Verb-sense disambiguation

Úvod do strojového učení (v počítačové lingvistice) (PFL054)

Petra Galuščaková
galuscakova@gmail.com

1. Popis úlohy

Rozpoznanie významu slova v danom kontexte je veľmi dôležitá úloha, ktorú je ale zároveň veľmi ťažké riešiť automaticky. Zvlášť dôležité je správne určenie významu slovesa vo vete. Od významu slovesa sa totiž ďalej odvíja význam celej vety.

Cieľom projektu je rozpoznať na základe daných informácií význam slovesa v danom kontexte. Konkrétne je potrebné rozpoznať správny význam u dvoch slovies – *přihlížet* a *odpovídat*. Slovo přihlížet môže nadobúdať dva významy (Tabuľka1), slovo *odpovídat* tri významy (Tabuľka2) [9].
Je teda potrebné riešiť klasifikačný problém a priradiť ku slovu v danom kontexte jeden význam. Na riešenie tejto úlohy sú využité metódy strojového učenia.

	Valenčný rámec	**Význam**	**Príklad**
1	ACT (obl, 1), PAT (obl, 3, jak)	Pasívne sledovať	přihlížel nečinně neštěstí
2	ACT (obl, 1), PAT (obl, k+3)	Brať v úvahu	při vyhodnocování přihlédli k počtu bodů

Tabuľka1: Významy slova *přihlížet*

	Valenčný rámec	**Význam**	**Príklad**
1	ACT (obl, 1), ADDR (obl, 3), PAT (opt, na+4), EFF (obl, 4, aby, ať, zda, že, cont), MANN (typ)	Dávať odpoveď	odpověděl mu na jeho dotaz pravdu
2	ACT (obl, 1), ADDR (opt, 3,), PAT(obl, za+4), MEANS (typ, 7)	Mať na zodpovednosť	odpovídá za své děti
3	ACT (obl, 1), PAT (obl, že+3), REG (typ, 7)	Byť v zhode	řešení odpovídá svými vlastnostmi požadavkům

Tabuľka2: Významy slova *odpovídat*

Pozn. V slovníku VALLEX sú dané štyri významy slova *odpovídat*. Štvrtým významom je reagovať (Pr. vojáci odpověděli střelbou), ktorý má valenčný rámec ACT (obl, 1) PAT (opt, na+4) EFF(obl, 7)

2. Dáta

Pre každé zo skúmaných slov je daných sedem súborov s trénovacími a sedem súborov s testovacími dátami (Tabuľka3). Pre každé slovo sú dané morfologické, syntaktické, idiomatické informácie, informácie o životnosti a informácie z WordNetu. Každý súbor pozostáva zo zoznamu hodnôt jednotlivých atribútov, ktoré sú oddelené čiarkou. Pre každý typ súboru je daný popis a možné hodnoty jednotlivých atribútov. Celkovo je pre každú inštanciu daných 1175 atribútov. Dáta pochádzajú z dizertačnej práce Jiřího Semeckého.

Súbory	Typ dát	Príklady atribútov	Počet atribútov
{test,train}.a.data	životnosť	Počet životných a neživotných substantív vo vete, existencia životného substantíva v nominatíve, závislého na slove, ...	18
{test,train}.i.data	idiomatika	Existencia idiomatického výrazu „v úvahu“ (napospas, na trh, k dobru, …) vo vete	118
{test,train}.m.data	morfológia	Slovný druh, (rod, číslo, pád, čas, …) daného slovesa (predchádzajúceho, predpredchádzajúceho, následujúceho, „ponásledujúceho“ slova)	75
{test,train}.n.data	morfológia (z hodnôt m.data vytvorené boolean hodnoty)	Je prvá (druhá, tretia, ...) pozícia morfologického tagu slovesa (predchádzajúceho, predpredchádzajúceho, následujúceho, „ponásledujúceho“ slova) rovná A (C, D, J, ...)	705
{test,train}.s.data	syntax	Existencia reflexívneho se, si (predložky v nominatíve, predložky „proti“ v datíve) závislého na slovese, ...	131
{test,train}.w.data	Informácie z WordNetu	Existencia substantíva zo sémantickej triedy Agentive (Instrument, Social, Time, …) vo vete, existencia substantíva zo sémantickej triedy Agentive (Instrument, Social, Time, …) závislého na slovese vo vete	128
{test,train}.x.data	Význam danej inštancie	Má slovo prvý, druhý (tretí) význam	1
Spolu			1175

Tabuľka3: Popis trénovacích a testovacích súborov

Dáta sú rozdelené na trénovaciu a testovaciu časť. Pre sloveso *přihlížet* obsahujú trénovacie dáta 66 inštancií a testovacie dáta 33 inštancií. Pre sloveso *odpovídat* obsahujú trénovacie dáta 65 inštancií a testovacie dáta 32 inštancií. Niektoré ďalšie informácie o dátach sú uvedené v Tabuľke4. V Tabuľke5 a v Tabuľke6 sa nachádzajú distribúcie jednotlivých tried pre obe slová.

	přihlížet	***odpovídat***
Celkový počet atribútov	1175	1175
Počet konštantných atribútov v trénovacích dátach	512	625
Počet konštantných atribútov v celých dátach	646	650
Počet atribútov v testovacích dátach, ktorých hodnota sa nevyskytuje v trénovacích dátach	24	164
Počet atribútov, ktorých množina hodnôt v trénovacích a množina hodnôt v testovacích dátach sú rôzne	212	215

Tabuľka4: Popis atribútov

	1	2	Spolu
Trénovacie dáta	36	30	66
Testovacie dáta	18	15	33

Tabuľka5: Distribúcia tried pre slovo *přihlížet*

	1	2	3	Spolu
Trénovacie dáta	21	4	30	65
Testovacie dáta	10	2	20	32

Tabuľka6: Distribúcia tried pre slovo *odpovídat*

3. Použité algoritmy

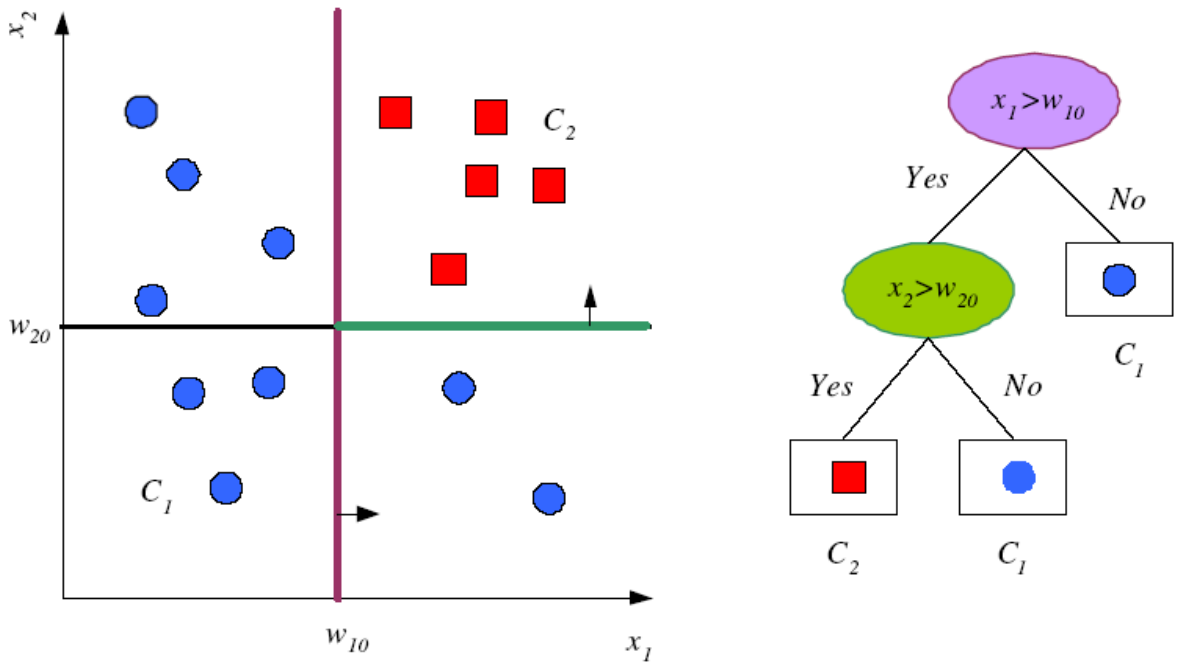
Pri riešení úlohy som použila algoritmy strojového učenia Pri vyhodnocovaní som ďalej použila bootstrapping. Pri výbere atribútov použitých na trénovanie a rozhodovanie týchto algoritmov boli použité algoritmy založené na entropii, korelácii a konzistencii dát. Na rozhodovanie o štatistickej významnosti odlišnosti použitých algoritmov som použila Welchov test.

3. 1. Algoritmy strojového učenia

V práci boli použité tri algoritmy strojového učenia s učiteľom: rozhodovacie stromy, Naive Bayes Classificator a Support Vector Machines.

- Rozhodovacie stromy

Cieľová rozhodovacia funkcia je v tomto prípade popísaná stromom. Vrcholom stromu odpovedajú vybrané atribúty. Hrany vychádzajúce z daného vrcholu zas zodpovedajú hodnotám daného atribútu. Listy stromu zodpovedajú jednotlivým klasifikáciám. Pri trénovaní modelu sa postupne stavia strom. Pritom je potrebné vyberať atribúty, ktorým budú zodpovedať vrcholy. Tento výber môže prebiehať na základe hodnoty entropie. Klasifikácia potom prebieha pomocou prechodu stromu. Príklad rozhodovacieho stromu je znázornený na Obrázku1.



Obrázok1: Príklad rozhodovacieho stromu [1, 9]

Pri stavbe stromu, ktorý presne zodpovedá trénovacím dátam hrozí pretrénovanie. Úspešnosť je tak možné zlepšiť orezávaním stromu. Pri orezávaní je napríklad možné stanoviť hodnotu o ktorú sa v každom kroku musí hodnota entropie zvýšiť. V prípade, že sa nezvýši, zastavíme stavbu stromu. [1, 11]

- Naive Bayes

Tento prístup je založený na pravdepodobnosti, konkrétne na Bayesovom pravidle. Najlepšia hypotéza je v tomto prípade najpravdepodobnejšia hypotéza. Dôležitým predpokladom klasifikátoru je podmienená nezávislosť hodnôt atribútov. Klasifikácia prebieha podľa formuly

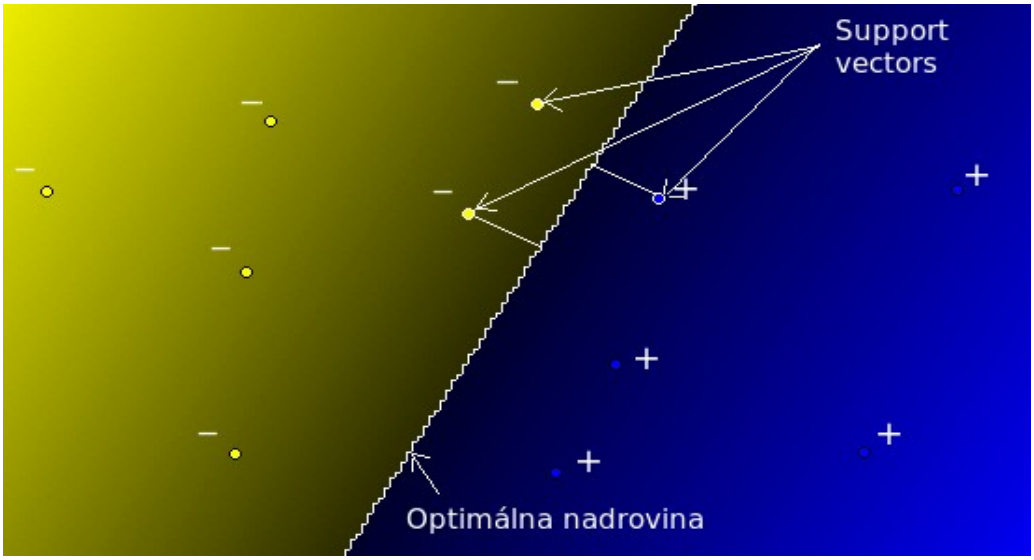
$$y_{NBC} = argmax_{y \in Y} P(y) \prod_{i=1}^{n} P(a_i|y)$$

Kde:
y_{NBC} je výsledná klasifikácia,
P(y) je relatívna frekvencia výskytu klasifikácie y v trénovacích dátach
$P(a_i \mid y)$ je podmienená pravdepodobnosť hodnoty atribútu a_i pri klasifikácii y

Výsledky klasifikátoru je možné zlepšiť pomocou vyhladzovania, konkrétne je možné použiť napríklad Laplacelovo vyhladzovanie. Ak je pre nejaké a_i , y je pôvodná hodnota $P(a_i \mid y)$ v trénovacích dátach rovná nule, potom je pri vyhladzovaní rovná nejakej malej nenulovej hodnote. Ostatné hodnoty pravdepodobností je potom tiež potrebné upraviť tak, aby ostali podmienky pravdepodobnosti zachované. U Laplacelovho vyhladzovania je to dosiahnuté tak, že sa ku počtu výskytov daného atribútu pri klasifikácii y vždy pripočítané kladné číslo. Toto kladné číslo je parametrom Laplacelovho vyhladzovania, ktorý je možné nastaviť.
[1, 6]

- Support Vector Machines (SVM)

SVM prevádza problém klasifikácie inštancií na optimalizačný problém. Nad množinou hodnôt atribútov môžeme skonštruovať priestor. Potom hľadáme jej nadrovinu, ktorá najlepšie oddeľuje rôzne klasifikované inštancie.
Medzi dvoma rôzne klasifikovanými množinami najprv nájdeme nadrovinu, ktorá ich oddeľuje. Najlepšia nadrovina je potom tá, ktorej vzdialenosť od najbližších bodov je čo najväčšia. Body, ktoré sa nachádzajú najbližšie ku oddeľovacej rovine sa nazývajú support vectors.



Obrázok2: Popis SVM

V najjednoduchšom prípade je možné rôzne klasifikované inštancie oddeliť lineárnou nadrovinou. Niekedy je potrebné pre správnu klasifikáciu priestor transformovať. Táto transformácia je daná jadrovou funkciou. Jadrová funkcia môže byť napríklad lineárna, polynomiálna a radiálna. Polynomiálna funkcia má tvar $k(x_i, x_j) = (x_i . x_j)^d$, kde d je parameter, ktorý je možné nastaviť. Radiálna funkcia má tvar $k(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$, kde γ je parameter, ktorý je možné nastaviť.
Nie vždy je však možné všetky inštancie od seba oddeliť, v niektorých prípadoch sú triedy neseparabilné. V tomto prípade môžeme stanoviť, aká veľká bude penalizácia zle klasifikovaných inštancií. Na oddelenie hľadáme nadrovinu, ktorá je „najlacnejšia“, teda súčet penalizácií za zle klasifikované inštancie bude čo najmenší.
Popísaný algoritmus klasifikuje dáta do dvoch tried. Pri klasifikácií do viacerých tried je možné problém transformovať na niekoľko klasifikácií do dvoch tried. Je pritom možné použiť prístupy 1-versus-1 a 1-versus-all.
Pri prístupe 1-versus-1 postupne klasifikujeme dáta do každej dvojice tried vybranej z pôvodnej množiny tried. Nakoniec vyberieme pomocou hlasovania výslednú klasifikáciu. Pri prístupe 1-versus-all konštruujeme funkcie tak, že vyberieme vždy jednu triedu z pôvodnej množiny tried a do druhej triedy zaradíme inštancie všetkých ostatných tried. Tento postup opakujeme pre všetky takto skonštruované triedy a novú inštanciu klasifikujeme nakoniec do triedy, ktorá má oddeľovaciu nadrovinu najbližšie. [8, 10, 12]

3. 2. Bootstrapping

Bootstrapping je metóda, ktorá slúži k zlepšeniu odhadu chyby pri vyhodnocovaní algoritmu. Je zvlášť vhodný pri odhade chyby na zašumených dátach. Pri bootstrappingu je algoritmus opakovane trénovaný a vyhodnocovaný na podmnožinách celkovej množiny dát. Prvky do týchto množín sú pritom z celej množiny dát vyberané náhodne s opakovaním. [1]
V práci som použila vyvážený bootstrapping. Pri vyváženom bootstrappingu sa každá inštancia musí vyskytnúť n-krát, kde n je počet opakovaní bootstrappingu. To ale nemusí znamenať, že v každej iterácii sa vyskytujú všetky inštancie. Napríklad prvá inštancia sa môže pri prvej iterácii použiť dvakrát a vôbec sa nepoužije v druhej iterácii. Výber inštancií pre danú iteráciu vznikne tak, že sa n-krát skopírujú všetky inštancie, vytvorí sa permutácia tejto postupnosti a v prvej iterácii sa zoberie prvých r prvkov tejto postupnosti, kde r je počet inštancií. Vyvážený bootstrapping môže ďalej zlepšiť presnosť bootstrappingu. [4]
Pri bootstrappingu nás zaujíma aritmetický priemer vypočítaných hodnôt a štandardná chyba. Štadradnú chybu vypočítame ako

$$Se = \frac{s}{\sqrt{n}}$$

kde s je štandardná odchýlka a n je počet pozorovaní. Zo štandardnej chyby je potom možné vypočítať konfidenčný interval: [x – Se * q; x +Se * q], kde x je aritmetický priemer a q je kvantil normálneho rozdelenia odpovedajúci požadovanej spoľahlivosti. [7]

3. 3. Výber atribútov

Úspešnosť algoritmov strojového učenia je možné zvýšiť aj výberom atribútov na základe ktorých sa daný algoritmus trénuje. Nie vždy je výhodné použiť všetky atribúty, ktoré sú k dispozícii, úspešnosť je často možné zvýšiť výberom podmnožiny atribútov. Táto podmnožina by mala byť relevantná vzhľadom ku klasifikačnej triede a neredundantná. Na zistenie redundancie a relevancie atribútov je možné použiť napríklad entropiu, konzistenciu alebo koreláciu.
Na výber atribútov som použila funkcie *cfs* a *consistency* z knižnice *FSelector* [5] a funkciu *Boruta* z knižnice *Boruta* [2]. Funkcia *cfs* používa pri výbere atribútov metriku založenú na entropii a korelácii. Priestor atribútov prehľadáva pomocou hladového algoritmu. Funkcia *consistency* používa metriku založenú na konzistencii atribútov, na prehľadávanie používa tiež hladový algoritmus. Funkcia *Boruta* na rozdiel od toho používa iteratívny algoritmus a využíva pritom klasifikátor randomForest [3].

3. 4. Welchov t-test

Tento štatistický test je úpravou Studentovho t-testu. Je možné ho použiť na porovnanie dvoch algoritmov. Máme daný vektor A čísel 1 a 0, ktoré zodpovedajú správnym, respektíve chybným výsledkom jedného algoritmu. Podobne máme daný vektor B čísel 1 a 0, ktoré zodpovedajú výsledkom druhého algoritmu. Tieto vektory vznikli porovnaním vektoru výsledkov predikovaných daným algoritmom so skutočnými testovacími hodnotami. V prípade, že algoritmus rozhodol o danom pozorovaní správne, pridáme do množiny 1, inak pridáme 0. Máme hypotézu H, podľa ktorej predpokladáme, že sa algoritmy nelíšia. Alternatívnou hypotézou je, že sa algoritmy líšia. V prípade, že zamietneme hypotézu H, platí alternatívna hypotéza. Na základe stredných hodnôt a rozptylu a počtu pozorovaní sa vypočíta hodnota t:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

kde X_i je počet stredná hodnota, s_i^2 je rozptyl a N_i je počet výsledkov daného algoritmu, teda veľkosť množiny A alebo B.

Hodnotu t potom porovnáme s tabuľkovými hodnotami na základe toho, akú spoľahlivosť testu požadujeme. Na základe tohto porovnania potom hypotézu prijmeme alebo zamietneme. [13]

4. Implementácia

Úlohu som riešila v jazyku R. Najprv som spravila jednoduchú klasifikáciu a získala som tak odhad baseline. Potom som skúsila každý algoritmus natrénovať na všetkých atribútoch. Ďalej som použila niekoľko funkcií na výber relevantných atribútov a skúsila som natrénovať algoritmy iba na týchto atribútoch. Ak to algoritmus umožňoval, vyskúšala som rôzne nastavenia parametrov tohto algoritmu. Nakoniec som pomocou bootstrappingu získala lepšiu presnosť algoritmov pri nastaveniach, pri ktorých dosahoval najlepšie výsledky.

Pri vyhodnocovaní som použila presnosť. Keďže je daných 33 inštancií na testovanie v prípade slova *přihlížet* a 32 inštancií v prípade slova *odpovídat*, zodpovedá jedna chyba pri vyhodnotení v oboch prípadoch približne trom percentám.

4. 1. Baseline

Jednoduchý odhad je možné získať tak, že sa každej inštancii priradí klasifikácia, ktorý sa vyskytuje v trénovacích dátach najčastejšie. Pre slovo *přihlížet* je to jeho prvý význam, pre slovo *odpovídat* jeho tretí význam. Presnosti takéhoto odhadu pre obe slovesá sú v Tabuľke7. Tento odhad môžeme pokladať za baseline. Pri slovese *přihlížet* majú oba významy veľmi podobný počet výskytov. U slovesa *odpovídat* sa najčastejšie objavuje jeho tretí význam. Menej častý je prvý význam, druhý význam sa vyskytuje veľmi málo. Preto je slepý odhad u tohto slovesa lepší ako u druhého slovesa, napriek tomu, že sa klasifikuje do viacerých tried.

Presnosť - *přihlížet*	**Presnosť - *odpovídat***
54.5	62.5

Tabuľka7: Baseline pre jednotlivé slová

4. 2. Výber atribútov

Pri výbere atribútov som použila funkcie, ktoré automaticky z množiny atribútov vyberajú podmnožinu atribútov, ktoré sú relevantné a neredundantné. Konkrétne som použila algoritmy z knižníc *Boruta* a *FSelector*.

Z knižnice *Boruta* som použila algoritmus *Boruta*: *boruta.model <- Boruta(x ~ ., data.train, confidence=0.95, maxRuns=11)*, kde *confidence* udáva požadovanú spoľahlivosť a *maxRuns*

maximálny počet iterácií algoritmu. Parameter *maxRuns* má hodnotu 11, prebehne preto nanajvýš 10 iterácií algoritmu.
Z knižnice *FSelector* som použila algoritmy *cfs*: *model.cfs <- cfs(x ~ ., data.train)* a *consistency*: *model.consistency <- consistency(x ~ ., data.train)*.
Zoznamy atribútov vybraných pomocou týchto funkcií sú uvedené v Tabuľke8.
Okrem toho som sa pokúsila vybrať atribúty na základe toho, čo lingvisticky reprezentujú. Pri disambiguácii by mohli byť napríklad dôležité morfologické značky predchádzajúceho a následujúceho slova. Dáta n.data je možné odvodiť zo súboru m.data. Preto som skúsila použiť iba atribúty m1 až m75.

Metóda	***Přihlížet***	***Přihlížet* – počet atribútov**	***Odpovídat***	***Odpovídat* – počet atribútov**
Boruta (confidence =0.95, maxRuns=11)	a1, a3, m33, m46, m47, m49, m50, m61, m62, m65, n433, s1, s21, s73	14	a10, a11, a12, m46, m47, m48, m49, m50, m63, m65, n433, n510, s22, s85, s127	15
FSelector: cfs	a3, a6, a9, m65, s1, s21, s73	7	a10, a11, m48, s22, s85, s127	6
FSelector: consistency	a2, m17, m65	3	a1, a2, a11	3

Tabuľka8: Atribúty vybrané pomocou rôznych metód

Pozn. Vybrané atribúty nie sú usporiadané podľa dôležitosti.

Pre slovo *odpovídat* sú atribúty vybrané pomocou *cfs* podmnožinou atribútov vybraných pomocou funkcie *Boruta*. Pre slovo *přihlížet* sa 5 zo 7 atribútov vybraných pomocou *cfs* nachádza aj v množine atribútov vybraných pomocou algoritmu *Boruta*. Pri slove *přihlížet* bol atribút m65 vybraný pomocou všetkých troch použitých algoritmov. Pri slove *odpovídat* to je atribút a11. 6 atribútov získaných pomocou funkcie *Boruta* je zhodných obe slová.

4. 3. Rozhodovacie stromy
Pri práci s rozhodovacími stromami som využila knižnicu *rpart*. Najprv som z trénovacích dát vytvorila rozhodovací strom: *rpart.model.all <- rpart(x ~ ., data.train.cut, control=ctrl)*, pomocou ktorého som potom klasifikovala testovacie dáta: *out.rpart.all <- predict(rpart.model.all, data.test, type="class")*. V objekte *ctrl* sa nachádzajú parametre pomocou ktorých je možné strom upravovať: *ctrl <- rpart.control(minsplit=msi, cp=cpi)*. Parameter *minsplit* udáva minimálny počet pozorovaní, ktoré musia ležať v danom uzle na to, aby ho bolo možné rozdeliť a parameter *cp* slúži na orezávanie stromu a udáva minimálnu hodnotu „zlepšenia“ potrebnú na to, aby sa strom ďalej štiepil.
Predikcia nemôže pracovať s atribútmi, ktoré nadobúdajú v testovacích dátach iné hodnoty ako v trénovacích dátach. Preto som tieto atribúty z trénovacích aj a z testovacích dát najprv odstránila.
Okrem výberu atribútov som skúsila ovplyvniť nastavenia parametrov *minsplit* a *cp*. Pre každý výber atribútov som pre každú hodnotu parametru *minsplit* od 1 po 30 pri kroku 5 vyskúšala hodnoty parameteru *cp* = 0.0001, 0.001, 0.01, 0.1 a 0.25. Úspešnosť sa pre slovo *přihlížet* pri žiadnom z týchto nastavení nezvýšila. Pre slovo *odpovídat* sa, naopak, úspešnosť pri niektorých nastaveniach parametrov značne zvýšila.

Najlepšie výsledky boli dosiahnuté pre slovo *přihlížet* pri všetkých použitých atribútoch, atribútoch vybraných pomocou algoritmov *Boruta*, *cfs* a *consistency* pri všetkých vyskúšaných nastaveniach a

pri morfologických atribútoch pre parametre *minsplit* = 1, 6, 26, *cp* = 0.0001, 0.001, 0.01, 0.1, 0.25 a *minsplit* = 11, 16, 21, *cp* = 0.1, 0.25. Najlepší výsledok zodpovedá štyrom chybám z 33 inštancií v testovacích dátach. Presnosť výsledkov, pri použití vyše tisícich atribútov a pri použití jediného atribútu je rovnaká. To znamená, že pri použití rozhodovacích stromov je možné namiesto veľkého počtu atribútov možné použiť iba veľmi malý počet atribútov, tieto atribúty musia byť ale vhodne vybraté.

Pre slovo *odpovídat* boli najlepšie výsledky dosiahnuté pre atribúty získané pomocou funkcie *Boruta* pri parametroch *minsplit* = 6, *cp* = 0.0001, 0.001, 0.01 a pre atribúty vybrané pomocou funkcie *cfs* a pre parametre *minsplit* = 1, 6 a *cp* = 0.0001, 0.001, 0.01. Najlepšie výsledky v týchto prípadoch zodpovedajú dvom chybám v testovacích dátach (32 inštancií). Najlepšie výsledky pre slovo *přihlížet* sú uvedené v Tabuľke9, najlepšie výsledky pre slovo *odpovídat* sú v Tabuľke10.

Výsledný strom pre slovo *přihlížet* pri použití všetkých atribútov a implicitných parametrov rozhoduje na základe dvoch atribútov – m65 a n236. Tento strom je znázornený na Obrázku3. Pri niektorých nastaveniach je však rovnaká presnosť dosiahnutá pri použití jediného atribútu – m65. V ďalších prípadoch je rovnaký výsledok dosiahnutý pri použití troch, štyroch až piatich atribútov. Vo všetkých týchto prípadoch bol ale použitý aj atribút m65.
Strom pre slovo *odpovídat* rozhoduje na základe piatich atribútov (a10, a11, m48, s22, s127) v prípade, že boli atribúty vybrané pomocou funkcie *Boruta* alebo pomocou funkcie *cfs* s parametrami *minsplit = 6, cp* = 0.0001, 0.001, 0.01 alebo na základe šiestich atribútov (a10, a11, m48, s22, s85, s127) v prípade, že boli atribúty vybrané pomocou funkcie *cfs* a použité parametre boli *minsplit*=1*, cp*=0.0001, 0.001, 0.01 . Zodpovedajúce stromy sú znázornené na Obrázku4 a Obrázku5. Napriek tomu, že prvý model má o jeden atribút menej (chýba atribút s85), dosiahnuté výsledky sú rovnaké.
Z vybraných atribútov pre slovo *přihlížet* je vidieť, že najlepší výsledok je možné dosiahnuť aj pri jedinom použitom parametre. Napriek tomu sú pri rôznych nastaveniach aj pri rôznych množinách vybratých atribútov použité ďalšie atribúty. Tieto atribúty je teda možné pokladať za nadbytočné a ideálne by malo byť možné ich počet redukovať a to napríklad práve pomocou množiny vybraných atribútov. To sa napríklad podarilo pomocou výberu parametrov pomocou funkcie *consistency*. Vybraný bol v tomto prípade jediný parameter, ktorý bol potom aj použitý pri vytvorení stromu a odhad na základe tohto stromu dosahoval najvyššiu úspešnosť.

Pre slovo *odpovídat* je zaujímavý výber atribútov s najlepšou presnosťou. V jednom takomto prípade strom používa päť atribútov, v druhom prípade je použitých tých istých päť atribútov a jeden navyše. Je to spôsobené orezávaním stromu, tieto dva prípady sa od seba líšia hodnotou parametru *minsplit.* Najvyššia presnosť bola dosiahnutá vtedy, keď sa vyberali atribúty z menšieho počtu kandidátov. V jednom prípade sa použilo 5 atribútov z 11 kandidátov, v druhom prípade 5, prípadne 6 atribútov zo 6. Ak sa však pri trénovaní použili všetky možné atribúty, potom boli dosiahnuté výsledky značne horšie. Výsledok sa však môže zhoršiť aj pri príliš malej množine kandidátov alebo pri zle vybraných atribútoch. Pri oboch slovách je teda vhodné „predspracovať“ výber atribútov. To môže nielen značne znížiť výpočtovú náročnosť, ale aj zlepšiť výsledky.

Parametre modelu	**Počet atribútov**	**Počet použitých atribútov**	**Použité atribúty**	**Presnosť**
Všetky dáta, implicitné hodnoty	1151	2	m65, n236	**87.9**
Všetky dáta	1151	1, 2, 4	m65 (m8, m61, s1, n236)	**87.9** (minsplit = 1, 6, 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
Boruta	10	1, 2, 3	m65	**87.9**

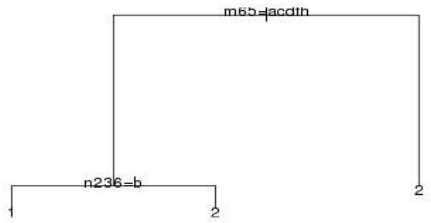
			(m61, s1)	(minsplit = 1, 6, 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
FSelector: cfs	7	1.2	m65 (s1)	**87.8** (minsplit = 1, 6, 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
FSelector: consistency	1	1	m65	**87.9** (minsplit = 1, 6, 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
Morfologické dáta	64	1, 3, 5	m65, (m8, m41, m49, m61)	**87.9** (minsplit = 1, 6, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25) (minsplit = 11, 16, 21 cp = 0.1, 0.25)

Tabuľka9: Porovnanie jednotlivých nastavení pre rozhodovacie stromy – slovo *přihlížet*

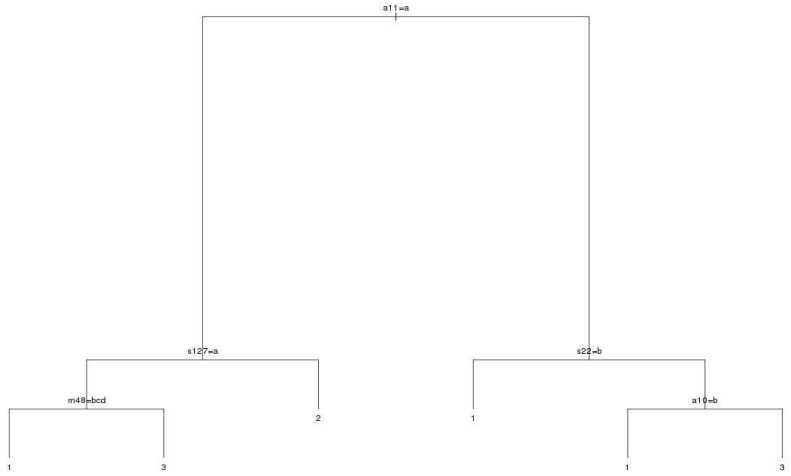
Parametre modelu	**Počet atribútov**	**Počet použitých atribútov**	**Použité atribúty**	**Presnosť**
Všetky dáta, implicitné hodnoty	1011	1	a11	**78.1**
Všetky dáta	1011	1, 2, 3	a11 (m3, s127)	**78.1** (minsplit = 1, 6 cp = 0.1, 0.25) (minsplit = 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
Boruta	11	5	a10, a11, m48, s22, s127	**93.8** (minsplit = 6, cp = 0.0001, 0.001, 0.01)
FSelector: cfs	6	5, 6	a10, a11, m48, s22, s127 (s85)	**93.8** (minsplit = 1, 6, cp = 0.0001, 0.001, 0.01)
FSelector: consistency	2	1	a11	**78.1** (minsplit = 1, 6 cp = 0.1, 0.25) (minsplit = 11, 16, 21, 26 cp = 0.0001, 0.001, 0.01, 0.1, 0.25)
Morfologické dáta	61	1	m48	**81.3** (minsplit = 1, 6, 11, 16, 21, 26, cp = 0.25) (minsplit = 26, cp = 0.1)

Tabuľka10: Porovnanie jednotlivých nastavení pre rozhodovacie stromy – slovo *odpovídat*

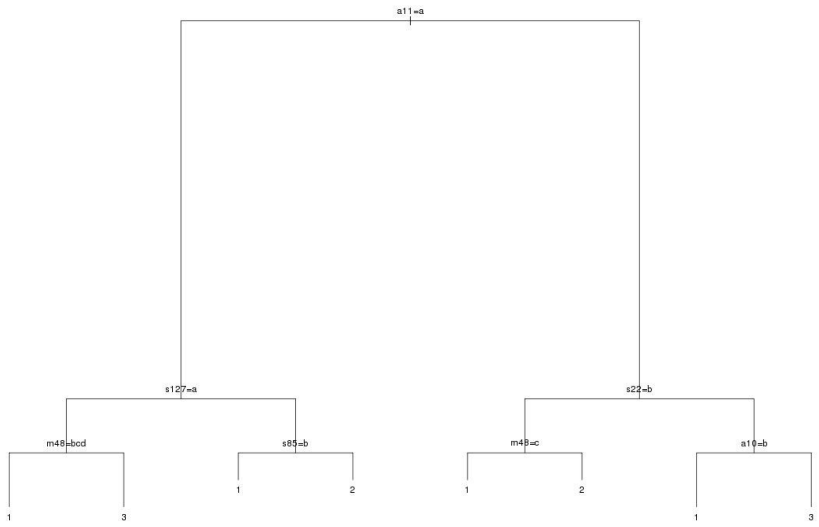
Pozn. Ak sú použité atribúty v zátvorke, potom boli použité iba pri niektorých nastaveniach, presnosť ale ostala rovnaká.



Obrázok3: Strom pre slovo *přihlížet*. Použité boli všetky atribúty a implicitné hodnoty parametrov



Obrázok4: Strom pre slovo *odpovídat*. Použité boli atribúty vybrané pomocou funkcie *Boruta,* a hodnoty parametrov *minsplit*=6*, cp*= 0.0001, 0.001, 0.01
Rovnaký strom je vytvorený pre atribúty vybramé pomocou funkcie *cfs* a pre hodnoty parametrov *minsplit =6, cp* = 0.0001, 0.001, 0.01



Obrázok5: Strom pre slovo *odpovídat*. Použité boli atribúty vybrané pomocou funkcie *cfs,* a hodnoty parametrov *minsplit*=1*, cp*=0.0001, 0.001, 0.01

4. 4. Naive Bayes

Pre prácu s algoritmom Naive Bayes som použila funkciu z knižnice *e1071*. Algoritmus som vyskúšala na rôznych podmnožinách atribútov. Pomocou príkazu *bayes.model.all <- naiveBayes(x ~ ., data = data.train.cut, laplace=l)* som vytvorila model. Parameter *laplace* zodpovedá použitej hodnote pri Laplacelovom vyhladzovaní. Z modelu som potom získala výslednú klasifikáciu: *out.bayes.all <- predict(bayes.model.all, data.test.cut, type="class")*. Algoritmus, podobne ako *rpart*, nemôže pracovať s atribútmi, ktoré majú v testovacích dátach hodnoty, ktoré sa v trénovacích dátach nenachádzali. Preto som tieto atribúty nepoužívala.

Najlepšie výsledky boli v prípade slova *přihlížet* dosiahnuté pri použití všetkých dát bez vyhladzovania a pri použití Laplacelovho vyhladzovania s parameterami vyhladzovania 17 až 30. Pri slove *odpovídat* boli najlepšie výsledky dosiahnuté pri použití všetkých atribútov a pri použití Laplacelovho vyhladzovania s parametrami 3 a 4.

Niektoré dosiahnuté výsledky sú horšie ako baseline. Pre slovo *přihlížet* boli výsledky dosiahnuté pri výbere atribútov pomocou funkcie *consistency* dokonca horšie ako náhodný odhad. To je ale spôsobené malým množstvom použitých atribútov, v tomto prípade tréning prebiehal na jedinom atribúte. Ak bol teda tento atribút vybraný zle, spôsobilo to veľmi zlé výsledky. Naopak, pri slove *odpovídat* boli výsledky získané pri použití atribútov vybraných pomocou funkcie *consistency* druhý najlepší. Táto funkcia teda môže byť vhodná na výber atribútov pre algoritmus Naive Bayes, v prípade, že je ale použité malé množstvo atribútov, môže dávať zlé výsledky.

Výsledky sa pri rapídnom zmenšení počtu použitých atribútov veľmi nezhoršili, pri slove *odpovídať* sa dokonca zlepšili. Pomocou dobrého výberu atribútov je teda možné znížiť výpočtovú zložitosť pri zachovaní presnosti. Výsledky sa v prípade slova *odpovídat* značne zlepšili pri použití vyhladzovania. Najlepšie výsledky sú uvedené v Tabuľke11.

Metóda	**Počet atribútov - *přihlížet***	**Presnosť - *přihlížet***	**Počet atribútov - *odpovídat***	**Presnosť - *odpovídat***
Všetky dáta	1151	**60.6**	1011	**53.1**
Všetky dáta + vyhladzovanie	1151	**60.6** (laplace = 17..30)	1011	**81.3** (laplace = 3, 4)
Boruta	10	**54.5**	11	**54.5**
FSelector: cfs	7	**57.6**	6	**57.6**
FSelector: consistency	1	**45.5**	2	**72.7**

Tabuľka11: Porovnanie jednotlivých metód pre Naive Bayes

4. 5. Support Vector Machine (SVM)

Na prácu s SVM je možné využiť funkciu, ktorá sa nachádza v knižnici *e1071*. Niektoré atribúty v dátach nadobúdajú viac ako dve hodnoty. Tieto atribúty je potrebné najprv previesť na binárne vektory tak, že pre každú hodnotu atribútu je vytvorený nový binárny vektor s hodnotami TRUE a FALSE. To je možné napríklad pomocou funkcie *as.dummy* z knižnice *cba*: *data.train.cut.svm.dummy <- as.dummy(data.train.cut.svm[,-length(data.train.cut.svm)])*.

Z trénovacích a testovacích dát som najprv odstránila všetky konštantné atribúty, ktoré sa v nich vyskytovali. Ďalej som odstránila atribúty, ktoré mali odlišné hodnoty v trénovacích a testovacích dátach. Trénovanie som potom spustila pomocou *svm.model.all <- svm(x ~ ., data.train.cut.svm, kernel="linear", cost=1)* a predikciu následne pomocou *svm.result.all.temp <- predict(svm.model.all, data.test.cut.svm, type="class")*. Parameter *kernel* udáva tvar jadrovej funkcie a parameter *cost* penalizáciu zle klasifikovaných inštancií.

V prípade slova *odpovídat* je potrebné klasifikovať do troch tried. Funkcia *svm* umožňuje klasifikáciu do viacerých tried. Napriek tomu som túto možnosť nepoužila, problém sama redukovala na problém klasifikácie do dvoch tried a použila som iba binárnu klasifikáciu.

Klasifikáciu som pustila na tri prípady (<1 : 2,3>, <2 : 1,3>, <3 : 1,2>) a počítala som koľkokrát inštancia spadne do danej klasifikácie. Potom som vybrala triedu, do ktorej bola inštancia klasifikovaná najčastejšie.
Ďalej som skúsila algoritmus spustiť na podmnožinách vybraných atribútov. Skúsila som zmeniť aj jadro modelu, parameter cost (hodnoty 1 až 30), pri polynomiálnom jadre stupeň polynómu (hodnoty 3, 4, 5) a pri jadre radial hodnotu gamma (hodnoty 0.1, 1, 10). Najlepší výsledok pre slovo *přihlížet* bol dosiahnutý pre všetky atribúty, ktoré bolo možné použiť, pre polynomiálne jadro, so stupňom polynómu 3 a hodnotami cost = 17 až 30 a so stupňom polynómu 4 a hodnotami cost = 29 a 30. Pre slovo *odpovídat* boli najlepšie výsledky dosiahnuté pri atribútoch získaných pomocou algoritmu *Boruta*, lineárnom jadre a hodnotách cost = 1 až 30. Najlepšie výsledky pre jednotlivé jadrové funkcie sú uvedené v Tabuľke12, najlepšie výsledky pre rôzne výbery atribútov sú uvedené v Tabuľke13.
Z tabuľky 13 je vidieť, že pre slovo *přihlížet* pri veľkom znížení počtu atribútov v jednom prípade nedochádza ku veľkému zníženiu presnosti (dochádza ku jednej ďalšej chybe) ale podstatne sa zníži výpočtová zložitosť. Pre slovo *odpovídat* dokonca dochádza pri zmenšení počtu atribútov ku značnému zlepšeniu presnosti. Záleží však na metóde výberu atribútov, pre morfologické atribúty napríklad presnosť o niečo klesá.

Jadro	**Presnosť - *přihlížet***	**Presnosť - *odpovídat***
Lineárne	**93.9** (všetky atribúty, cost = 1..30)	**90.6** (atribúty Boruta, cost = 1..30)
Polynomiálne	**96.9** (všetky atribúty, degree = 3, cost = 17..30) (všetky atribúty, degree = 4, cost = 29, 30)	**87.5** (atribúty cfs, degree = 3..5, cost = 1..30)
Radiálne	**93.9** (atribúty Boruta, gamma = 0.1, cost = 1) (atribúty cfs, gamma = 0.1, cost = 1)	**87.5** (atribúty Boruta, gamma = 0.1, cost = 1..30) (atribúty cfs, gamma = 0.1, cost = 2..30)

Tabuľka12: Porovnanie jednotlivých metód pre SVM, najlepšie výsledky pre jednotlivé jadrové funkcie

Metóda	**Počet atribútov - *přihlížet***	**Presnosť - *přihlížet***	**Počet atribútov - *odpovídat***	**Presnosť - *odpovídat***
Všetky dáta, implicitné hodnoty	451	60.6	334	68.8
Všetky dáta	451	96.9	334	78.1
Boruta	5	93.9	9	90.6
FSelector: cfs	6	93.9	5	87.5
FSelector: consistency	0	-	1	78.1
Morfologické dáta	18	90.9	22	75

Tabuľka13: Porovnanie jednotlivých metód pre SVM, najlepšie výsledky pre daný výber atribútov

5. Výsledky

Najlepšia presnosť bola dosiahnutá pre slovo *přihlížet* pri algoritme SVM pri trénovaní na všetkých atribútoch, a pre slovo *odpovídat* pri rozhodovacích stromoch pri trénovaní na atribútoch ktoré boli vybrané pomocou algoritmov *Boruta* a *cfs*. Prehľad presností pre jednotlivé metódy sa nachádza v

Tabuľke14 a v Tabuľke15. Tabuľky 16 a 17 sú kontingenčné tabuľky pre najlepšie klasifikácie jednotlivých slov. V prípade slova *přihlížet* je zle klasifikovaný jediný prípad z 33, v prípade slova *odpovídat* sú to dva prípady z 32. Kontingenčné tabuľky pre slovo *odpovídat* sú pre oba výbery atribútov rovnaké.

Metóda	**Presnosť - *přihlížet***
Baseline	54.5
Rozhodovacie stromy (najlepší výsledok)	87.9
Naive Bayes (najlepší výsledok)	60.6
SVM (najlepší výsledok)	96.9

Tabuľka14: Porovnanie najlepších výsledkov pri jednotlivých metódach - slovo *přihlížet*

Metóda	**Presnosť - *odpovídat***
Baseline	62.5
Rozhodovacie stromy (najlepší výsledok)	93.8
Naive Bayes (najlepší výsledok)	81.3
SVM (najlepší výsledok)	90.6

Tabuľka15: Porovnanie najlepších výsledkov pri jednotlivých metódach - slovo *odpovídat*

	1	**2**
1	18	0
2	1	14

Tabuľka16: Kontingenčná tabuľka pre najlepší výsledok – *přihlížet*
Použitá metóda SVM

	1	**2**	**3**
1	10	0	0
2	1	1	0
3	1	0	19

Tabuľka17: Kontingenčná tabuľka pre najlepší výsledok – *odpovídat*
Použité rozhodovacie stromy

5.1. Bootstrapping

Na získanie lepšej informácie o presnosti algoritmov som použila bootstrapping. Pre každý algoritmus som použila nastavenia a výber atribútov pri ktorých boli získané najlepšie výsledky. Použila som pritom vyvážený bootstrapping.

Pri boostrappingu som použila knižnicu *boot*. Bootstrapping sa potom spustí pomocou príkazu *boot(data = data.cut.svm, statistic = boot.svm.funct, R = iterations, sim = "balanced")*, kde *boot.svm.funct* je funkcia, ktorá sa v každej iterácii spustí a jej výstupom je presnosť pre vstupné dáta, hodnota *iterations* udáva počet opakovaní funkcie a *balanced* označuje, že ide o vyvážený boostrapping.

Trénovacie a testovacie dáta som spojila a v každom kroku som ich rozdelila na 2/3 trénovacích a 1/3 testovacích inštancií. Spolu som spravila vždy 30 iterácií. Výsledná presnosť je potom priemerom presností získaných v jednotlivých iteráciách. Získané hodnoty pre slovo *přihlížet* sú takmer rovnaké ako najlepšie hodnoty pre konkrétny algoritmus. Získané hodnoty pre slovo

odpovídat sú vo všetkých prípadoch o niečo nižšie ako najlepšie hodnoty. Pri algoritme SVM je rozdiel najlepšej hodnoty a hodnoty získanej pomocou bootstrappingu takmer 30 percent. To môže byť spôsobené tým, že nastavenia parametrov algoritmu SVM sú vhodné iba pre konkrétne jedno rozdelenie dát na trénovacie a testovacie. Pre slovo *přihlížet* dosiahol najvyššiu presnosť algoritmus SVM, pre slovo *odpovídat* to boli rozhodovacie stromy.
Dosiahnuté hodnoty bootstrappingu pre rôzne metódy strojového učenia sú uvedené v Tabuľke18 a v Tabuľke 19.

Metóda	**Presnosť**	**Štandardná chyba**	**Konfidenčný interval (0,95)**
Rozhodovacie stromy	87.88	9.57	[69,12 ; 100]
Naive Bayes	63.64	7.66	[48,61 ; 78,65]
SVM	96.97	6.91	[83,4 ; 100]

Tabuľka18: Porovnanie jednotlivých metód – Bootstrapping – slovo *přihlížet*

Metóda	**Presnosť**	**Štandardná chyba**	**Konfidenčný interval (0,95)**
Rozhodovacie stromy	81.25	8.04	[65,49 ; 97,01]
Naive Bayes	78.13	10.13	[58,28 ; 97,98]
SVM	65.63	4.26	[57,28 ; 73,98]

Tabuľka19: Porovnanie jednotlivých metód – Bootstrapping – slovo *odpovídat*

Pre slovo *přihlížet* leží vypočítaná presnosť pre rozhodovacie stromy v konfidenčnom intervale algoritmu SVM, rovnako leží presnosť algoritmu SVM v konfidenčnom intervale algoritmu pre rozhodovacie stromy. To znamená, že algoritmy rozhodovacie stromy a SVM by mohli byť „rovnako dobré“. Pre slovo *odpovídat* leží presnosť pre rozhodovacie stromy v konfidenčnom intervale algoritmu Naive Bayes a presnosť algoritmu Naive Bayes leží v konfidenčnom intervale pre rozhodovacie stromy. V tomto prípade by sme mohli pokladať algoritmy rozhodovacie stromy a Naive Bayes za „rovnako dobré“. Presnosť algoritmu SVM ďalej leží v konfidenčných intervaloch rozhodovacích stromov aj algoritmu Naive Bayes.

5. 2. Porovnanie algoritmov

U algoritmov som porovnávala ich najlepšie výsledky. Na porovnanie som použila Welchov t- test a využila som na to funkciu t.test. Hranica štatistickej významnosti bola stanovená na 0.95. Pre obe slová dosahovali všetky porovnania štatisticky významné rozdiely. V žiadnom prípade nemali žiadne dva algoritmy natoľko podobné výsledky, aby neboli štatisticky významné.

	Baseline	**Rpart**	**Bayes**	**SVM**
Baseline	x	-	-	-
Rpart	-	x	-	-
Bayes	-	-	x	-
SVM	-	-	-	x

Tabuľka20: Štatisticky významná podobnosť algoritmov pre slovo *přihlížet*

	Baseline	Rpart	Bayes	SVM
Baseline	x	-	-	-
Rpart	-	x	-	-
Bayes	-	-	x	-
SVM	-	-	-	x

Tabuľka21: Štatisticky významná podobnosť algoritmov pre slovo *odpovídat*

6. Diskusia

Dôležitou časťou úlohy disambiguácie významu je výber atribútov, na ktorých prebieha tréning a následné ladenie parametrov daného algoritmu. Rôzne spôsoby výberu atribútov môžu byť viac alebo menej vhodné pre rôzne algoritmy strojového učenia. Napriek tomu by mohli existovať atribúty, ktoré „sú vhodnejšie“ pri väčšine algoritmov strojového učenia. Atribúty, ktoré boli vybrané pomocou rôznych algoritmov viackrát a teda sú vhodnými kandidátmi na takéto „kvalitné“ atribúty sú uvedené v Tabuľke22.

Atribút	Význam	Použitý pri slovese
a1	Počet životných substantív vo vete	obe
a2	Počet neživotných substantív vo vete	obe
a3	Existencia životného substantíva	*přihlížet*
a10	Počet životných substantív vo vete	*odpovídat*
a11	Počet neživotných substantív vo vete	*odpovídat*
m46	Slovný druh slova následujúceho slova	obe
m47	Bližší popis slovného druhu následujúceho slova	obe
m48	Rod následujúceho slova	*odpovídat*
m49	Číslo následujúceho slova	obe
m50	Pád následujúceho slova	obe
m65	Pád ďalšieho slova za následujúcim slovom	obe
n433	Je následujúce slovo predložka?	obe
s1	Existencia reflexívneho se, závislého na slovese	*přihlížet*
s21	Existencia predložky v datíve, závislej na slovese	*přihlížet*
s22	Existencia predložky v akuzatíve, závislej na slovese	*odpovídat*
s73,	Existencia predložky k v datíve, závislej na slovese	*přihlížet*
s85	Existencia predložky na v akuzatíve, závislej na slovese	*odpovídat*
s127	Existencia predložky za v akuzatíve, závislej na slovese	*odpovídat*

Tabuľka22: Význam najčastejšie použitých atribútov

Zaujímavé sú aj atribúty, ktoré sa používajú pri rozhodovaní pomocou stromov. V prípade slova *přihlížet* je možné za najdôležitejší považovať atribút m65. V niektorých prípadoch pozostával rozhodovací strom iba z tohto jediného atribútu, pričom výsledná presnosť pri rozhodovaní pomocou tohto stromu bola pomerne vysoká. „Dobrými“ atribútmi v tomto prípade sú aj m8, m61, n236 a s1.

Pri slove *přihlížet* možno za veľmi dôležitý považovať atribút a11. Tento atribút bol použitý pri stavbe viacerých rozhodovacích stromoch a bol použitý v rozhodovacích stromoch pomocou ktorých bola dosiahnutá najvyššia presnosť, v tomto prípade to bol dokonca koreň stromu. Okrem tohto atribútu sú kvalitné aj atribúty a10, m48, s22 a s127

Dôležitými atribútmi pre obe slová sú teda morfologické kategórie slova, ktoré nasleduje po slovese. Dôležitým atribútom pri výbere významu je predložka, ktorá závisí na danom slovese. Danú predložku je totiž často možné použiť iba pri jednom význame slova. Predložky, ktoré sa viažu s daným významom slova je možné nájsť vo valenčnom lexikóne, vo valenčnom rámci daného významu. Pri výbere atribútov je preto vhodné vyhľadať vo valenčnom slovníku predložky, ktoré sa viažu s daným významom a pridať atribúty, ktoré zodpovedajú týmto predložkám, závislých na slovese. Podobne je možné spracovať reflexívne se/si, ktoré sú závislé na slovese. V prípade slova *odpovídat* je veľmi dôležitá aj informácia o tom, koľko životných a neživotných substantív sa nachádza vo vete. V prípade týchto atribútov môže ísť o náhodu spôsobenú menším počtom trénovacích prípadov.
Správny výber atribútov môže nielen zvýšiť úspešnosť ale aj značne zmenšiť výpočtovú náročnosť. Počet vybraných atribútov býva totiž rádovo menší ako počet všetkých atribútov a často dosahuje rovnaké, prípadne lepšie výsledky.

Vo väčšine prípadov sú výsledky dosiahnuté pre slovo *přihlížet* lepšie ako pre slovo *odpovídat*, pričom počty trénovacích a testovacích dát u obidvoch slov sú takmer rovnaké. Vyššia presnosť pre slovo *přihlížet* je daná tým, že pri tomto slove stačí rozhodovať medzi dvoma kategóriami, pri slove *odpovídat* sú to tri kategórie. Pre sloveso *přihlížet* sú však obe kategórie podobne rozdelené, u slova *odpovídat* sa jedna kategória vyskytuje veľmi často a jedna kategória veľmi málo.

Záver
Cieľom práce bolo automaticky určiť význam slov *přihlížet* a *odpovídat* vo vete. Použité na to boli metódy strojového učenia, pričom práca bola implementovaná v jazyku R. Dôležitou časťou práce bol výber atribútov. Boli použité tri funkcie na automatický výber kvalitných atribútov: *Boruta*, *cfs* a *consistency*. Pri klasifikácii boli potom použité tri algoritmy: Naive Bayes, SVM a rozhodovacie stromy. Výsledky jednotlivých algoritmov boli nakoniec porovnané pomocou bootstrappingu a Welchovho testu.
Najlepšia dosiahnutá presnosť pri rozhodovaní medzi dvoma významami slova *přihlížet* je takmer 97 percent. Táto presnosť bola dosiahnutá pri použití algoritmu SVM, s polynomiálnou jadrovou funkciu so stupňom 3 a s hodnotami parametru *cost* 17 až 30. Rovnaký výsledok bol dosiahnutý pri použití algoritmu SVM s polynomiálnou funkciou so stupňom polynómu 4 a s hodnotami cost 29 a 30. Pri trénovaní a testovaní algoritmu boli v oboch prípadoch pritom použité všetky atribúty, ktoré bolo možné použiť.
Pri rozhodovaní medzi troma významami slova *odpovídat* bola najlepšia dosiahnutá presnosť takmer 94 percent. Táto presnosť bola dosiahnutá pomocou rozhodovacích stromov, pre atribúty získané pomocou funkcie *Boruta* pri parametroch *minsplit* = 6, *cp* = 0.0001, 0.001, 0.01 a pre atribúty vybrané pomocou funkcie *cfs* a pre parametre *minsplit* = 1, 6 a *cp* = 0.0001, 0.001, 0.01.
Jedna chyba pritom zodpovedá v oboch prípadoch približne trom percentám.

Na základe výsledkov dosiahnutých pomocou bootstrappingu je možné pre slovo *přihlížet* považovať algoritmy rozhodovacie stromy a SVM za „rovnako dobré“. Pre slovo *odpovídat* je možné pokladať algoritmy rozhodovacie stromy a Naive Bayes za „rovnako dobré“. Konfidenčné intervaly pri takto získaných hodnotách sú však príliš široké, čo môže byť spôsobené malým počtom dát. Pri porovnaní najlepších hodnôt získaných pomocou algoritmov prostredníctvom Welchovho testu nie sú žiadne dva algoritmy signifikantne podobné.

Výsledky sú veľmi citlivé na výber atribútov a na nastavenia parametrov. Aby sme dosiahli dobrú

úspešnosť pri úlohe disambiguácie slovies, bolo by vhodné ladiť parametre a výber atribútov pre každé sloveso zvlášť. Vhodnými kandidátmi na kvalitné atribúty, ktoré môžu byť často použité pri disambiguácii slovies, sú morfologické kategórie následujúceho slova a predložka, ktorá závisí na danom slovese.
Dobrý výber atribútov môže podstatne znížiť výpočtovú náročnosť a zlepšiť výsledky. V prípade rozhodovacích stromov môže byť pri slove *přihlížet* z vyše tisícky atribútov vybraný jediný atribút, ktorý je použitý pri stavbe stromu, pričom presnosť je v prípade použitia všetkých atribútov a jediného atribútu rovnaká. Tento jediný atribút bol vybraný pomocou funkcie *consistency* z knižnice *Fselector*. V prípade slova *odpovídat* je možné z tisícky atribútov vybrať 5 atribútov, ktoré sa použijú pri stavbe stromu, pričom sa presnosť oproti použitiu všetkých atribútov zvýši takmer o 15 percent. Tieto atribúty boli vybraté pomocou algoritmu *Boruta* z knižnice *Boruta*. Podobne je možné pri algoritme SVM pre slovo *odpovídat* použiť namiesto vyše 300 atribútov 9 atribútov, vybratých pomocou algoritmu *Boruta***,** pričom sa presnosť zvýši o 12 percent. Pri Bayesovom klasifikátore sú naopak najlepšie výsledky dosiahnuté pri použití všetkých atribútov a vyhladzovania.

Problém disambiguácie významu slovesa je teda možné riešiť automaticky s pomerne vysokou úspešnosťou. V praxi však môže byť problémom extrakcia správnych hodnôt atribútov z danej vety. Kvalitné výsledky je však možné dosiahnuť aj pri použití malého počtu správne vybratých atribútov. Preto, ak vieme, ktoré atribúty sú pre určenie významu daného slovesa kvalitné, je potrebné z vety extrahovať iba malú množinu hodnôt. Väčšia množina však môže viesť ku robustnejšiemu rozhodovaniu.

Literatúra

[1] Alpaydin E.: *Introduction to Machine Learning*, Massachusetts Institute of Technology, 2004.
[2] *Boruta* http://cran.r-project.org/web/packages/Boruta/Boruta.pdf
[3] Breiman L.: *Random Forests*. In: Machine Learning 45 (1), 5–32, 2001.
[4] Dixon P. M.: *Bootstrap Resampling*
http://www.wiley.com/legacy/wileychi/eoenv/pdf/Vab028-.pdf
[5] *FSelector* http://cran.r-project.org/web/packages/FSelector/FSelector.pdf
[6] *Naive Bayes classifier* http://en.wikipedia.org/wiki/Naive_Bayes_classifier
[7] *Standard error*
http://en.wikipedia.org/wiki/Standard_error_(statistics)
[8] *Support Vector Machine* http://en.wikipedia.org/wiki/Support_vector_machine
[9] VALLEX 2.5 http://ufal.mff.cuni.cz/vallex/2.5/data/html/generated/functors/index-156.html
[10] Venables W. N., Ripley B. D.: *Modern applied statistics with S Fourth Edition*, Springer, 2002.
[11] Vidová Hladká B.: *Decision trees learning*
http://ufal.mff.cuni.cz/~hladka/ML/LECTURES/jsmath/test/decision-trees.html
[12] Vidová Hladká B.: *Support Vector Machines*
http://ufal.mff.cuni.cz/~hladka/ML/LECTURES/jsmath/test/support-vector-machines.html
[13] *Welch's t test*
http://en.wikipedia.org/wiki/Welsch's_t_test