnější mezi k- nejbližších sousedů ($k \in N_+$).
Pokud k = 1, pak je vzorek přiřazen třídě jediného nejbližšího souseda.

K-NN je typ učení, založené na instancích[9], nebo též lazy metoda(líná), kde funkce je aproximována pouze lokálně, a všechny výpočty jsou odloženy až do samotné klasifikace. K-NN algoritmus patří mezi nejjednodušší ze všech algoritmů strojového učení. [23] [22] [24]

Nedostatkem algoritmu k-NN je to, že je citlivý na lokální strukturu dat. Algoritmus nemá nic společného s algoritmem k-means, který je další populární metoda strojového učení.

7.4.1 Princip

Existuje několik možností výběru nejbližších sousedů. Základní metriky aplikované v algoritmu kNN jsou popsané níže v tabulce 1. [24]

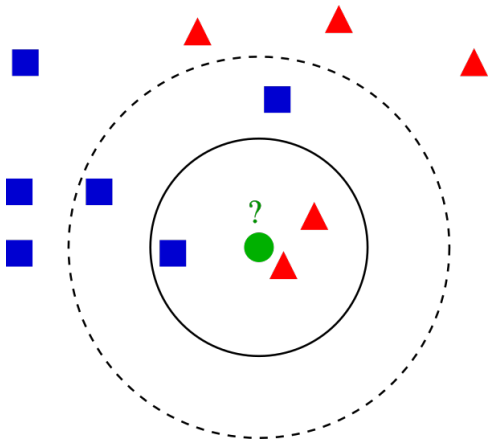
[8]Neparametrické metody klasifikace - tyto metody jsou založeny na podstatně slabších předpokladech než metody parametrické, neboť u nich nepředpokládáme znalost tvaru pravděpodobnostních charakteristik tříd

[9]Učení založené na instancích - buduje hypotézy přímo z trénovacích instancí. Jinými slovy, složitost hypotézy může růst exponenciálně s přibývajícími daty, v nejhorším případě, hypotéza je seznam n trénovacích vzorků. Výpočetní složitost klasifikace jedné nové instance je O(n). Jednou z výhod, které tato metoda má oproti jiným metodám strojového učení je její schopnost přizpůsobit svůj model na dosud nespatřená data.

Pro účely korekce lékařských zpráv jsem si postačil se základním nastavením tohoto algoritmu. Používal jsem euklidovskou vzdálenost mezi jednotlivými sousedy. Důvodem je malé množství trénovacích dat, tím i celková asymptotická složitost algoritmu nepřesahovala O(n).

Metrika	**Matematické vyjádření**
Euklidovská vzdálenost	$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$
Hammingova (Manhattan) vzdálenost	$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \lvert a_r(x_i) - a_r(x_j) \rvert}$
prekrytí (overlap)	$d(x_i, x_j) = \sum_{r=1}^{n} (1 - \delta(a_r(x_i), a_r(x_j)))$
kosínova metrika	$d(x_i, x_j) = \frac{\sum_{r=1}^{n} (a_r(x_i), a_r(x_j))}{\sqrt{\sum_{r=1}^{n} (a_r(x_j), a_r(x_j)) . \sum_{r=1}^{n} (a_r(x_i), a_r(x_i))}}$

Tabulka 1: Metriky pro nalezení k nejbližších sousedů

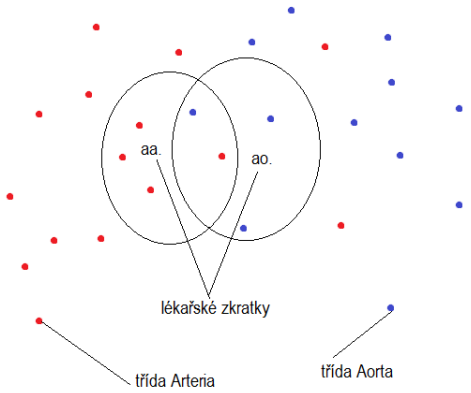


Obrázek 8: Příklad k-NN klasifikace. Zkušební vzorek (zelený kruh), by měly být klasifikovány buď do první třídy modrými čtvercemi nebo do druhé třídy červených trojúhelníků. Je-li k = 3 (plná čára kruhu) je přiřazen k druhé třídy, protože jsou k dispozici 2 trojúhelníky a pouze 1 čtverec uvnitř vnitřního kruhu. Je-li k = 5 (přerušovaná čára kruhu) je přiřazena první třídy (3 čtverce vs. 2 trojúhelníky uvnitř vnějšího kruhu).[24]

7.4.2 Příklady použití

S narůstajícím množstvím trénovacích dat, narůstá též chybovost nebo nepřesnost tohoto algoritmu. Nejlepší volba k záleží na datech. Obecně platí, že vyšší hodnoty k snižují rozptyl při klasifikaci [wiki], ale způsobují méně zřetelné hranice mezi třídami. Vhodně velké k může být zvolené různými heuristickými technikami. Přesnost k-NN algoritmu může být vážně snížena přítomností hlučných nebo irelevantních příznaků (klasicky nepotřebná slova, spojky apod.).[23] [22]

- **1-NN** - Zjistíme vzdálenosti všech prvků trénovací množiny od neznámého prvku. Vybereme daný prvek trénovací množiny, který je nejblíže a neznámý prvek klasifikujeme do stejné třídy.
- **3-NN** - Kolem neznámého prvku vytvoříme hyperkouli, která obsahuje právě tři nejbližší prvky trénovací množiny. Neznámý prvek klasifikujeme do té třídy, která je v hyperkouli zastoupena největším počtem prvků.
- **k-NN** - Při použití metod k-NN pro $k > 1$ je velmi důležitá volba k. Pro dvě třídy volíme k vždy liché (kvůli jednoznačnosti rozhodování) pro více tříd mohou nastat situace, kdy nelze jednoznačně rozhodnout.



Obrázek 9: Příklad k-NN klasifikace lékařských zkratek

8 Implementace řešení

8.1 Volba vývojového prostředí

Pro splnění účelu práce bylo nutné nejprve najít vhodný nástroj a to takový, který bude splňovat dvě hlavní kritéria. První z nich byla implementace v jazyce Java, dalším pak, aby byl nástroj open source. Na internetu existuje řada nástrojů a knihoven pro vývoj aplikací umělé inteligence. Důvod použití tohoto nástroje jsou jeho obsáhlost, pokročilost a optimalizovanost. Podle řady průzkumů[10] na internetu je jeden z nejlepších, momentálně, open-source dataminingového produktu. Dále se mi při programování s touto knihovnou dobře pracovalo a za půl roku jsem se naučil poměrně dobře s ní pracovat.

8.2 WEKA API

WEKA (zkratka z Waikato Environment for Knowledge Analysis) je prostředí pro analýzu znalostí. Obsahuje balík programů strojového učení napsaný v Javě, vyvinutý na University of Waikato, Nový Zéland. Weka je svobodný software dostupný pod GNU licencí. Tyto dva předpoklady naplňují cíle této práce a to byl důvod, abych si vybral tuto knihovnu jako primární zdroj algoritmů. Knihovna je open-source a to dodává velkou svobodu při práci s ní i její případné nadstavby či editaci algoritmů. [25]

- Obsahuje nástroje pro předzpracování dat, třídění, spojení pravidel, regresi a vizualizaci.
- Poskytuje možnost využití svých algoritmů voláním z jiné aplikace.

[10] `http://www.predictiveanalyticstoday.com/top-15-free-data-mining-software/`, `http://www.junauza.com/2010/11/free-data-mining-software.html`

Software Weka je dodáván ve dvou standardních edicích, jsou to:

- Book
- Developer

8.2.1 Format vstupních dat

Podporovaný formát, ve kterém jsou zpracovávaná data je **.arff**. Na štěstí framework nabízí řadu metod jak manipulovat s .arff soubory nebo konvertovat data z jiných formátů (jako např.: .csv, .json, .txt apod.) do formátu .arff.

Soubor definuje 2 hlavní části, se kterými pracuje. První část obsahuje hlavičku, ve které definuje název relace a atributy, a druhá část je tělo ve které se nachází samotná data. V mém případě atributy jsou 2 typy - text a class (atribut text typu string je textová hodnota, což je samotný text daného článku a atribut class je její třída, nebo-li kategorie ke které patří z n možných). Takto strukturovaná data jsou zpracovávaná vnitřně, za pomoci speciálních metod frameworku. Ukázka formátu trénovacích dat po transformaci do formátu arff je možné vidět na obrázku 5 na straně 27. Kód psaný ve formátu ARFF je case-insensitive, nerozlišuje mezi velikostí písmen (např. příkazy @relation a @RELATION jsou stejné). Dále mezery mezi klíčovými slovy a mezi jednotlivými hodnotami jsou nevýznamné. [25]

Popis použitých atributů

- **String**

  Atributy String umožňují vytváření atributů, které obsahují libovolné textové hodnoty. To je velmi užitečné v text-mining[11] aplikací. Je vnitřně reprezentován jako číselná hodnota(vektor), proto je potřeba použít filtr pro manipulaci řetězce (např.: StringToWordVectorFilter). Atributy String jsou

[11] Text-mining je vědecká disciplína na pomezí dolování z dat, strojového učení a počítačové lingvistiky. Vyvíjí se především s potřebou automatického zpracování ohromného množství informací dostupných v podobě volného textu.

deklarovány takto:

ATTRIBUTE zkratka string

- **Nominální atributy**

  Nominální hodnoty jsou definované jako jmenné specifikace seznamu možných hodnot.

  $< nominal - name1 >, < nominal - name2 >, < nominal - name3 >, ...$

  Atribut typu nominal pak může nabývat pouze jedné z uvedených hodnot. Příkladem může být atribut dx obsahující tři nominální hodnoty reprezentující medicínskou zkratku:

  @attribute class dx,dex,dextra

  Atribut tímto definuje možné klasifikační třídy, které lékařská zkratka může nabývat. Případně to můžou být klasifikační třídy, které mají význam výstupní predikce.

8.2.2 Datová část

Datová část formátu ARFF slouží k definování jednotlivých hodnot atributů deklarovaných v hlavičce, jinými slovy k ukládání konkrétních dat pro jednotlivé hlavičkové informace. Datová část se deklaruje @data na novou řádku. Chybějící hodnoty zapisujeme pomocí znaku ?. Každá řádka reprezentuje jednotlivou instanci trénovacích dat. Která datová hodnota patří ke konkrétnímu atributu se pozná podle pořadí datových hodnot. Záleží tedy na pořadí hodnot a také na dodržení jejich počtu, který musí být shodný s počtem atributů v hlavičce. Pokud máme 3 atributy, pak každá instance musí obsahovat tři hodnoty. Za poslední hodnotou v instanci se již čárka nezapisuje.[25]

8.3 Rozvržení aplikace

Tato podkapitola popisuje jednotivé balíky a způsob navržení aplikace. Implementace je abstraktní a znovupoužitelná. Jinými slovy při programování jsem přemýšlel globálně a navrhoval aplikace, tak aby zpracovávala použité algoritmy stejným způsobem a nebylo třeba měnit nic, kromě výběru samotného algoritmu. Metody pro trénování a klasifikace dat jsou izolované, robustní a znovupoužitelné.

8.4 Uložení trénovacích dat

Trénovací data byla ukládána do databáze pro jednodušší práce a lepší přenos v případu migrace na jiný systém. V prvním konceptu této diplomové práce byla data ukládána a testována v datovém skladu ve formě SaaS [12] služby v cloudu od IBM Bluemix [13]. Služba používaná pro testování toho projektu nabízela IBM BLU Acceleration in-memory [14] ukládání dat. To přinášelo naprosto skvělou rychlost ve zpracování výsledků. Relační model byl jednoduchý a snadno-přenesitelný do jiných relačních databází. Stejně tak aplikace byla navržena, tím způsobem, aby stačilo přepsat JDBC [15] připojení k databázi a mohla se používat i s jinou databázi.

8.4.1 Struktura databáze

Na nasledujících obrázcích je vidět navržená struktura navržené schéma databáze a ukázka vzorových trénovacích dat v databázi, používané při opravování zkratek v aplikaci.

[12] SaaS - Software as a Service
[13] IBM Bluemix - Platform as a Service Cloud Provider IBM pro vývoj webových aplikací v cloudu
[14] in-memory - sloupcové ukládání dat, namísto klasického řádkové
[15] JDBC - Java Database Connectivity

ID [INTEGER]	ZKRATKA [VARCHAR(50 OCTETS)]	TRAINING_SENTENCE [VARCHAR(200 OCTETS)]	VYZNAM [VARCHAR(100 OCTETS)]
1	a.	k vyš. z 9.4.2010. Rozsáhlá malacie v celém povodí a. cerebri med. sin. , ...	arteria
2	a.	IC hemorhagie. Perfúze mozková. Nekrotické okrsky v povodí a. cerebri ...	arteria
3	arter.	mm (min. 22 mm) vůči okolnímu parenchymu hyperdenzní v arter. fázi (...	arteriální
4	arter.	i ostatní ložiska, která jsou dobře diferencovatelná jen v arter. fázi. Jinak...	arteriální
5	bazal.	ložisko laterálně od front. rohu pravé postr. komory v bazál. gangliích v...	bazálních
6	bilat.	šíře do 20 mm, vlevo v.s. stopa tekutiny. Bilat. dor sobasálně pruhovité n...	bilaterálně
7	bilat.	ACM dx., nález na bazilárním povodí vč. ACP bilat. s e nemění. CT plic a...	bilaterálně
8	dg.	mm o nízkých denzitách do 20 HU v dif. dg. nekrotická meta či cysta. Po...	diagnóza
9	dif.	15 mm o nízkých denzitách do 20 HU v dif. dg. nekrotická meta či cysta....	diferenciální
10	dif.	3 cm, dif. dg kontrakce. Kličky v levém epig. ne jsou dostatečně naplněn...	diferenciální
11	dx.	změn vč. kinkingu na ACC sin. a ACI dx. (zde až hraniční významnosti), i...	dextra
12	dx.	nález beze změn. Závěr: I přes rekanalizaci ACM dx. do šlo k vyjádření n...	dextra
13	dx.	embolu v bifurkaci ACC dx. a rekanalizace M1 dx. Vyj ádřené ischemick...	dextra
14	hl.	patol. vzhledu do vel. 24 mm. Širší plicnice (hl. větv e šíře kolem 28 mm...	hlavní
15	i.v.	nemění. CT plic a mediastína s k.l. i.v. Tu e xpanze vpravo centrálně nad ...	intravenozně
16	i.v.	tepen a mozku: Provedeno po podání kontrastní látky i.v. I ntrakraniálně...	intravenozně
17	k.l.	bilat. se nemění. CT plic a mediastína s k.l. i.v. Tu expanze vpravo centrá...	kontrastní látky
18	k.l.	cirkulár. zesílené (6,5mm), bez nápadnějšího sycení k.l.. Term. úsek ilea v...	kontrastních látek
19	later.	velikosti. Neměnná nekróza po RFA v pravém laloku jater. Je dna z výše ...	laterální
20	lož.	V S6 v těsné blízkosti porty hypodenzní ostře ohraničené lož. vel. 15 x 1...	ložisko
21	lymf.	dutinách. Na krku vlevo laterálně od štítné žl. lymf. uzli na vel. 15 a 18 m...	lymfatická
22	lymf.	a v pravém plic. hilu zmnožené a zvětšené lymf. uzli ny patol. vzhledu d...	lymfatické

Obrázek 10: Ukázka ukládání trénovacích dat medicínských textů pro korekce lékařských zkratek

ID [INTEGER]	ACCRONYM [VARCHAR(50 OCTETS)]	TRAINING_SENTENCE [VARCHAR(200 OCTETS)]	MEANING [VARCHAR(100 OCTETS)]
1	CTAG	CT AG: vyš. po bolu k.l.. Po zklidnění nemocné ...	počítačová tomografická angiogr...
2	AG	CT AG katotid a intrakraniálního řečiště:	angiografie
3	CT	CT mozku bez k.l. i.v. Krvácení neprokázáno, m...	počítačová tomografická
4	CT	CT AG vyš. po bolu k.l. Po zklidnění nemocné s...	počítačová tomografická
5	CT	CT perfúze provedeno po podání kontrastu i.v....	počítačová tomografická
8	ACM	Krátký uzávěr či těsná stenoza na 2 periferníc...	arteria cerebri media
9	KL	Provedeno po aplikaci KL intravenozně ve dvo...	kontrastní látky
6	ACC	V bifurkaci ACC bilat jen drobné kalcifikace, b...	ACC
10	ACE	ACETaké bifurkace obou ACC a odstupy ACI a ...	arteria carotis externa

Obrázek 11: Ukázka ukládání trénovacích dat medicínských textů pro korekce lékařských abreviatur

8.4.2 Práce s databází

Celkový koncept byl navržen tak, aby podporoval nejpoužívanější databáze na trhu. Uvažoval jsem nad znovupoužitelností kódu a migrací dat z různých databází při programování. Zpracování dataminingových metod a volání dat z databází bylo testováno nad IBM DB2 s optimalizací pro sloupcové vyheldávání, MySQL 5.6.20, PostgreSQL 9.3. Dále jsem v přiloženém zdrojovém kódu připravil konfigurační soubory pro případ použití projektu s MSSQL Server, SQLITE3, Oracle, HSQL databázemi. Rychlost zpracování výsledků se liší vybranou technologií a nastavením transakčního výkonu, zpracovávájícího dotazy k databázi. Lze tvrdit, že jako nejnáročnější operace je časté dotazování na databázi při nalezení lékařské

zkratky ke korekci. Ukázka jednoduchého dotazu pro naučení klasifikátorů:

```
SELECT * FROM VYZNAM WHERE ZKRATKA = 'a.';
```

Dále v přiloženém CD lze nalézt DDL[16] skripty pro trénovací data, včetně samotných trénovacích dat, připravené pro reálné použití a nasazení do ostrého provozu.

```
InstanceQuery query = new InstanceQuery();
File props = new File(vcapServices.getPropsPath());
query.setCustomPropsFile(props);
query.setDatabaseURL(vcapServices.getUrl());
query.setUsername(vcapServices.getUser());
query.setPassword(vcapServices.getPassword());
String trainingQuery =
"SELECT * FROM VYZNAM WHERE ZKRATKA = '"+zkratka+"'";
```

Objekt **InstanceQuery** převádí výsledky SQL dotazu rovnou na instance dat, kterými následně natáhne do paměti a umožňuje snadnější práci.

```
query.connectToDatabase();
Instances train;
// Retrieve the query from the DataWarehouse
query.setQuery(trainingQuery);
train = query.retrieveInstances();
train.setClassIndex(train.numAttributes() - 1);
query.disconnectFromDatabase();
if(train.size() == 0){
        train = null;
}
```

[16]DDL - Data Definition Langugage

Další sekvence kódu znázorňuje posílání předpřipraveného SQL dotazu a pomocí metody **query.retrieveInstances();** uskutečněný dotaz vrací data přímo do tvaru Instance.

8.5 Balík cz.zcu.fav.kiv.mre.controllers

Balík obsahuje logickou část pro zpracování požadavků klienta na opravu textů. Stará se o obsluhu požadavků a volání pomocných metod pro trénování klasifikátoru, ten volá další pomocné metody a následně serveru vrací opravený text ve formě http response odpovědi klientovi. Za účelem regularizace trénovacích dat a omezení v přeučením trénovacích vzorků jsem používal regulární výrazy, které odchytávaly postačující kontext kolem vyhledávaných zkratek. Na základě praktických zkušeností jsem používal kontext v rozmezí 5 slov před a po nalezené zkratce. Jednalo-li se o více jak 10 slov se prvek přeučil a nevykazovalo to dobré výsledky, naopak nastavil-li jsem přílíš malý kontext nalezené zkratky, výsledky znovu nebyly tak dobré. Tím jsem se dopracoval k optimálnějšímu nastavení kontextu, potřebný pro natrénování dané zkratky, které model následně by používal a rozhodoval sám.

8.5.1 Regulární výrazy

String regex = `"((?:[^\\s]+\\s+)(0,5)([\\w!@#$&ěščřžýáíé*]+ \\.[\\w!@#$&ěščřžýáíé*]+\\.|[\\w!@#$&ěščřžýáíé*]+\\.)(?:\\s+[^\\s]+)(0,5))"`
Pattern pattern = Pattern.compile(regex);

Tento zdánlivě na první pohled dost komplikovaný regulární výraz hraje klíčovou roli při hledání zkratek pro zpracování textu na webu. Výraz prohledává všechna slova ukončená tečkou a rovnou získává kontext kolem nich. Tím ošetřuji rovnou při zpracování dat nadbytečné informace a klasifikuji danou zkratku vůči svému lokálnímu kontextu. Příklad na odkaze `https://regex101.com/r/aQ3zJ3/16` a

také v příloze A na obrázku 20 na straně 70.

8.5.2 Pseudokód hledání zkratek v textu

```
Pattern pattern =
Pattern.compile("(slova_pred) zkratka (slova_po))");
Matcher matchDot = pattern.matcher(textToCorrect);
while (matchDot.find()) {
  String tmp = matchDot.group(1);
  //train model for particular acronym
  boolean wellTrained =
  trainModel(ac, vcapServices, matchDot.group(2));
  if(wellTrained){
   //classify acronym
  String correctedText = correctMedicalAcronym(ac, classify, tmp);
  String correctedContext =
  matchDot.group(1).replace(matchDot.group(2), correctedText);
  correctedMedicalReport =
  correctedMedicalReport.replace(tmp, correctedContext);
        }
}
```

První while cyklus pro kažkou nalezenou zkratku natrénuje model s daty z databáze a ihned zklasifikuje(opraví) zkratku v rámci svého kontextu. Při opravě medicínských semistrukturovaných dat a lékařských textů je použit lokální kontext zkratek. Tím se zamezuje rovnou nadbytečné natrénování. Míra úspěšnosti klasifikace záleží do velké míry na trénovacích datech. V rámci tohoto projektu na poskytnutých datech funguje naprosto vynikajícím způsobem při zvolených algoritmech.

```
Pattern patternACRR =
Pattern.compile("(slova_pred) abreviatura (slova_po))");
Matcher matchAccr =
patternACRR.matcher(correctedMedicalReport);
while (matchAccr.find()) {
  String tmp = matchAccr.group(1);
  //train model for particular abbreviature
  boolean wellTrained = trainModel(ac, vcapServ, matchAccr.group(2));
  if(wellTrained){
   //classify the particular abbreviature
   String correctedText = correctMedicalAcronym(ac, classify, tmp);
   String correctedContext =
   matchAccr.group(1).replace(matchAccr.group(2), correctedText);
   correctedMedicalReport =
   correctedMedicalReport.replace(tmp, correctedContext);
        }
}
```

Druhý while cyklus pro kažkou nalezenou abreviaturu funguje stejným způsobem - natrénuje model s daty z databáze a ihned zklasifikuje(opraví) lékařskou abreviatúru v rámci svého kontextu. Rozdělení do dvou cyklů je z důvodu přehlednosti, jelikož regulární výraz by mohl být příliš komplikovaný a mohlo by dojít k chybám a nenalezeným zkratkám. Tím lze na čisto vyhledat pouze abreviatúry a ty následně opravit. Cyklus je druhý v pořadí, jelikož obecně abreviatůr je méně v textu a jejich oprava je tímto rychlá.

8.6 Balík cz.zcu.fav.kiv.mre.datamining

Balík datamining obsahuje třídy pro zpracování vstupních dat a práci s frameworkem WEKA. Musel jsem se naučit podrobně pracovat s objekty a metodami nabízené touto knihovnou. Balík též obsahuje třídy pro předzpracování dat a jejích vyhodnocení.

V tomto balíku jsou též k nalezení metody pro převod slov na vektory **StringToWordVector** nebo filtrování dat trénovací množiny a nastavení maximálného počtu slov jako restrikci proti přeučení. Příklad použití algoritmu StringToWordVector ve spojení s medicinskými daty lze nalézt v příloze A, obrázek 21:

Je zde také implementovaná důležitá metoda jako **removeUnknownWords()** pro vytvoření tzv. **bag of words** [17], popsaná detailněji v kapitole 5.2 na straně 22 a **extractTrainingDataAccronyms()** pro získání dat z databáze a pomocí nich následné nakrmení klasifikátoru.

Zde se též nachází dvě nejdůležitější metody, použité ve smyčce opravující jednotlivé zkratky popsané v kapitole 8.5.2 na straně 44:

- **buildClassifier()** - vytváří klasifikátor pro danou zkratku na základě dat v databázi. Jestliže neexistují vhodná data je zkratka ignorována
- **classify()** - klasifikuje do správně třídy danou zkratku, jinými slovy ji rozepisuje či opravuje. V případě, že není dostatek trénovacích dat, zkratku ponechává.

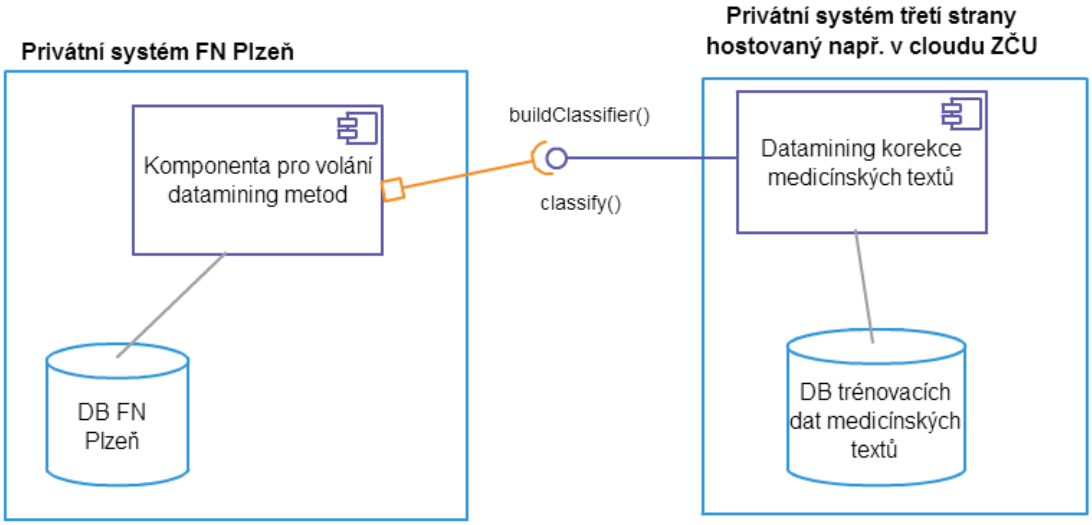
[17] BoW - model slov, udržující četnost všech důležitých slov trénovacího modelu

8.7 Balík cz.zcu.fav.kiv.mre.utils

Balík utils slouží jako provozní obsloužení metod ostatních balíků. Má primární účel jako pomocný balík a hlavně se využívá pro napojení k databázi a získání connection object pro úspěšné připojení k danému datovému zdroji obsahující trénovací data a rozepsané lékařské zkratky.

8.8 Modely nasazení

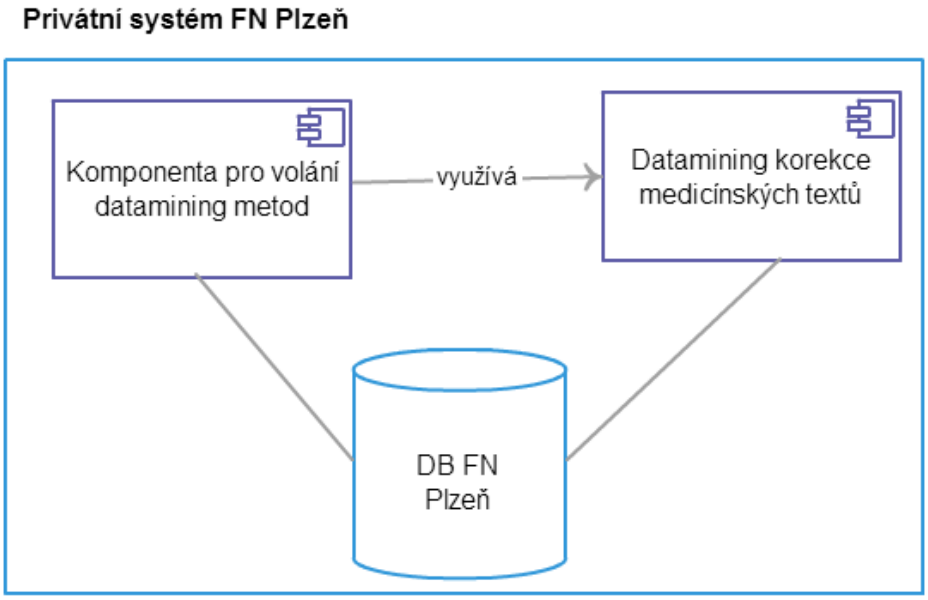
Ve své práci jsem uvažoval nad dvěmi hlavními scenáři použití naprogramovaného programu pro korekci textů realtime ve webovém prostředí. Jednak jsem navrhnul nejlepší možný způsob integrace řešení mé diplomové práce do stávajícího systému Fakultní nemocnice v Plzni. Tato podkapitola popisuje jejích možné nasazení, pomocí diagramu komponent.



Obrázek 12: První scenář nasazení systému pro korekce zkratek

Na prvním obrázku je vidět nasazení pomocí REST API přes firewall systému FN Plzeň. Stávající systém nemocnice by se tím vyhnul spravování podsystému

pro korekci zkratek. Řešení vyžaduje opatření pro vytvoření bezpečnostního komunikačního kanálu, případně zmapování endpointů při generování REST API, přes autentizační token a nastavení práv uživatelů, přistupujících k vyvěšenému API z bepečnostních důvodů.



Obrázek 13: Druhý scenář nasazení systému pro korekce zkratek

V druhém scenáři jsem uvažoval, že balík tříd a dataminingových metod by mohl být nasazen přímo v stávajícím systému. Tím by se mohla využít společná databáze běžící pod firewallem FN Plzeň. Řešení ušetří náklady, nicméně by mohlo zpomalit celý vývojový proces z důvodů byrokratických a legislativních norem, potřebných pro nasazení jakéhokoliv IT řešení ve státní nemocnici. Stejně tak údržba by vyžadovala další oprávnění a povolení přístupu, jejichž získání by mohlo stagnovat celý projekt.

Druhé řešení lze považovat za nejvhodnější z hlediska bezbečnosti dat a jejích ukládání uvnitř stávající IT infrastruktury nemocnici a úvahy ohledně osobních údajů pacientů

9 Porovnávání dataminingových metod

Výsledky simulace klasifikace medicínských semistrukturovaných dat či lékařských zpráv jsou rozděleny do několika dílčích položek, pro snadnější analýzu a hodnocení. V první části, správně i nesprávně klasifikované instance[18] budou rozdělené na číselné a procentuální hodnoty. Následně se zaměřím na statistický Cohen's Kappa koeficient, střední absolutní chyba a standardní kvadratická chyba, které budou také pouze číselné hodnoty. Dále zhodnotím relativní absolutní chyby a kořenový relativní čtvercové chyby v procentech. Výsledky simulace jsou uvedeny v tabulkách 3 a 4 níže. Tabulka 3 shrnuje především výsledky založené na přesnosti a času potřebné pro každou simulaci. Mezitím, tabulka 4 ukazuje výsledek na základě chyby během simulace. Obrázky 14 a 15 jsou grafická znázornění výsledky simulace. [27] [28]

9.1 Hodnotící kritéria

K ohodnocení kvality natrénovaných modelů jsem použil matriky nabízené používaného frameworku pro analýzu klasifikátorů. Při porovnávání byly vypočítávány následující hodnoty[29]:

- **Průměrná absolutní chyba (Mean Absolute Error)** - Ve statistice, tato hodnota (MAE) je veličina používaná k měření, jak blízko predikce nebo předpovědi jsou k případným skutečným výsledkům. Počítá se následovně:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i|\,. \tag{6}$$

  Z názvu lze vydedukovat, že střední absolutní chyba je průměr absolutních chyb $|e_i| = |p_i - a_i|$, kde p_i je predikce a a_i skutečná hodnota. Průměrná absolutní odchylka je obyčejně mírou chybné předpovědi pro analýzu časových

[18] Instance - vstupní data

řad [19], kde pojem "průměrná absolutní chyba" je někdy používán k záměně s více standardními definicemi střední absolutní odchylky.

- **Střední kvadratická odchylka - (Root mean square error, tzv. RMSE)** je často používaná míra rozdílů mezi hodnotami předpovídaného modelu a hodnotami skutečně pozorovanými. RMSE představuje ukázku směrodatné odchylky rozdílu mezi předpokládanými hodnotami a pozorovanými hodnotami. Tento rozdíl se nazývá rezidua. Počítá se matematickým vzorcem:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2} = \sqrt{MSE} \tag{7}$$

  Kde MSE je Mean Squared Error (Střední kvadratická chyba), p_i je predikce a a_i je skutečná aktuální hodnota. Hodnoty veličiny RMSE pro každý klasifikátor slouží jako agregátor chyb predikci v čase. Je silné měřítko, používané především pro porovnávání jednotlivých algoritmů na základě naměřených chyb v modelu. `http://www.saedsayad.com/model_evaluation_r.htm`

- **Relative Absolute Error - Relativní absolutní chyba** je velmi podobná relativní čtvercové chybě v tom, že je relativní vzhledem k jednoduché předpovědi. V tomto případě odchylka je celková absolutní chyba namísto celkové čtvercové chyby. To znamená, že relativní absolutní chyba se vypočítává jako celková absolutní chyba a normalizuje se vydělením celkové absolutní chyby jednoduché předpovědi. Matematicky, lze relativní absolutní chybu e_i vyjádřit:

$$\mathrm{RAE} = \mathrm{e_i} = \frac{\sum_{i=1}^{n}|P_{ij} - a_j|}{\sum_{i=1}^{n}|a_j - \hat{a}|} \tag{8}$$

[19] Časová řada stručně představuje soubor takových pozorování x_i, které jsou získány (naměřeny) ve specifickém čase t. Dále můžeme rozlišovat tzv. stochastické a deterministické časové řady nebo aditivní, multiplikativní a smíšené. [3]

Kde P_{ij} je hodnota předpovědi i pro vzorek dat j (z n vzorků), a_j je cílová hodnota pro vzorek j a $\hat{a}$ je dáno vzorcem:

$$\hat{\mathrm{a}} = \frac{1}{n}\sum_{i=1}^{n}|a_i| \tag{9}$$

Pro dokonalé výsledky, čitatel musí být roven 0 a $e_i = 0$. Tím dosáhneme toho, že index e_i se pohybuje v rozmezí od 0 až do nekonečna, kde 0 odpovídá ideálnému případu (dokonalá přesnost výsledků). V praxi se snažíme tuto hodnotu minimalizovat, ne vždy je to možné.

- **Root Relative Squared Error - kořenová relativní čtvercová chyba** je relativní vůči tomu, co by bylo, kdyby byla použita jednoduchá předpověď. To znamená, že relativní čtvercová chyba získává celkovou čtvercovou chybu a normalizuje ji vydělením celkové čtvercové chyby jednoduché předpovědi. Tím, získáním druhé odmocniny relativní čtvercové chyby snižuje chybu do stejných dimenzí jako je samotná jednoduchá předpověď. Matematicky, kořenová relativní čtvercová chyba e_i se vyhodnocuje podle rovnice:

$$\mathrm{RRAE} = \mathrm{e_i} = \sqrt{\frac{\sum_{i=1}^{n}(P_{ij} - a_j)^2}{\sum_{i=1}^{n}(a_j - \hat{a})^2}} \tag{10}$$

Kde P_{ij} je hodnota předpovědi i pro vzorek dat j (z n vzorků), a_j je cílová hodnota pro vzorek j a $\hat{a}$ je dáno vzorcem:

$$\hat{\mathrm{a}} = \frac{1}{n}\sum_{i=1}^{n}|a_i| \tag{11}$$

Znovu pro dokonalé výsledky, čitatel musí být roven 0 a $e_i = 0$.

9.2 Další kriteria

Další kriteria používaná k ohodnocení úspěšnosti klasifikace textu byly následující metriky:

- TP Rate: Míra pravdivých pozitiv (instance správně klasifikované do dané třídy) Příklad výpočtu:

- FP Rate: Míra falešných pozitivních (instance nesprávně klasifikovaná do dané třídy)

- Přesnost: Přesnost je podíl instancí z dokumentů získaných, které jsou relevantní pro informační potřeby uživatele.

- Kappa koeficient Cohen je míra souhlasu v rozsahu hodnot 0-1.

$$\mathrm{K} = \frac{P(A) - P(E)}{1 - P(E)} \tag{12}$$

  Kde P(A) je procentuální soulad mezi realitou a klasifikátorem, P(E) je podíl náhodné shody. K=1 znamená plnou lineární závislost veličin, K=0 je žádnou lineární závislost.

Na základě těchto koeficientů lze hodnotit nejlépe správně klasifikované/predikované lékařské termíny a medicínské zkratky. Dále jsou tyto metriky vhodné i pro určování špatně klasifikovaných instancí. Tyto markery slouží jako evaluace a vytvoří tzv. hodnotící kriteria, poté co byl vytvořen vzor (pattern), kterým se algoritmus řídí a rozhoduje. Význam těchto parametrů je důležitý pro datové analytici a v rámci tohoto projektu pro porovnání datamining algoritmů a následném výběru nejoptimálnějších z ních.

9.3 Zhodnocení výsledků

9.3.1 Jednotkové výsledky

Z poskytnutých datových sad jsem zvolil náhodně 20 různorodých vzorků (20 lékařských textů), které jsem použil jako vstupní data na otestování výsledné aplikaci. V následující tabulce jsou jednotkové výsledky z použitých zpráv, které jsem v dalších pokapitolách využíval jako agregovaný zdroj k analýze a ohodnocení.

	Správně klasifikované zkratky				**Špatně klasifikované / Nenalezené zkratky**				**Doba odezvy (s.)**			
	NBM	SVM	j48	IBk	NBM	SVM	j48	IBk	NBM	SVM	j48	IBk
LZ1	$^{6}/_{9}$	$^{6}/_{9}$	$^{6}/_{9}$	$^{6}/_{9}$	$^{3}/_{9}$	$^{3}/_{9}$	$^{3}/_{9}$	$^{3}/_{9}$	0.68	0.48	0.45	0.64
LZ2	$^{7}/_{10}$	$^{6}/_{10}$	$^{5}/_{10}$	$^{6}/_{10}$	$^{3}/_{10}$	$^{4}/_{10}$	$^{5}/_{10}$	$^{4}/_{10}$	0.23	0.25	0.23	0.27
LZ3	$^{13}/_{25}$	$^{13}/_{25}$	$^{12}/_{25}$	$^{13}/_{25}$	$^{7}/_{25}$	$^{7}/_{25}$	$^{8}/_{25}$	$^{7}/_{25}$	0.62	0.67	0.55	0.57
LZ4	$^{6}/_{8}$	$^{6}/_{8}$	$^{6}/_{8}$	$^{6}/_{8}$	$^{2}/_{8}$	$^{2}/_{8}$	$^{2}/_{8}$	$^{2}/_{8}$	0.29	0.33	0.25	0.37
LZ5	$^{8}/_{19}$	$^{7}/_{19}$	$^{7}/_{19}$	$^{8}/_{19}$	$^{11}/_{19}$	$^{12}/_{19}$	$^{12}/_{19}$	$^{11}/_{19}$	0.69	0.54	0.45	0.39
LZ6	$^{2}/_{10}$	$^{2}/_{10}$	$^{2}/_{10}$	$^{2}/_{10}$	$^{8}/_{10}$	$^{8}/_{10}$	$^{8}/_{10}$	$^{8}/_{10}$	0.38	0.25	0.22	0.34
LZ7	$^{16}/_{29}$	$^{16}/_{29}$	$^{16}/_{29}$	$^{16}/_{29}$	$^{13}/_{29}$	$^{13}/_{29}$	$^{13}/_{29}$	$^{13}/_{29}$	0.51	0.66	0.62	0.43
LZ8	$^{5}/_{10}$	$^{5}/_{10}$	$^{4}/_{10}$	$^{4}/_{10}$	$^{5}/_{10}$	$^{5}/_{10}$	$^{6}/_{10}$	$^{6}/_{10}$	0.14	0.13	0.12	0.22
LZ9	$^{12}/_{18}$	$^{12}/_{18}$	$^{11}/_{18}$	$^{12}/_{18}$	$^{6}/_{18}$	$^{6}/_{18}$	$^{7}/_{18}$	$^{6}/_{18}$	0.27	0.33	0.17	0.20
LZ10	$^{13}/_{17}$	$^{13}/_{17}$	$^{13}/_{17}$	$^{13}/_{17}$	$^{4}/_{17}$	$^{4}/_{17}$	$^{4}/_{17}$	$^{4}/_{17}$	0.33	0.34	0.19	0.22
LZ11	$^{11}/_{18}$	$^{12}/_{18}$	$^{11}/_{18}$	$^{11}/_{18}$	$^{7}/_{18}$	$^{6}/_{18}$	$^{7}/_{18}$	$^{7}/_{18}$	0.34	0.29	0.20	0.28
LZ12	$^{13}/_{20}$	$^{11}/_{20}$	$^{12}/_{20}$	$^{11}/_{20}$	$^{7}/_{20}$	$^{9}/_{20}$	$^{8}/_{20}$	$^{9}/_{20}$	0.46	0.45	0.24	0.55
LZ13	$^{16}/_{18}$	$^{14}/_{18}$	$^{15}/_{18}$	$^{16}/_{18}$	$^{2}/_{18}$	$^{4}/_{18}$	$^{3}/_{18}$	$^{2}/_{18}$	0.37	0.44	0.32	0.37
LZ14	$^{19}/_{32}$	$^{19}/_{32}$	$^{18}/_{32}$	$^{18}/_{32}$	$^{13}/_{32}$	$^{13}/_{32}$	$^{14}/_{32}$	$^{14}/_{32}$	0.5	0.52	0.45	0.48
LZ15	$^{10}/_{28}$	$^{10}/_{28}$	$^{8}/_{28}$	$^{8}/_{28}$	$^{18}/_{28}$	$^{18}/_{28}$	$^{20}/_{28}$	$^{20}/_{28}$	0.32	0.32	0.25	0.38
LZ16	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	$^{24}/_{48}$	0.36	0.37	0.28	0.55
LZ17	$^{16}/_{22}$	$^{16}/_{22}$	$^{16}/_{22}$	$^{16}/_{22}$	$^{6}/_{22}$	$^{6}/_{22}$	$^{6}/_{22}$	$^{6}/_{22}$	0.36	0.37	0.39	0.55
LZ18	$^{18}/_{25}$	$^{18}/_{25}$	$^{19}/_{25}$	$^{19}/_{25}$	$^{8}/_{25}$	$^{8}/_{25}$	$^{7}/_{25}$	$^{7}/_{25}$	0.45	0.52	0.48	0.48
LZ19	$^{24}/_{40}$	$^{25}/_{40}$	$^{24}/_{40}$	$^{24}/_{40}$	$^{16}/_{40}$	$^{15}/_{40}$	$^{16}/_{40}$	$^{16}/_{40}$	0.46	0.47	0.38	0.56
LZ20	$^{12}/_{24}$	$^{12}/_{24}$	$^{10}/_{24}$	$^{12}/_{24}$	$^{12}/_{24}$	$^{12}/_{24}$	$^{14}/_{24}$	$^{12}/_{24}$	0.47	0.5	0.52	0.48

Tabulka 2: Jednotkové výsledky klasifikace náhodných 20 lékařských zpráv

Kde:

LZ je Lékařská zpráva,

Správně klasifikované zkratky $= \frac{P}{N}$, kde P je počet správně opravených relevantních zkratek a N je počet všech skutečně relevantních zkratek

Špatně klasifikované zkratky $= 1 - \frac{P}{N}$,

Doba odezvy je celkový čas k realizace korekce dané LZ.

Tyto zprávy po automatické korekci programem byly procházeny znovu lékařem a kontrolovány, zda nenastala někde kritická chyba při klasifikaci, která by změnila význam celé lékařské zprávy. Z výsledků tabulky 2 je patrné, že u všech algoritmů je skoro ve všech případech úspěšnost větší nebo rovná 50%. Jmenovatel jednotlivých klasifikací naznačuje počet skutečných lékařských zkratek nalezené odborníkem manuální kontrolou. Čitatel ukazuje skutečně správně klasifikované medicínské zkratky či nespávně klasifikované. Mezi nesprávně klasifikovanými můžeme například řadit, špatně klasifikované nebo ignorované zkratky, které program nenalezl nebo nebyl doposud natrénován a ponechal v původním tvaru.

Na první pohled lze odvodit to, že skoro u všech vzorků dat je Naivní Bayes o něco lepší než ostatní algoritmy. Optimalizovaný SVM a Naivní Bayes Multinomial vedou jednoznačně i přesto, že generované rozhodovací stromy a metoda nejbližších sousedů mají velice podobné výsledky vůči ostatním algoritmům. To co je patrné, ale nejvíce a také viditelné jsou časové odezvy. Jako nejrychlejší ve všech případech jsou generované rozhodovací stromy algoritmu J48.

9.3.2 Celkové výsledky

Tato podkapitola popisuje agregovaná zhodnocení jednotlivých algoritmů ze zpráv předchozí kapitoly, na základě kterých jsem dedukoval závěr této práce. V následujících tabulkách jsou vidět porovnávání algoritmů klasifikace medicínského textu.

První tabulka uvádí správnost nebo-li správně zařazené vzorky do tříd a doba odezvy. K výsledkům jsem se dopracoval po simulaci 20ti lékařských zpráv z dostupných anonymizovaných dat, poskytnuté výzkumnou skupiny MRE KIV ve

spolupráci s FN Plzeň.

Algoritmus	**Správně klasifikované (%)**	**Špatně klasifikované (%)**	**Doba odezvy(s)**
NBM	59,2874%	40,7127%	0,4115
SMO	57,8070%	42,1930%	0,4115
J48	55,5471%	44,4529%	0,338
IBk	57,4700%	42,5300%	0,4165

Tabulka 3: Správnost klasifikace textu

Jak je vidět na první pohled nejpřesnější výsledky přináší algoritmus NBM. Na druhém místě je optimalizovaný SVM algoritmus SMO, který pro klasifikaci textu funguje s velmi dobrou přesností, liší se o málo ve srovnání s NBM. Ostatní algoritmy prokazují částečně dobré výsledky, ale také v porovnání poměrně dost chyb. Nicméně nezanedbatelnou rychlost prokazuje stromový algoritus J48. [28] [27]

Druhá tabulka uvádí chyby modelů popsané v kapitole 8.1 na základě data-miningového zpracování těchto 20ti medicínských lékařských textů.

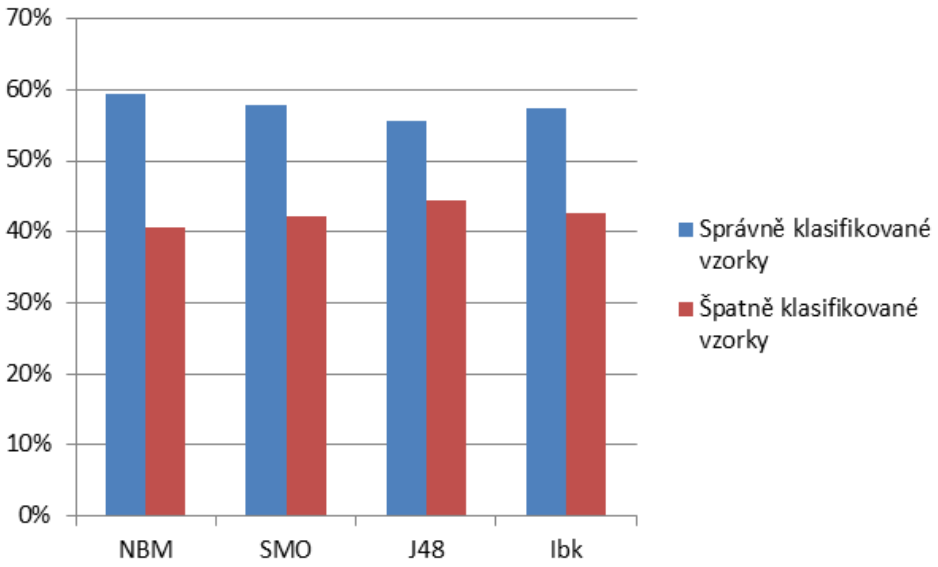
Algoritmus	**Mean Absolute Error**	**Root Mean Squared Error**	**Relative Absolute Error (%)**	**Root Relative Squared Error (%)**
Naivní Bayes	0.2334	0.316	0.413	0.123
SMO	0.732	0.341	0.201	0.634
J48	0.463	0.412	0.238	0.889
IBk	0.672	0.731	0.804	0.298

Tabulka 4: Chybovost klasifikace textu

Z výsledků tabulky 4 lze potvrdit výsledky tabulky 3. Nejpřesnější je znovu algoritmus NBM. Hodnoty má velmi blízké s algoritmem SVM. Graficky lze vidět rozdíly na obrázcích 14 a 15. Jako nejpřesnější a nejúplnější je algoritmus Bayes. [28] [27]

9.4 Porovnávání

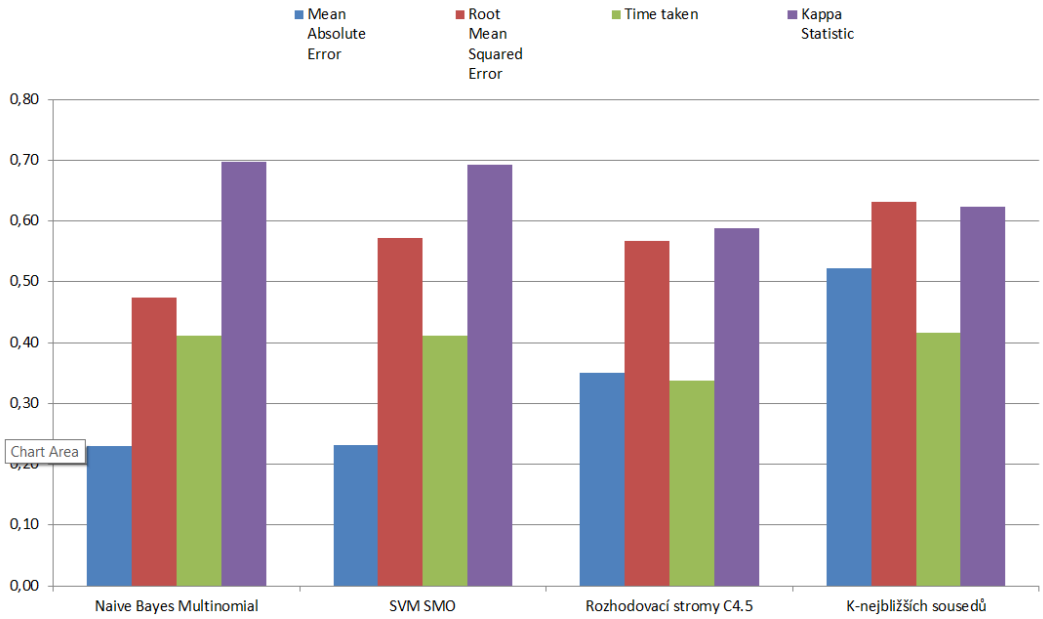
Tato podkapitola nastíňuje a vizualizuje dosažené výsledky jednotlivých kriteriálních hodnot.



Obrázek 14: Porovnání správně opravených zkratek v LZ

Z obrázku je patrné, že nejpřesnější výsledky z dostupných dat měl naivní bayes. Jako dalším s méně správnými výsledky jsou optimalizace algoritmu SVM - SMO a metoda nejmenších sousedů s využití euklidovské vzálenosti. Ostatní algoritmy jsou především rychlé, ale ne až tak přesné.

Přesnost výsledků je silně závislá na trénovacích datech. V oboru medicíny je složité natrénovat modely korektně, kvůli problematice anonymizovaných dat a rizika přetrénování modelu nerovnoměrnou distribucí zkratek či abreviatur. [28] [27]



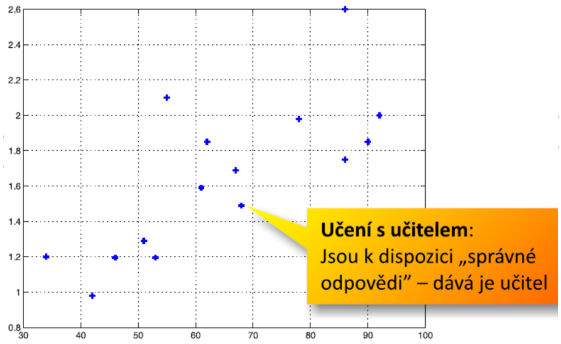
Obrázek 15: Porovnání metrik jednotlivých algoritmů

Na tomto grafu je jasně vidět převládání podobných hodnot. Nicméně hodnoty kappa statistik jsou u naivního bayese a SVM konvergující k 1 výrazně víc než u ostatních algoritmů. Tento faktor ukazuje celkovou lepší přesnost nebo-li nižší chybovost tohoto algoritmu při klasifikaci. Na druhou stranu jako nejrychlejší je vidět, že vyhrává algoritmus C4.5 generující rozhodovací stromy, ale zároveň má také vysokou střední kvadratickou odchylku. Z toho lze dedukovat, že nelze tento algoritmus považovat za dostatečně přesný a vhodný pro korekci lékařských zpráv. Přesnost je důležitější než rychlost v našem případě. Stejné výsledky prokazuje i algoritmus k nejblížších sousedů. Má výrazně nepřesné výsledky a zároveň je časově náročný, kvůli exponenciálně narůstajícím čase při opravě velkého množství dat. Jako nejvýkonnější algoritmus pro aplikace klasifikace lékařských textů lze považovat Multinomiální Naivní Bayes. [27]

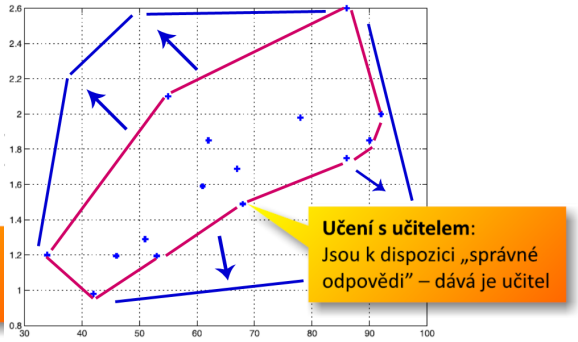
10 Zhodnocení

Úspěšnost jednotlivých algoritmů byla testovaná na dodaných datech výzkumného týmu MRE KIV. 75% vzorků bylo použito pro trénování a zbylých 25% pro testování funkčnosti a kvality algoritmů pro klasifikace medicínského textu.

Hlavní kriteria úspěšnosti byly dvě hlavní veličiny - **úspěšnost(přesnost)** klasifikovaných medicínských terminů a jejich **úplnost**. Úplnost lze vyjádřit porovnáváním následujících obázků:



Obrázek 16: Ukázka trénovacích dat [2]



Obrázek 17: Maximalizace úplnosti trénovacích dat

Toto porovnávání nám udává možný spektrum trénovacích dat. Vizualizace na prvním obrázku ukazuje skutečnost, dokud obrázek napravo znáznorňuje perfektní realitu. Ta je ovšem těžko dosažitelná, protože je potřeba mít všechny medicínské zkratky v češtině, to zahrnuje i data z jiných zdravotnických stanic. Neexistuje však univerzální zdroj anonymizovaných dat, proto projekty jako tento fungují dobře lokálně pro daný region, na základě poskytnutých dat. Osa X grafů vyjadřuje instance testovacích dat, klasicky se jedná o medicínské zkratky, které je potřeba zařadit do dané kategorie. Kategorie jsou vyjádřené osou Y v grafech. Je hezky vidět, že čím širší spektrum dat, tím lépe natrénovaný klasifikátor. Úplnost modelu je klíčová pro lepší výsledky. [2]

10.1 Dosažené výsledky

Výsledná aplikace pro korekce lékařských textů je do jisté míry zavislá hlavně na rychlosti zpracování dotazů k databázi i na kvalitu trénovacích dat. Následující obrázky ukazují výsledky náhodně vybraném medicínském textu nad natrénovanou množinou dat.

(a) Ukázka medicínských textů (b) Opravené lékařské texty

Figure 18: Ukázka úspěšně opravených medicínských zkratek

Je vidět, že 5 zkratek nejsou opravené, za to 10 dalších ano, z čeho vyplývá, že pro tento konkrétní příklad úspěšnost je $\frac{15-5}{15} = 67\%$. Důvody neopravených zkratek mohou být chybějící trénovací data pro konkrétní zkratky nebo nedokonalé odchytávání zkratek regulárním výrazem při větší frekvenci zkratek ve stejném souvětí blízko sebe.

Samotná aplikace je snadno itegrovatelná s jinými databázemi a lze ji použít jako sprostředkovatel mezi ruznými zdroji dat a nemocničních či medicínskcých systémů, které potřebuji korekce specifického textů. Další použití aplikace je například využití jako agregační vrstva pro testování korekce medicínského textu

a další výzkumné aktivity v oboru medicíny a práci s přirozenou řečí, konkrétně v českém jazyce.

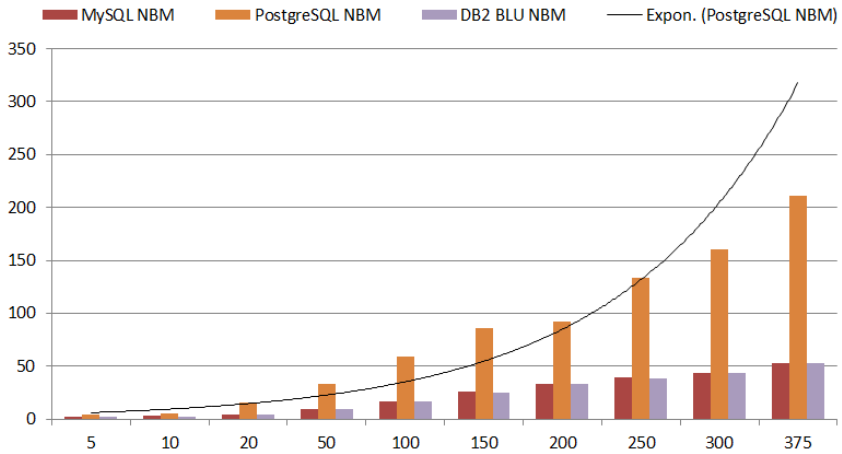
10.2 Průběžný čas zpracování

Optimální dataminingové algoritmy umělé inteligence pro korekci medicínských semistrukturovaných dat v reálném čase jsou do jisté míry závislé i na rychlosti zpracování dotazů k databázi. Za účelem otestování výkonnosti algoritmů jsem zvolil několik databázi - IBM DB2 s optimalizací pro sloupcové vyheldávání, MySQL 5.6.20, PostgreSQL 9.3. Testování dat proběhlo nad všemi algoritmy.

#	**MySQL**				**PostgreSQL**				**IBM DB2**			
	NBM	SVM	j48	IBk	NBM	SVM	j48	IBk	NBM	SVM	j48	IBk
5	2,11	3,11	1,75	2,16	4,67	5,67	4,41	5,48	1,23	2,25	1,06	1,46
10	2,92	3,92	2,12	3,31	5,1	6,10	4,35	5,71	1,93	3,39	1,79	3,07
20	4,79	5,79	4,70	5,46	15,59	16,59	14,86	16,47	4,44	5,07	3,81	5,38
50	9,67	10,67	9,35	10,12	33,68	34,68	33,03	34,45	9,61	10,15	8,76	9,97
100	16,63	17,63	15,90	17,08	59,57	60,57	59,11	60,03	15,97	17,39	15,70	16,70
150	25,50	26,50	25,11	26,45	86,13	87,13	86,04	86,59	24,74	25,97	25,04	26,18
200	33,33	34,33	33,13	34,32	92,3	93,30	92,19	92,76	32,57	33,46	32,69	34,26
250	39,00	40,00	38,73	39,25	133,4	134,40	132,91	133,99	38,95	39,58	37,79	38,73
300	43,55	44,55	43,30	44,20	160,3	161,30	159,31	160,38	42,60	44,02	43,14	43,33
375	53,29	54,29	53,15	53,69	211,3	212,30	210,46	212,21	53,23	54,19	52,45	53,64

Tabulka 5: Jednotlivé doby odezvy (s.) korekce lékařských zpráv

Na tabulce jsou vidět výsledky jednotlivých měření rychlosti zpracování databázových dotazů. Levý sloupec naznačuje počet opravovavaných medicínských zpráv souběžně a ostatní sloupce uvádí rychlost odezvy celkové zpracování textů v sekundách na jednotlivých databázích pro jednotlivé algoritmy. Na první pohled je patrné, že výsledky databáze IBM DB2 s optimalizací pro sloupcové ukládání jsou značně rychlejší. Celková výkonnost databáze záleží na přiděleném a dostupném transakčním výkonu databázovém stroji. Existuje velké množství faktorů, které mohou zpomalit celkovou korekci zkratek při používání relační databáze.



Obrázek 19: Závislost rychlosti opravy lékařských zpráv na čase a databázi

Na obrázku 19 jsou vidět grafické výsledky naměřených hodnot tabulky 5. Osa X znázorňuje počet lékařských zpráv ke korekci a osa Y je celková doba zpracování. Tato doba zahrnuje odeslání http requestu, jeho následné zpracování, průběžné trénování modelu a automatická korekce, za pomoci naimplementovaného datamining API a následné vrácení http response s opraveným textem. Je jasně vidět, že s narůstajícím množstvím medicínských semistrukturovaných dat je i jejich korekce časově náročnější, jedná-li se o hromadnou opravu lékařských textů. Naopak, jedná-li se o menší množství textů je naimplementované řešení zcela vhodné pro opravu dat v reálném čase. Exponenciální křivka naznačuje možný trend v časové náročnosti databáze PostgreSQL v závislosti na počtu dat při nynějším nastavení. Dále na obrázku je patrné, že databáze PostgreSQL je výrazně pomalejší. Důvodem je to, že jsem nastavil maximální počet **Connections Objects** pro **Connection Pool**[20] na 4. Tím je v grafu hezky vidět, že transakční výkon databáze má důležitou roli při používání algoritmů umělé inteligence v reálném čase. Graf používá pouze data algoritmu Naivní Bayes pro lepší přehlednost.

[20]Connection Pool - je vyrovnávací paměť databázových připojení udržovaných tak, že připojení mohou být znovu použity, když budou vyžadovány v budoucnosti

10.3 Úspěšnost jednotlivých algoritmů

Kriteriem úspěšnosti po otestování programu a jeho konečné celkové zhodnocení byly po celou dobu 2 hlavní faktory - přesnost a úplnost.

Do velké míry je úplnost závislá na trénovacích datech. Nynější řešení je univerzální a lze ho pouze vylepšovat o například automatické učení na základě napojení na FN Plzeň a postupnému rozšiřování slovníků lékařských textů. Dále tato aplikace slouží jako podklad k výzkumným prácím týmu MRE KIV. Porovnávané algoritmy fungují naprosto odlišným způsobem, nicméně se skoro všechny z nich dopracovaly k podobným výsledkům. Rozdíly jsou pouze časové a do určité míry vykazují chyby. Každý z algoritmů má své výhody a nevýhody, který by měl být použit, či ignorován pro účely této diplomové práci.

- SVM SMO - Algoritmus je komplikovaný a v praxi někdy nevyužitý kvůli nedostatku dat. Mnohdy výstupy si vystačí s binární klasifikací a často ani nedochází k porovnávání více kernelů pro vice dimenzí.

- Implementace algoritmu C4.5 - Algoritmus je velice rychlý, ale na úkor toho, že je velmi citlivý při výběru prahových atributů. Z toho lze vydedukovat, že je více náklonný k chybám a nepřesným výsledkům. Tím je i přílíš závislý na trénovacích datech.

- Metoda nejmenších sousedů funguje dobře, ale není vhodná pro lékařské texty z důvodu velké přítomnosti hlučných příznaků dat, které mohou v určitých případech ovlivnit celkové výsledky špatně. Například:

"Krevní obraz: B-Le: 13,50 B-Ery: 4,83 B-Hb: 157 B-HTK: 0,463 B-Obj ery.: 96 B-Hb ery: 32,5 B-Hb konc: 338 B-Erytr.křivka: ".

Zde zkratky jako B-Hb, B-Hb ery., B-Hb konc. mají vždy jiný význam a uvažování algoritmu by vykazovalo nepřesné výsledky.

- Naivní Multinomiální Bayes prokázal nejlepší výsledky v celkovém projektu. Jednoduchá metoda uvažování rovnoměrné distribuci funguje úspěšně a to s velmi dobrými výsledky. Kvalitní natrénování modelu s dostupnými daty by dopomohlo k jeho zdokonalení.

Další možnosti pro aplikace dataminingových algoritmů nad lékařskými texty v českém jazyce můžou být kombinace několik algoritmů například Bayes a SVM, případně zahrnout metody Itemsets a N-gramy, které je možno znovu zkombinovat a otestovat nad dostupnými daty. Další vylepšení tohoto projektu lze dosáhnout lepším předzpracování výsledků. Například implementací pokročilejší filtrace dat před trénováním, ale za úkor delšího časového zpracování či celkové výkonnosti nynějšího řešení.

V rámci tohoto projektu jsem naimplementoval metodu pro filtraci dat, která ignoruje zkratky, které nezná. Důvodem je snížít riziko špatné klasifikace jednotlivých nalezených zkratek. Je to vhodná metoda pro úplnost textů. Důvodem je, že medicínské texty jsou odborné a jakákoliv chyba v reálném světě může stát zdraví daného pacienta.

11 Závěr

Náplní této práce bylo navrhnout, implementovat a otestovat optimální metody dataminingu pro zpracování medicínských semistrukturovaných dat. Metody dataminingu jsem použil pro korekci lékařských zkratek a medicínských termínů. Ze všech testovaných metod jsem zhodnotil a vybral tu nejefektivnější pro účely této diplomové práci.

Na základě praktických zkušeností při používání dataminingových algoritmů a práci s lékařem jsem se dopracoval k výsledkům této diplomové práce. Zjistil jsem, že trénovací data jsou základem dobré klasifikace. Toto byl důvod poměrně podobných výsledků jednotlivých algoritmů. Výrazné rozdíly byly časové odezvy. Jemnější rozdíly byly patrné v chybovosti a přesnosti jednotlivých algoritmů.

Problematika umělé inteligence v medicíně je běh na dlouhou trať. Výstupem této práce je program umožňující automatickou korekci lékařských textů, který má sloužit jako podpůrnou komponentu výzkumného týmu MRE KIV. Na základě dostupných dat jsem dosáhnul nejpřesnější a nejúplnější výsledky s algoritmem Multinomial Naive Bayes. Též bych tento algoritmus doporučoval jako nejlepší pro zpracování textů.

Výstup této práce lze použít jako komponentu pro již existující systém. Aplikace je navržená jako API, kterou lze nasadit do provozu na interním systému nebo hostovat a spravovat extérně a volat přes API metody.

Zadání bylo úspěšně splněno a celková očekávaná úspěšnost algoritmů při opravě lékařských textů je minimálně 60%.

12 Reference

[1] doc. Dr. Ing. KLEČKOVÁ, Jana — přednášky z předmětu Databázové systémy a metody zpracování dat. [online] Použito na str. 15

[2] Ing. EKŠTEIN, Kamil Ph.D. — přednášky Teorie Kognitivních systémů. [online] Použito na str. 20, 58, 69

[3] Ing. ŤOUPAL, Tomáš Ph.D. — přednášky Modely řízení ve firmě. [online] Použito na str. 50

[4] BUNEMAN, Peter; DAVIDSON, Susan; FERNANDEZ, Mary; SUCIU, Dan - Adding Structure to Unstructured Data, International Conference on Database Theory [online] (1997). ISBN: 3-540-62222-5. Použito na str. 15

[5] SEBASTIANI, Fabrizio. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34(1):1–47, 2002. ISSN 0360-0300. Použito na str. 19

[6] KUO, Cheng-Ju; LING, Maurice HT; LIN, Kuan-Ting and HSU, Chun-Nan, "BIOADI: a machine learning approach to identifying abbreviations and definitions in biological literature", [online] DOI:10.1186/1471-2105-10-S15-S7, Online ISSN 1471-2105 (2009). Použito na str. 10

[7] POKORNÝ M., SNÁŠEL J., KOPECKÝ, M. Dokumetografické informační systémy. 2. vydání. Praha: Karolinum, 2005. ISBN 80-246-1148-1. Použito na str. 20, 21, 22

[8] KUČERA, Martin - Modifikace metody Naive-Bayes o prvky Itemsets metody. Plzeň: Západočeská univerzita. Fakulta aplikovaných věd. Katedra informatiky a výpočetní techniky, Diplomová práce (2002). Použito na str. 21

[9] USAMA Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, From Data Mining to Knowledge Discovery in Databases. [Online] DOI: http://dx.doi.org/10.1609/aimag.v17i3.1230. Použito na str. 23

[10] SOUMEN Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, Wei Wang, Data Mining Curriculum: A Proposal (Version 1.0), [Online] URL: `http://www.kdd.org/curriculum/index.html` April 30, 2006. Použito na str. 23

[11] EIBE, Frank; BOUCKAERT, Remco R. , Naive Bayes for Text Classification with Unbalanced Classes, Computer Science Department, University of Waikato, New Zealand. ISBN:3-540-45374-1 978-3-540-45374-1 DOI:10.1007/11871637 49 (2013). Použito na str. 25, 26

[12] RAGHAVAN, Prabhakar - Text Classification : The Naïve Bayes algorithm - Adapted from Lectures by Prabhakar RAGHAVAN (Yahoo and Stanford) and Christopher Manning (Stanford), Stanford University (2013). [online] URL `http://cecs.wright.edu/~tkprasad/courses/cs707/L13NaiveBayesClassify.ppt` Použito na str. 25, 26, 27

[13] BARBER, David - Bayesian Reasoning and Machine Learning, [online] URL `http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/240415.pdf` ISSN: 0163-5700 DOI:10.1145/2636805.2636813 (2008). Použito na str. 26

[14] PLATT, John - Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, ISBN: 0-262-19416-3 (1998). Použito na str. 29

[15] CHANG, Chih-Chung; LIN, Chih-Jen. "LIBSVM: A library for support vector machines". ACM Transactions on Intelligent Systems and Technology (2011). DOI:10.1145/1961189.1961199 Použito na str. 30

[16] ZANNI, Luca. Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems (2006). ISSN: 1532-4435 EISSN: 1533-7928. Použito na str. 30

[17] doc. Ing. ŽIŽKA, Jan CSc - Support vector machines (SVM): Algoritmy podpůrných vektorů [online]. posl. revize 9. 12. 2004 [cit. 2012–04–25]. Vyňatek z přednášek `http://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf`. Použito na str. 30

[18] RIFKIN, Ryan - "Everything Old is New Again: a Fresh Look at Historical Approaches in Machine Learning", Ph.D. thesis (2002). Použito na str. 30, 31

[19] QUINLAN, J. R., C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, (1993). ISBN 1-55860-238-0. Použito na str. 32, 33

[20] QUINLAN, J. R., Improved use of continuous attributes in c4.5. Journal of Artificial Intelligence Research, 4:77-90, ISSN 1076 - 9757. (1996). Použito na str. 33

[21] PATERA, Jan - Rozhodovací stromy. Brno: FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY, Diplomová práce (2008). Použito na str. 32

[22] ALTMAN, N. S., "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. ISSN 0003-1305 (Print) (1992). 34, 36

[23] COOMANS, D.; MASSART, D.L., "Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules". DOI:10.1016/S0003-2670(01)95359-0 (1982). Použito na str. 34, 36

[24] Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JB. "Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization". Journal of Chemical Information and Modeling. Print Edition ISSN: 1549-9596. Web Edition ISSN: 1549-960X (2006). Použito na str. 34, 35

[25] Weka [online]. [cit. 2012–04–11]. URL `http://weka.wikispaces.com`. Použito na str. 37, 38, 39

[26] Mohd Fauzi bin Othman,Thomas Moh Shan Yau - "Comparison of Different Classification Techniques", Control and Instrumentation Department, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Skudai, Malaysia. ISSN: 2231-2307 (2007). Použito na str. 33

[27] Mark A. Hall, Correlation-based Feature Selection for Machine Learning, Computer Science Department, University of Waikato, New Zealand (1999). PhD. thesis. Použito na str. 49, 55, 56, 57

[28] Ms S. Vijayarani, Ms M. Muthulakshmi, "Comparative Analysis of Bayes and Lazy Classification Algorithms". Department of Computer Science, School of Computer Science and Engineering, India. ISSN (Print): 2319-5940, ISSN (Online): 2278-1021 (2013). Použito na str. 49, 55, 56

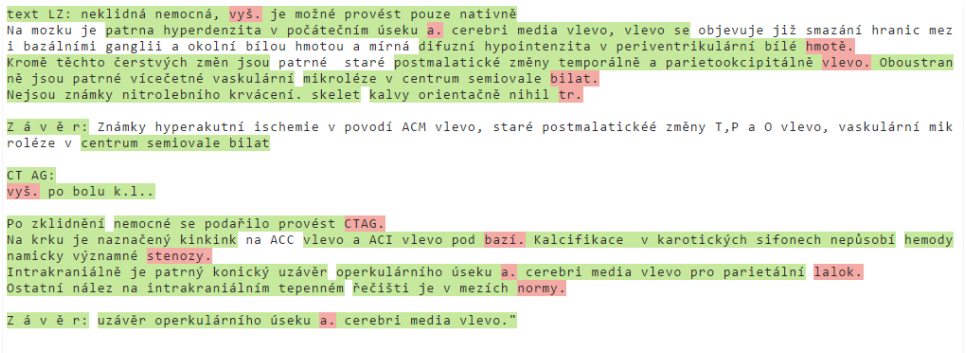
[29] J. Scott Armstrong and Fred Collopy. "Error Measures For Generalizing About Forecasting Methods: Empirical Comparisons" (PDF). International Journal of Forecasting 8 (1). DOI:10.1016/0169-2070(92)90008-w. (1992). Použito na str. 49

Seznam obrázků

Seznam tabulek

13 Příloha A

V této příloze se nachází větší obrázky, ukázky a příklady



Obrázek 20: Příklad aplikace regulárního výrazu pro hledání zkratek

```
@relation 'dx-weka.filters.unsupervised.attribute.StringToWordVector'

@attribute class {dx,dex,dextra,'?'}
@attribute ACC numeric
@attribute ACI numeric
@attribute ACM numeric
@attribute I numeric
@attribute M1 numeric
@attribute Závěr numeric
@attribute a numeric
@attribute až numeric
@attribute beze numeric
@attribute bifurkaci numeric
@attribute centrální numeric
@attribute dex numeric
@attribute do numeric
@attribute dx numeric
@attribute embolu numeric
@attribute expanzivními numeric
@attribute hraniční numeric
@attribute intrakraniálně numeric
@attribute ischemické numeric
@attribute již numeric
@attribute k numeric
@attribute kinkingu numeric
@attribute ložisko numeric
@attribute mírnými numeric
@attribute na numeric
@attribute nekrotickA1ch numeric
@attribute nález numeric
@attribute oblasti numeric
@attribute patrném numeric
@attribute přes numeric
@attribute při numeric
@attribute rekanalizace numeric
@attribute rekanalizaci numeric
@attribute rozsahu numeric
@attribute s numeric
@attribute sin numeric
@attribute v numeric
@attribute vpravo numeric
@attribute vyjádření numeric
@attribute významnosti numeric
@attribute vč numeric
@attribute zde numeric
@attribute změn numeric
@attribute změnami numeric
@attribute Vyjádřené numeric
@attribute šlo numeric

@data
{0 dextra,1 1,2 1,3 1,8 1,9 1,15 1,18 1,19 1,23 1,26 1,28 1,33 1,37 1,41 1,42 1,43 1,44 1}
{0 dextra,3 1,4 1,7 1,10 1,13 1,14 1,21 1,22 1,27 1,28 1,30 1,31 1,32 1,34 1,35 1,38 1,40 1,44 1,47 1}
{0 dextra,1 1,5 1,6 1,8 1,11 1,12 1,15 1,16 1,17 1,20 1,24 1,25 1,29 1,33 1,36 1,38 1,39 1,45 1,46 1}
```

Obrázek 21: Ukázka trénovacích dat na obrázku 5, přetransformované na vektory slov pro účely dataminingu

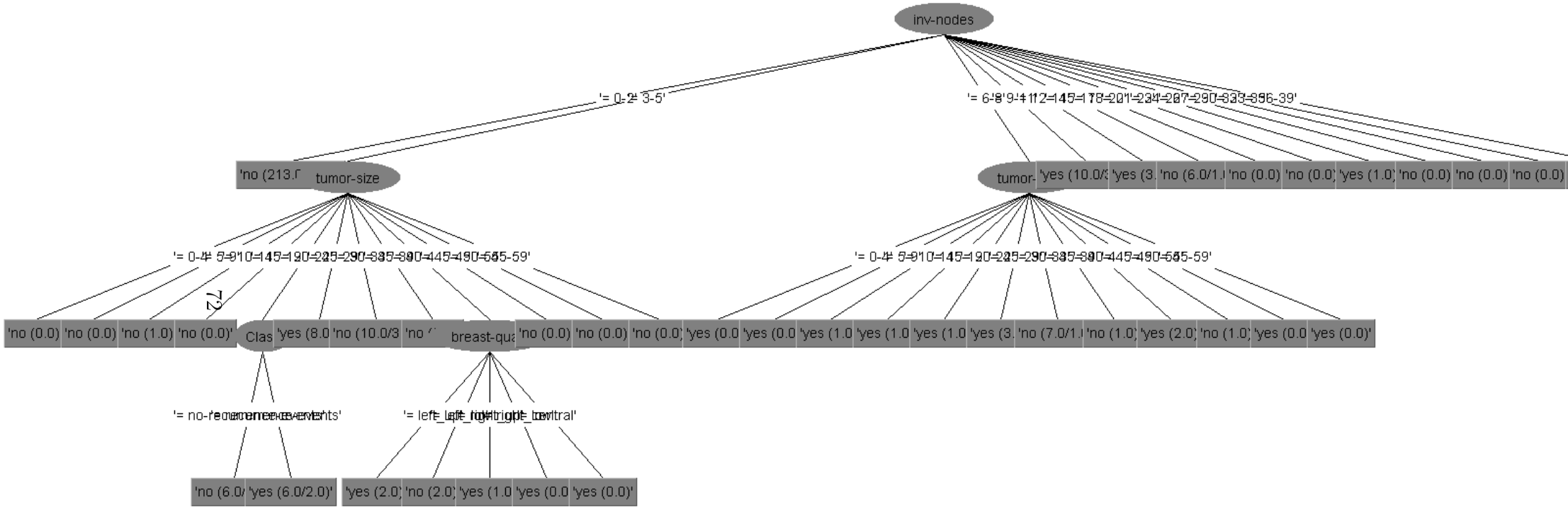


Figure 22: Vygenerovaný rozhodovací strom algoritmu J48 pro presenci rakovinu prsou (ANO/NE) v závislosti na velikosti nádoru či velikosti uzlu v hlavní části prsu.

14 Příloha B

Obsah DVD:

Data

database - DDL a DML skripty jednotlivých databázi, obsahujicích trénovací data

*.xls - Zhodnocení nad testovacích datech. Grafy a tabulky, včetně manuálně připravovaný trénovací korpus

Prerekvizity

Java JDK, JRE, Tomcat a Eclipse - prerekvizity pro nasazení aplikace

Program

dep-jar - knihovny a závislosti, potřebné pro kompilace projektu

bin - zkompilované zdrojové kódy do spustitelné podoby

src - kompletní okomentovaný zdrojový kód programu

WebContent - Front-endová část navrženého řešení

mrekiv.war - zkomprimovaný soubor J2EE projektu

build.xml - soubour pro sestavení J2EE projektu

Text

tex - text diplomové práce ve formátu LaTeX

pdf - text diplomové práce ve formátu Adobe PDF