Disambiguácia významu slovies

Peter Fabian

13. máj 2010

Kapitola 1

Úvod

Prirodzené jazyky obsahujú veľké množstvo nejednoznačnosti, jednou z nich je i nejednoznačnosť významu slov. Ľudia si často túto nejednoznačnosť neuvedomujú (najmä kvôli kontextu, ktorý často napovie správny význam), no pri spracovaní prirodzeného jazyka pomocou počítačov je potrebné poznať presný význam každého slova. Tento problém sa nazýva disambiguácia významu slova (word-sense disambiguation). Prístupy k riešeniu tohto problému sa rôznia podľa spôsobu reprezentácie významu slov, (ne)obmedzenosti domény spracovávaných slov alebo sústredení sa iba na istú podmnožinu slov, ako je to v našom konkrétnom prípade, keď sme sa obmedzili na riešenie problém výberu vhodného *významu slovies*. Slovesá sú centrálne prvky viet s vplyvom na význam celej vety, preto je ich disambiguácia dôležitá a môže významne pomôcť pri určovaní významu celej vety, pre príklad viď tabuľka 1.1.

Cz	**Eng**
Petr **dal** Janě knihu.	Peter **gave** Jane a book.
Petr si **dal** klíče do kapsy.	Peter **put** his keys in his pocket.
Petr si **dal** Guiness do půllitru.	Peter **ordered** a pint of Guiness.

Tabuľka 1.1: Príklad nejednoznačnosti českého slovesa dát

Ďalšou otázkou, ktorú si musíme na začiatku zodpovedať je, nakoľko do hĺbky chceme rozlišovať význam slov, aká granularita nás zaujíma. Ako príklad nám môže poslúžiť prípad uvedený v [5]:

i Pokrájala zeleninu šéfkuchárovým *nožom*.

ii Muž bol zbitý a dobodaný *nožom*.

V prípade (i) ide o význam noža ako nástroja, v prípade (ii) ide o význam zbrane, hoci obe vety sa môžu vzťahovať k tomu istému fyzickému nožu. Vidíme, že nie je ľahké oddeliť význam slov do tried, ktoré by sa neprekrývali. Jednotlivé významy spolu súvisia a ovplyvňujú sa.

Náš experiment, ako sme uviedli vyššie, sa zaoberá disambiguáciou významu slovies. Ak by sme chceli byť precízni, mali by sme poznamenať, že nás zaujíma

nie konkrétny význam, ale valenčný rámec slovesa[1].

Formálne pod disambiguáciou významu slovies rozumieme schopnosť pomocou počítačového systému rozoznať, ktorý význam slovesa je aktivovaný v danom kontexte. Na túto úlohu sa môžeme dívať ako na klasifikačný problém, pri ktorom každému slovesu priradíme jeden (prípadne viac) význam z množiny jeho možných významov (tried). Jedná sa teda o samostatnú klasifikačnú úlohu pre každé spracovávané sloveso [5].

Vyriešenie problému určenia správneho významu slov vo vete môže významne pomôcť pri automatickom spracovávaní veľkých objemov textu a úlohách, ktoré sú na tomto princípe založené, napr. automatický preklad, sémantizácia webu, atď. Dosiaľ sa disambiguácia slov nevyužíva veľmi často v komplexnejších systémoch, najmä pre jej nie veľmi vysokú úspešnosť na otvorenej doméne slov, no spoľahlivý systém rozpoznávania významu by mohol signifikantne pomôcť pri riešení problém získavania informácií (information retrieval & extraction), strojovom preklade, obsahových analýzach textov, v lexikografii atď [5].

Veľkou prekážkou pri riešení disambiguácie významu slovies, ostatne ako i pri iných úlohách strojového učenia s dohľadom, je fakt, že je silne závislá na vstupných vedomostiach a znalostiach, často i na ručnej anotácii a rozšírenie pre každý jazyk či novú doménu znamená ďalšiu prácu ľudských anotátorov. Tento problém sa nazýva *úzke hrdlo získavania vstupných znalostí* (knowledge acquisition bottleneck) [5]. Je možné využiť existujúce zdroje ako napr. WordNet, slovníky, no i rôzne korpusy (najmä sémanticky anotované) a informácie o kolokáciách slov[2]. Najmä WordNet obsahuje mnoho informácií o významoch slov, homonymii, synonymii, hyper/hyponymii a sémantické siete slov.

Problém pri získavaní vstupných dát sa prejavil i na našej úlohe, keď sme dostali k dispozícii informácie iba o dvoch slovesách a pre každé z nich necelých 100 inštancií.

V nasledujúcej kapitole sa nachádza analýza vstupných dát, ktoré máme k dispozícii. Kapitola 3 obsahuje teoretický popis použitých metód strojového učenia. V kapitole 4 stručne opíšeme metriky použité pri evaluácii výsledkov. V Piatej kapitole nájdeme analýzu výsledkov jednotlivých metód. Zhrnutie sa nachádza napokon v šiestej kapitole.

[1] Pod pojmom valencia slovesa sa rozumie schopnosť slovesa viazať na seba určitý počet iných, syntakticky závislých jazykových jednotiek

[2] spoločný výskyt slov vo vetách a frázach

Kapitola 2

Dáta

Dáta, ktoré máme k dispozícii, obsahujú informácie o dvoch slovesách – *přihlížet* a *odpovídat*. Sú rozdelené na testovacie a trénovacie v pomere 33 ku 66 inštanciám pre prvé, resp 32 ku 65 inštanciám pre druhé menované sloveso. Spolu teda máme k dispozícii 99, resp. 97 inštancií. Prvé sloveso v našich dátach nadobúda dva, druhé tri významy. Rozdelenie na testovacie a trénovacie dáta je rovnomerné, viď tabuľka 2.1.

přihlížet			
význam	train	test	spolu
1	36	18	54
2	30	15	45
spolu	66	33	99

odpovídat			
význam	train	test	spolu
1	21	10	31
2	4	2	6
3	40	20	60
spolu	65	32	97

Tabuľka 2.1: Rozdelenie vstupných dát pre jednotlivé slovesá

V prípade prvého slovesa sa v trénovacích i testovacích dátach nachádza rovnaký pomer inštancií s jedným i druhým významom (pomer 1,2 ku 1 v prospech prvého významu); pre druhé sloveso je pomer tiež veľmi podobný, pre jednotlivé významy je v trénovacích dátach rozdelenie 21:4:40, v testovacích potom 5:1:10.

2.1 Rysy

Príprava jednotlivých rysov nebola našou úlohou, dáta sme dostali priamo pripravené ako tabuľku rysov s cieľovým atribútom, čo zodpovedá úlohe strojového učenia s dohľadom (supervised learning). Pôvodný kontext daných slovies (ich výskyt v texte, či samotný text, z ktorého boli rysy vytvárané) teda nepoznáme, jeho reprezentáciou sú teda nasledujúce rysy:

- morfologické informácie o najbližších slovách slovesa – súbory m a n,
- informácie o výskyte vybraných slov/slovných druhov/slov v daných pádoch v závislosti na skúmanom slovese – súbor s
- informácie o výskyte vybraných idiómov vo vete so slovesom – súbor i

- informácie o počte a výskyte životných/neživotných substantív vo vete/v závislosti na danom slovese – súbor a
- sémantické informácie z WordNetu o slovách vo vete a slovách závislých na slovese – súbor w

2.1.1 Súbor m

Súbor m obsahuje 75 kategoriálnych morfologických rysov, v ktorých sú zakódované informácie o dvoch slovách pred slovesom, danom slovese, a dvoch slovách za príslušným slovesom (v poradí podľa povrchovej štruktúry vety). Pre každé z týchto 5 slov máme k dispozícii nasledujúce rysy:

- slovný druh,
- detailnejšie určený slovný druh,
- rod,
- číslo,
- pád,
- vlastníkov rod,
- vlastníkovo číslo,
- osoba,
- čas,
- stupeň komparácie,
- negatívnosť,
- slovesný rod

Tieto rysy pochádzajú z pozičného tagu PDT [2] a sú, samozrejme, prítomné len ak to pre dané slovo dáva zmysel, teda napríklad pre substantíva nebude vyplnený čas či osoba, pre slovesá pád a pod.

2.1.2 Súbor n

Súbor n obsahuje tie isté informácie ako súbor m, avšak v bool-inizovanej forme: napríklad ak boli možnosti pre pád slova 1,2,3,4,5,6,7 a X, tak v súbore n budeme mať 8 rysov slovo_pád_1, slovo_pád_2 až slovo_pád_X, každý s hodnotou true alebo false. Tento súbor teda obsahuje 705 rysov kategoriálnej povahy.

Môže sa zdať, že tieto rysy sú nadbytočné, keďže obsahujú totožné informácie ako súbor m, no v niektorých metódach strojového učenia sa práve tieto môžu kvôli menšej doméne ukázať ako vhodnejšie než atribúty zo súboru m.

2.1.3 Súbor s

Súbor s obsahuje informácie o slovách syntakticky závislých na skúmanom slovese. Dva predchádzajúce súbory rysov (m a n) sa týkali roviny morfologickej, tento súbor prechádza do roviny syntaktickej. Všetkých 131 rysov je kategoriálnej povahy (konkrétne ide o boolovské rysy).

Príkladmi rysov z tohto súboru sú:

- je aktuálne sloveso závislé na inom slovese?
- existuje sloveso v infinitíve závislé na aktuálnom slovese?
- existuje substantívum alebo substantívne zámeno v akuzatíve závislé na danom slovese? (podobne pre všetky pády)
- existuje nejaká predložka (resp. adjektívum) v nominatíve závislé na danom slovese? (pre všetky pády)
- existuje podraďovacia spojka "že" závislá na danom slovese? (pre vybraných 37 spojok)
- existuje predložka "oproti" v datíve závislá na danom slovese? (pre vybrané predložky a pády – 69 rysov)

2.1.4 Súbor i

Súbor i obsahuje 118 boolovských rysov, ktoré poskytujú informácie o idiómoch nachádzajúcich sa vo vete so skúmaným slovesom.

Príkladmi rysov z tohto súboru sú:

- existuje vo vete so slovesom idióm "jako v bavlnce"?
- existuje vo vete so slovesom idióm "kolem krku"?
- existuje vo vete so slovesom idióm "při životě"?

2.1.5 Súbor a

Súbor a obsahuje spolu 18 rysov, ktoré poskytujú informácie o životných a neživotných substantívach vyskytujúcich sa v danej vete/závislých na slovese. Štyri rysy sú spojitej povahy – počty životných a neživotných substantív vo vete s daným slovesom, resp. závislých na danom slovese. Ostatných 14 boolovských rysov má nasledovnú povahu:

- existuje vo vete životné substantívum v nominatíve? (podobne pre 7 základných pádov)
- existuje životné substantívum v nominatíve závislé na slovese? (pre všetkých 7 pádov)

2.1.6 Súbor w

Súbor w obsahuje spolu 128 boolovských rysov, ktoré reprezentujú sémantické informácie o substantívach vo vete s daným slovesom (resp. syntakticky závislých na danom slovese). Informácie o sémantických triedach substantív pochádzajú z WordNet-u, v našom prípade máme informácie o 64 triedach pre substantíva v rovnakej vete a (tých istých) 64 triedach pre substantíva závislé na slovese.

Príkladmi rysov z tohto súboru sú:

- existuje vo vete substantívum s významom reprezentácie peňazí?
- existuje substantívum s významom reprezentácie peňazí syntakticky závislé na slovese?
- existuje vo vete substantívum s významom dopravného prostriedku?
- existuje substantívum s významom dopravného prostriedku syntakticky závislé na slovese?
- existuje vo vete substantívum zo sémantickej triedy času?

Pre každú inštanciu máme teda k dispozícii 1175 atribútov a 1 atribút je cieľový – označuje valenčný rámec daného slovesa. Vzťah medzi významom slovesa a jeho valenčným rámcom nie je vzájomne jednoznačný. Jeden valenčný rámec korešpondovať s viacerými významami a, naopak, jeden význam môže byť vyjadrený pomocou viacerých valenčných rámcov, pre bližší popis viď [8].

Jedným z problémov tejto úlohy je veľký počet atribútov pre každú inštanciu (1175), pri malom počte dostupných inštancií (< 100), naše dáta sú teda veľmi riedke. Ak by sme chceli jeden príklad pre každú unikátnu kombináciu rysov, potrebovali by sme približne $2^{1175} \approx 5.1 \times 10^{353}$ príkladov. Tento problém sa vyskytuje pri všetkých úlohách, kde sa používajú dáta s veľkým počtom dimenzií a nazýva sa *prekliatie vysokej dimenzie* (curse of dimensionality).

Kapitola 3

Teoretický popis použitých metód

Po analýze vstupných dát, s prihliadnutím na prevažne kategoriálnu povahu rysov a predchádzajúce výskumy ohľadom úspešnosti metód strojového učenia sme sa rozhodli použiť nasledujúce metódy:

- rozhodovacie stromy
- naivný bayesovský klasifikátor
- support vector machines

Ich bližší teoretický popis nasleduje v ďalších častiach tejto kapitoly, výsledky ich použitia sa nachádzajú v piatej kapitole.

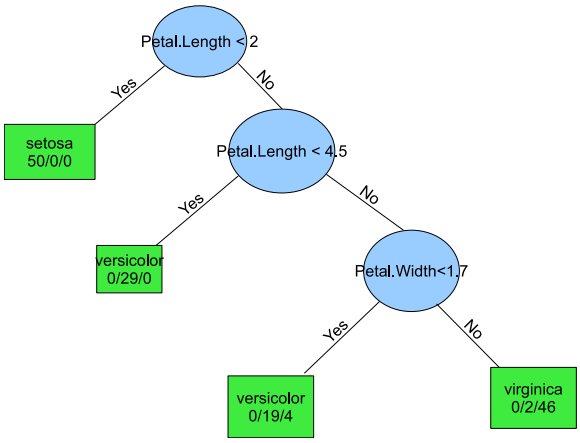
3.1 Slepý väčšinový klasifikátor

Slepým klasifikátorom označujeme základný väčšinový klasifikátor pre dané dáta. Je založený na jednoduchom princípe – z trénovacích dát vyberieme najčastejšie sa vyskytujúcu hodnotu cieľového atribútu a ňou ohodnotíme všetky inštancie v testovacích (prípadne evaluačných či reálnych) dátach. Tento jednoduchý klasifikátor nám dáva základný odhad (baseline), kedy výsledky zložitejších klasifikátorov prinášajú nejakú pridanú hodnotu vo forme kvalitnejšej predikcie.

3.2 Rozhodovacie stromy

Rozhodovací strom je súvislý acyklický graf s koreňom zložený z vnútorných uzlov a koncových listov. Každý vnútorný uzol predstavuje podmienku (alebo funkciu), ktorá delí dáta na konečný počet disjunktných skupín. Takýmto spôsobom sa trénovacie dáta rekurzívne delia. Čím hlbšie[1] v strome postupujeme, tým sú dáta rozdelené na menšie skupiny. Rozhodovací strom tak reprezentuje disjunkciu konjunkcií podmienok kladených na atribúty inštancie. Klasifikácia na

[1] Hĺbkou v strome myslíme počet vrcholov na ceste od koreňa k aktuálnemu vrcholu



Obrázok 3.1: Jednoduchý rozhodovací strom

existujúcom strome je cesta od koreňa k niektorému z listov určená hodnotou atribútov inštancie. [5]

Pri stavaní rozhodovacieho stromu je potrebné vybrať, ktoré z atribútov budú tvoriť podmienky v uzloch. Pre túto úlohu bolo vyvinutých niekoľko metód, ktoré sa snažia nájsť odpoveď na otázku, ktorý atribút najlepšie klasifikuje trénovacie dáta (prípadne ich príslušnú časť pri väčšom zanorení). Dve najznámejšie metriky sú *informačný zisk* a *Giniho index*.

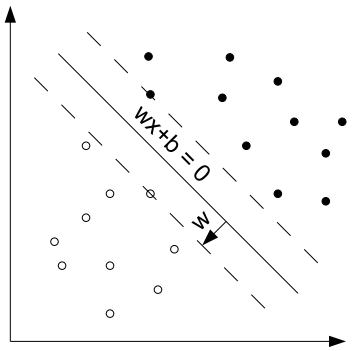
Medzi časté problémy, ktoré sú spojené s rozhodovacími stromami, patrí pretrénovanie (overfitting). Táto situácia nastáva, keď sa rozhodovací strom príliš presne naučí podmienky vyhovujúce trénovacím dátam – na trénovacích dátach sa jeho presnosť zvšuje, no na testovacích znižuje – a iný, menší rozhodovací strom má lepšie výsledky na testovacích dátach (a horšie na trénovacích). Proti pretrénovaniu sú používané orezávacie (pruning) metódy, ktoré sa líšia tým, kedy sú aplikované, na prepruning a postpruning. Prvý variant sa snaží zastaviť rast stromu počas jeho tvorenia, druhý sa aplikuje na strom po jeho vytvorení.

Ďalším problémom pre našu úlohu je fakt, že rozhodovacie stromy sú náchylné na chyby, ak máme k dispozícii malý počet príkladov s mnohými atribútmi [5]. Bohužiaľ však nemáme možnosť otestovať chybovosť na väčšom objeme dát.

3.3 Support Vector Machines

Metóda support vector machines (SVM) [5] je založená na myšlienke skonštruovania lineárnej nadroviny, ktorá čo najlepšie oddeľuje kladné a záporné inštancie problému umiestnené v hyperpriestore atribútov. Nadrovina je vybraná tak, aby maximalizovala vzdialenosť k najbližším pozitívnym a negatívnym bodom v hyperpriestore (ktoré nazývame support vectors) a zároveň minimalizovala chybu klasifikácie. Všeobecná rovnica nadroviny je $\mathbf{wx} + b = 0$, pre novú inštanciu $\mathbf{x}$ stačí porovnať $\mathbf{wx}$ s 0, ak $\mathbf{wx} > 0$, inštancia patrí do jednej triedy, inak do druhej.

Ak nie je možné oddeliť body pomocou lineárnej nadroviny, metóda svm môže byť rozšírená o uvoľňujúce premenné (slack variables), ktoré dovolia jed-



Obrázok 3.2: Geometrické znázornenie svm

notlivým bodom nachádzať sa neďaleko od roviny na opačnej strane ako ostatné body, ktoré patria do rovnakej triedy. Inou možnosťou je použiť transformačnú funkciu nazývanú jadrová (kernel function) a priestor transformovať na taký, kde sa lineárna nadrovina nájsť dá. Forma jadrovej funkcie je $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$, kde ϕ je transformačná funkcia a k je kernel funkcia.

V prípade, že cieľový atribút nadobúda viac než dve hodnoty, jedná sa o tzv. mnohotriedny svm (multiclass svm). V tomto prípade je možné urobiť klasifikáciu dvoma základnými spôsobmi:

- jeden-vs-ostatní – vezme sa jedna trieda a všetky ostatné sa stanú súčasťou druhej "metatriedy"; výsledná klasifikácia inštancie je určená najvyššou výstupnou funkciou
- jeden-vs-jeden – natrénuje sa svm pre každú dvojicu tried; výsledná klasifikácia inštancie je výsledkom hlasovania všetkých klasifikátorov

Podľa [5] je metóda svm často najúspešnejšou metódou pri riešení úlohy rozpoznávania významu slov, preto môže byť zaujímavé porovnať korešpondenciu medzi nami dosiahnutými výsledkami a predchádzajúcim výskumom.

Problémom pri riešení úlohy pomocou svm môže byť príliš veľké množstvo atribútov dostupných pre každú inštanciu, i keď podľa [6] v istých prípadoch žiadne filtrovanie potrebné nie je a svm si poradí i s 5000 atribútmi, iní autori ([11]) hovoria o výraznom zlepšení pri odstránení nerelevantných atribútov.

3.4 Naivný bayesovský klasifikátor

Bayesovské učenie, ktorého je naivný bayesovský klasifikátor (NBK) [5] špeciálnym prípadom, prináša k hodnoteniu hypotéz pravdepodobnostný prístup. Pri nachádzaní najpravdepodobnejšej hypotézy pre dané trénovacie dáta D využíva Bayesov princíp pre podmienenú pravdepodobnosť:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

kde $P(A|B)$ je je pravdepodobnosť javu A, ak nastal jav B.

Pri aplikovaní na prípad strojového učenia, kde $P(D|h)$ je pravdepodobnosť výskytu trénovacích dát D vo svete, kde platí hypotéza h a $P(h|D)$ je pravdepodobnosť platnosti hypotézy h pri trénovacích dátach D dostávame

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)},$$

keďže nás zaujíma najpravdepodobnejšia hypotéza, hľadáme

$$h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} \frac{P(D|h)P(h)}{P(D)} = argmax_{h \in H} P(D|h)P(h),$$

posledná rovnosť vyplýva z toho, že pri zmene hypotéz je $P(D)$ konštantná, a teda neovplyvní výsledok funkcie argmax.

Pre NBK pridáme kvôli zjednodušeniu ešte ďalšie podmienky:

- každá inštancia je tvorená konjunkciou n hodnôt atribútov $(a_1, a_2, ..., a_n)$
- hodnoty jednotlivých atribútov sú podmienene nezávislé na hodnote cieľového atribútu y

Dostávame teda

$$y_{MAP} = argmax_{y_j \in Y} P(a_1, a_2, ..., a_n|y_j)P(y_j) = argmax_{y_j \in Y} P(y_j) \prod_{i=1}^{n} P(a_i|y_j),$$

kde posledná rovnosť plynie z podmienenej nezávislosti. Pravdepodobnosti $P(y_j)$ a $P(a_i|y_j)$ sú vypočítané z trénovacích dát ako relatívna frekvencia výskytu cieľovej hodnoty y_j, resp frekvencia výskytu hodnoty atribútu a_i pri cieľovej hodnote y_j. V našej konkrétnej úlohe ide o relatívny výskyt konkrétneho významu slovesa v trénovacích dátach a relatívny výskyt danej hodnoty atribútu pri nejakom význame slovesa.

Podobne ako pri iných pravdepodobnostných prístupoch, i pri NBK nastáva problém nevidených dát, kde sú počty výskytu konkrétnych hodnôt rovné 0. Tento problém sa rieši pomocou rôznych metód vyhladzovania (smoothing).

V našom konkrétnom prípade však pre využiteľnosť metódy NBK vidím ako hlavný problém vybrať z veľkého množstva atribútov vhodnú kombináciu, pri ktorej by mal NBK dobré výsledky.

Kapitola 4

Metóda vyhodnotenia výsledkov

Pri vyhodnocovaní úspešnosti metód strojového učenia sa používajú štandardné metriky – *presnosť* (precision), *úspešnosť* (recall) a ich kombinácia – *F-measure*. Tieto miery vychádzajú z hodnotenia pri úlohe vyhľadávania a extrakcie informácií [4] a preto sú pôvodne binárne. Pri vyhľadávaní informácií máme k dispozícii kolekciu dokumentov, v ktorej vyhľadávame dokumenty, ktoré sú pre nás relevantné. To, aká je automatická metóda úspešná pri vyhľadávaní dokumentov, môžeme merať pomocou viacerých ukazovateľov. Budeme vychádzať z tzv. *confusion matrix*[1], viď tabuľka 4.1.

	skutočnosť	
nájdené	relevantné	irelevantné
relevantné	tp	fp
irelevantné	fn	tn

Tabuľka 4.1: Confusion matrix

Jednotlivé bunky tabuľky označujú

- tp – true positive – počet relevantných dokumentov, ktoré boli správne označené ako relevantné
- tn – true negative – počet irelevantných dokumentov, ktoré boli správne označené ako irelevantné
- fp – false positive – počet irelevantných dokumentov, ktoré boli *ne*správne označené ako relevantné
- fn – false negative – počet relevantných dokumentov, ktoré boli *ne*správne označené ako irelevantné

a potom

[1] v tomto prípade používame anglický termín, pretože nepoznáme vhodný slovenský ekvivalent, jeho použitie by mohlo byť rovnako zmätočné ako daná matica :)

$$P = \frac{tp}{tp + fp}; R = \frac{tp}{tp + fn};$$

$$F - measure = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)\frac{1}{R}},$$

kde *precision*(P) vyjadruje podiel skutočne relevantných dokumentov z tých, ktoré boli nájdené a *recall*(R) vyjadruje, akú časť z relevantných dokumentov sa nám podarilo nájsť. *F-measure* kombinuje tieto dve metriky do jednej hodnoty a umožňuje pomocou váhového koeficientu α dať pri vyhodnocovaní väčší dôraz na jednu zo spomínaných metrík.

Ďalšou často používanou mierou je *accuracy*, ktorá jednoducho určuje percentuálny podiel správnych klasifikácií

$$A = \frac{tp + tn}{tp + fp + fn + tn},$$

táto miera je však málo citlivá na tp, fp a fn, keďže číslo tn, teda počet nevybraných nerelevantných dokumentov je často omnoho väčšie, než ostatné ukazovatele. Vysokú hodnotu *accuracy* je teda jednoducho možné dosiahnuť tým, že nevyberieme nič, a preto ju v tejto práci nebudeme používať.

V našom prípade, pri hľadaní valenčných rámcov slovies, však máme viac než dve kategórie, preto sme sa rozhodli použiť na vyhodnotenie vyššie spomínanú metriku *F-measure* adaptovanú na problém s k triedami. Podľa [9] existujú dva spôsoby vyhodnotenia, F-measure s *makro-* a *mikro*-priemerovaním. *Makropriemer* dáva rovnaký dôraz na každú triedu, bez ohľadu na to, koľko inštancií do nej patrí, zatiaľ čo *mikropriemer* dáva rovnaký dôraz na každú inštanciu, a teda favorizuje väčšie triedy. V našej práci budeme používať obe tieto metriky. Vzťahy pre vypočítanie makropriemeru (F_M) a mikropriemeru (F_μ) pre k tried sú nasledovné:

$$P_M = \frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i + fp_i}}{k}; R_M = \frac{\sum_{i=1}^{k} \frac{tp_i}{tp_i + fn_i}}{k};$$

$$F_M - measure = \frac{1}{\alpha \frac{1}{P_M} + (1 - \alpha)\frac{1}{R_M}}.$$

$$P_\mu = \sum_{i=1}^{k} \frac{tp_i}{tp_i + fp_i}; R_\mu = \sum_{i=1}^{k} \frac{tp_i}{tp_i + fn_i};$$

$$F_\mu - measure = \frac{1}{\alpha \frac{1}{P_\mu} + (1 - \alpha)\frac{1}{R_\mu}}.$$

4.1 Rozlišovacia schopnosť

Čitateľ by si však mal uvedomiť, že kvôli malej vzorke dát znamená jedna chyba v testovacích dátach pre sloveso *přihlížet* rozdiel približne 3% pre F_μ i F_M, v testovacích dátach je to rozdiel polovičný (keďže sú trénovacie a testovacie dáta rozdelené v pomere 2:1).

Pri slovese *odpovídat* je situácia trochu zložitejšia: v testovacích dátach jedna chyba znamená rozdiel v F_μ približne 3%, rozdiel v F_M sa pohybuje od 2.5% (ak chybne priradíme slovesu z triedy 3 triedu 1) po 10% (ak chybne priradíme slovesu z triedy 2 triedu 1). Toto je spôsobené malou veľkosťou triedy 2 pri slovese *odpovídat*. Rozdiel v úspešnosti pri chybe v trénovacích dátach je kvôli rozdeleniu dát na polovičnej úrovni.

Kapitola 5

Praktické aspekty riešenia a výsledky

Pri praktickom riešení úlohy boli využité nasledujúce implementácie:

- rozhodovacie stromy – knižnica `rpart` z programu R a `C5.0` (viď [3])
- NBK – knižnica `e1071` z programu R
- SVM – knižnica `e1071` z programu R

Nasledujúce podkapitoly prinášajú výsledky zvolených metód na dátach pre slovesá *přihlížet* a *odpovídat*. Keďže pre každé z týchto slovies ide o samostatnú klasifikačnú úlohu (a i kvôli prehľadnosti), výsledky metód budeme vyhodnocovať pre každé sloveso zvlášť.

5.1 Predspracovanie vstupných dát

Vstupné dáta, bližšie popísané v kapitole 2, obsahujú pre každú inštanciu 1175 atribútov. Kvôli tomuto veľkému počtu atribútov sme sa rozhodli použiť heuristickú metódu, ako počet atribútov pre metódy NBK a SVM zmenšiť.

5.1.1 Prvotný filter

Ako prvotný filter sme zvolili vyškrtnutie atribútov obsahujúcich výlučne neznáme hodnoty, iba jednou hodnotou pre daný atribút alebo ich kombináciu. Knižnica `e1071`, z ktorej pochádza nami používaná implementácia NBK tieto hodnoty i tak nedokáže spracovať, takže týmto filtrom by sme nemali stratiť žiadnu dôležitú informáciu [1]. Z pôvodných 1175 atribútov nám po použití prvotného filtra ostalo

- 133 atribútov pre každú inštanciu slovesa *přihlížet*,
- 128 atribútov pre každú inštanciu slovesa *odpovídat*.

[1] je možné, že tento predpoklad je trochu riskantný, pretože podľa [1] môžu i zdanlivo irelevantné atribúty v kombinácii s inými poskytovať cennú informáciu, no v tomto prípade sme ho vzhľadom na veľký počet atribútov a malý počet inštancií rozhodli urobiť

Tieto sady atribútov potom tvorili základ pri ďalších pokusoch s metódami strojového učenia.

5.2 Výber atribútov

Keďže i 130 atribútov pri zhruba polovičnej veľkosti trénovacích dát je stále dosť, je potrebné z nich vybrať niekoľko atribútov, ktoré najlepšie predpovedajú valenčný rámec slovesa.

Metódy používané na výber atribútov sa delia na dve skupiny:

i wrapper metódy

ii filtre

Wrapper metódy slúžia ako akýsi „obal“ nad pôvodnými klasifikátormi – po zvolení vhodnej funkcie na porovnanie výsledkov natrénovaného modelu (zvyčajne zrejme pôjde o accuracy, precision, recall alebo ich rôzne kombinácie) sa snažia nájsť kombináciu atribútov, ktorej skóre by bolo čo najvyššie. Tieto metódy sú často časovo veľmi náročné, pretože pri veľkom počte parametrov je počet ich rôznych kombinácií obrovský. Používajú sa preto často namiesto hrubej sily (teda skúšania všetkých možností) najmä rôzne heuristiky alebo algoritmy hľadajúce aspoň lokálne maximá.

Ja som použil tzv. *forward-selection* algoritmus, ktorý vyberie prvý najlepšie hodnotený parameter, k nemu hľadá druhý tak, aby čo najväčšmi zvýšil hodnotu určenej funkcie a tak pokračuje až kým k žiadnemu zlepšeniu nedôjde.

Na podobnom princípe, no z opačného konca funguje *backward-select* algoritmus, ktorý odoberá zo všetkých atribútov ten, ktorého odobratie spôsobí najväčší nárast cieľovej funkcie. Ja som vo svojej práci používal forward-selection algoritmus. Použitie backward-selection sa ukázalo ako problematické, pretože množstvo atribútov obsahuje neznáme (NA) hodnoty, s ktorými si algoritmy strojového učenia ťažko poradia (napríklad ignorujú danú inštanciu a môže sa teda stať, že nám napokon neostane na trénovanie žiadna inštancia).

Filtre sa snažia z princípu fungovania jednotlivých algoritmov analyticky odvodiť, aké atribúty by pre daný algoritmus boli vhodné, prípadne hľadaním rôznych súvislostí v dátach nájsť vhodnú funkciu, ktorá ohodnotí závažnosť jednotlivých atribútov a ich kombinácií. Jedným z nami skúšaných filtrov bol prístup opísaný v [7], kde je využitý algoritmus C4.5 na výber vhodných atribútov pre metódu NB (v našej práci sme použili novší algoritmus C5.0). Dosiahnuté hodnoty mikropriemeru F-measure boli 90.9%, resp 90.6% (pre slovesá *přihlížet*, resp *odpovídat*) sú celkom sľubné, no s použitím wrapper metód sa nám podarilo dosiahnuť lepšie výsledky. V programe R sa nachádzajú aj ďalšie knižnice slúžiace na výber parametrov (`FSelector` alebo `dprep`) pomocou filtrových metód, tieto však v našej práci skúmané neboli.

5.3 Slepý klasifikátor (baseline)

Najjednoduchším klasifikátorom je slepý väčšinový klasifikátor, ktorý ohodnotí všetky nové inštancie najčastejšou hodnotou cieľového atribútu z trénovacích dát.

5.3.1 Přihlížet

Hodnota mikropriemeru F-masure pre testovacie dáta je na úrovni 54.5%; makropriemer nadobúda hodnotu 35.5%. Tieto základné hodnoty sa budeme snažiť pomocou metód strojového učenia prekonať. Výsledky sa nachádzajú i v súhrnnej tabuľke 6.1.

5.3.2 Odpovídat

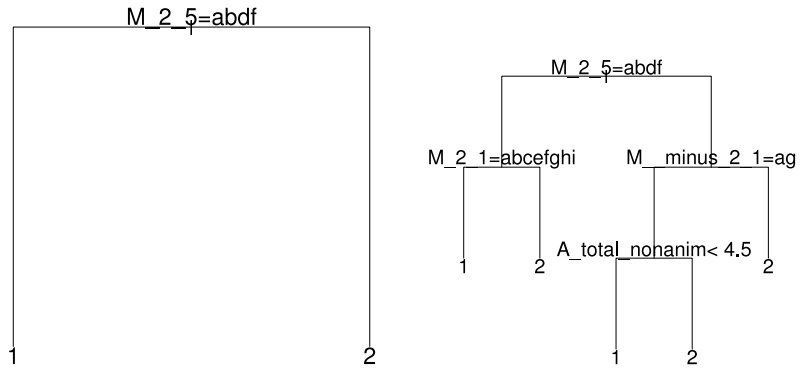
Mikropriemer F-masure pre testovacie dáta je na úrovni 62.5%; makropriemer nadobúda hodnotu 25.6%. Výsledky sa nachádzajú i v súhrnnej tabuľke 6.2.

5.4 Rozhodovacie stromy – rpart

Pri práci s **rozhodovacími stromami** nie je potrebné vyberať, ktoré atribúty použiť, z princípu rozhodovacích stromov vyplýva, že si ich algoritmus volí sám. Môžeme poznamenať, že sme skúsili spustiť metódy vytvárajúce rozhodovacie stromy i nad nefiltrovanými dátami, no ich výsledky boli totožné ako pri použití dát upravených prvotným filtrom (viď podkapitola 5.1.1). Výsledky pre jednotlivé slovesá nasledujú.

5.4.1 Přihlížet

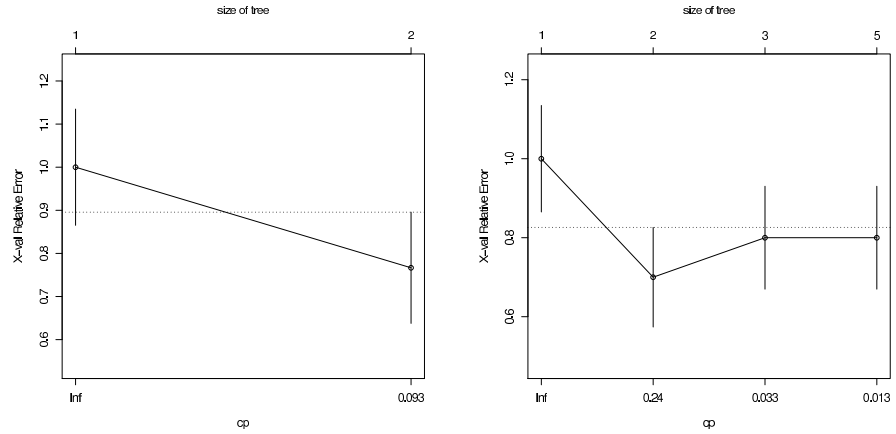
Strom vytvorený funkciou `rpart` pre sloveso *přihlížet* sa nachádza na obrázku 5.1a.



(a) Neoptimalizovaný (b) Optimalizovaný

Obrázok 5.1: rpart – stromy pre sloveso přihlížet

Algoritmus rpart si vybral ako jediný a najvýznamnejší atribút *M_2_5*, čo zodpovedá pádu slova dve pozície za slovesom *přihlížet*. Pre nominatív, genitív, akuzatív a lokál je pridelený slovesu rámec číslo 1, pre datív, vokatív, inštrumentál a neznámy pád rámec 2. Tento strom pri pohľade na vysledok funkcie `plotcp` nie je pretrénovaný (viď obrázok 5.2a).



(a) Neoptimalizovaný (b) Optimalizovaný

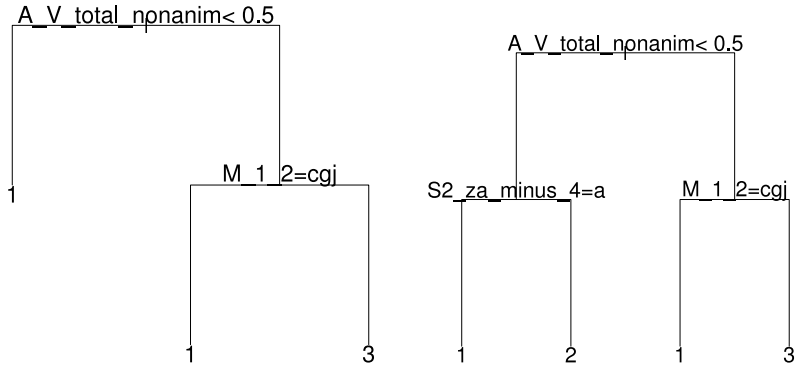
Obrázok 5.2: Plotcp pre sloveso přihlížet

Optimalizácia

Pri pokuse optimalizovať strom slovesa *přihlížet* pomocou parametrov `minspilt`, `minbucket` a `maxsurrogates` sa podarilo dosiahnuť mierne zvýšenie hodnoty makropriemeru F-measure, keď je hodnota `minbucket` = 2 (viď obrázok 5.1b). Toto zlepšenie je však iba za cenu pretrénovania stromu (viď obrázok 5.2b). Pri použití orezávacieho parametra `cp` dostávame rovnaký rozhodovací strom ako na obrázku 5.1a. Výsledky na trénovacích a testovacích dátach, ktoré sa nachádzajú v tabuľke 6.1, teda zodpovedajú základnému natrénovaniu bez použitia ďalších parametrov.

5.4.2 Odpovídat

Strom vytvorený funkciou `rpart` pre sloveso *odpovídat* sa nachádza na obrázku 5.3a.



(a) Neoptimalizovaný (b) Optimalizovaný

Obrázok 5.3: rpart – stromy pre sloveso odpovídat

Algoritmus rpart vytvoril strom s dvoma uzlami a tromi koncovými uzlami.

Vybrané paramtre v uzloch sú *A_V_total_nonanim* a *M_1_2*, ich význam je nasledujúci:

- A_V_total_nonanim – počet neživotných substantív vo vete závislých na danom slovese
- M_1_2 – detailný slovný druh slova za slovesom

Ak sa vo vete nenachádza neživotné substantívum, slovesu je pridelený rámec číslo 1. Inak sa strom pozrie na detailný slovný druh slova za slovesom odpovídat a ak je to jedna z hodnôt `^`, `A`, `b`, `D`, `g`, `N`, `S` [2], pridelí slovesu rámec 3, inak mu pridelí rámec 1.

Už aj tento základný strom je trochu pretrénovaný, ako môžeme vidieť na obrázku 5.4a. Myslím, že je to dôsledok toho, že strom nie je orezaný na menej listov, ako je počet možných výsledkov, ktoré majú byť predpovedané. V našom prípade však listy stromu klasifikujú len rámce 1 a 3, rámec 2 ostal neidentifikovateľný.

Optimalizácia

Podobne ako pri predchádzajúcom slovese, i tu sme uskutočnili optimalizáciu metódy vzhľadom na parametre `minspilt`, `minbucket` a `maxsurrogates`. Zistili sme, že najlepšie výsledky dosahuje model pre parameter `minbucket` = 4 (obr 5.3b). Toto nie je prekvapením, lebo v trénovacích dátach sa nachádzajú práve 4 inštancie s rámcom číslo 2 a teda tento parameter spôsobí, že strom bude vedieť identifikovať aj rámec s číslom 2. K vybraným atribútom pribudol jeden ďalší:

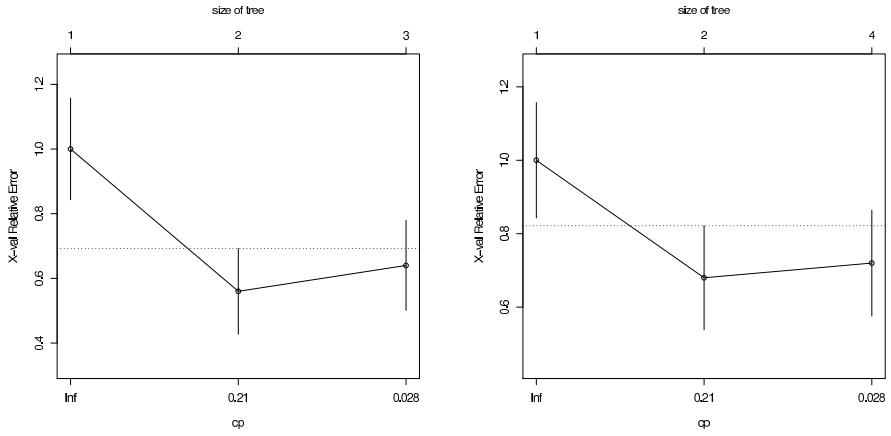
- A_V_total_nonanim – počet neživotných substantív vo vete závislých na danom slovese
- M_1_2 – detailný slovný druh slova za slovesom
- **S2_za_minus_4** – predložka „za“ v akuzatíve je závislá na slovese odpovídat

Ten rozlišuje, či sa vo vete bez neživotných substantív nachádza predložka „za“ syntakticky závislá na danom slovese.

Keďže výstup funkcie `plotcp` pre túto optimalizáciu (viď obrázok 5.4b) je takmer totožný ako pre trojlistový východiskový strom bez optimalizácie (obr. 5.4a), myslím, že je vhodnejšie použiť tento optimalizovaný strom, pretože je schopný identifikovať aj rámec číslo 2. Naše vstupné trénovacie dáta obsahujú dosť málo inštancií a optimalizácia priniesla zlepšenie nielen na trénovacích, ale i na testovacích dátach, preto si myslím, že pri použtí parametra `minbucket` s hodnotou 4, nedochádza k veľkému pretrénovaniu stromu.

V tabuľke 6.2 sú preto zapísané výsledky optimalizovaného stromu.

[2] `^` = koordinácia spájajúca hlavné klauzule, A = adjektívum, b = adverbium, ktoré nemá negáciu ani sa nedá stupňovať, D = ukazovacie zámeno, g = adverbium, ktoré je schponé tvoriť negácie a je možné ho stupňovať, N = substantívum, S = privlastňovacie zámeno



(a) Neoptimalizovaný (b) Optimalizovaný

Obrázok 5.4: Plotcp pre sloveso odpovídat

```
S2_prep_plus_3 = t: 2 (25/2)
S2_prep_plus_3 = f:
:...S2_part_se = t: 2 (3)
    S2_part_se = f: 1 (38/4)
```

Obrázok 5.5: C5.0 – strom pre sloveso přihlížet

5.5 Rozhodovacie stromy – C5.0

Algoritmus C5.0, ako sme spomínali vyššie, je vylepšením algoritmu C4.5 a ID3 na vytváranie rozhodovacích stromov. Jeho presná podoba nie je známa, je distribuovaný ako komerčný nástroj a voľne na stiahnutie je prístupná jeho demoverzia (viď [3]), ktorej jediné obmedzenie spočíva v tom, že berie do úvahy maximálne 400 inštancií, čo je pre nás dostačujúce. Vylepšenia oproti známemu algoritmu C4.5 spočívajú v rýchlejšom behu a zapojení techniky tzv. *boostingu*, čo v praxi znamená, že pre jedny dáta je vytvorených viacero stromov a pri hodnotení každej inštancie z testovacích/evaluačných dát sa rozhoduje pomocou hlasovania. Sami autori algoritmu však priznávajú ([3]), že pri väčšmi inkonzistentných dátach táto technika môže byť skôr na škodu.

5.5.1 Přihlížet

Pri slovese *přihlížet* dostávame strom s iba dvoma vrcholmi a troma listami:

- S2_prep_+3 – nejaká predložka v datíve je syntakticky závislá na danom slovese
- S2_part_se – reflexívne zámeno ,,se“ je závislé na danom slovese

Strom je znázornený na obrázku 5.5.

Optimalizácia

Keďže program, ktorý aplikuje algoritmus C5.0, neumožňuje skúmať, nakoľko je strom pretrénovaný, tieto informácie nemáme. Stromy sú však s pruningom i bez neho rovnaké.

Vyskúšali sme však vyššie popísanú možnosť použitia *boostovania*. Z našich malých dát môžeme vidieť, že boosting pomáha zlepšiť presnosť skôr na trénovacích ako testovacích dátach, ide teda skôr o príklad pretrénovania (pre konkrétne výsledky viď tabuľka 5.1). Do tabuľky výsledkov je preto zapísaný údaj bez boostingu.

Výsledky C5.0 pre sloveso přihlížet				
	train		test	
variant	F_μ	F_M	F_μ	F_M
bez boostingu	90.9%	90.9%	90.9%	90.9%
s boostingom	97.0%	96.9%	87.9%	87.9%
δ *inst*	1.5%	1.5%	3%	3%

Tabuľka 5.1: Výsledky jednotlivých metód pre sloveso přihlížet (pozn.: δ *inst* označuje, akú percentuálnu zmenu predstavuje jedna inštancia z príslušných dát, viď tiež 4.1)

5.5.2 Odpovídat

Pre sloveso *odpovídat* dostávame strom s nasledujúcimi vrcholmi (usporiadanými podľa hĺbky v strome):

- S2_za_minus_4 – predložka „za“ v akuzatíve je syntakticky závislá na slovese
- S2_prep_plus_4 – nejaká predložka v akuzatíve je závislá na danom slovese
- A_V_anym1 – životné substantívum v nominatíve je závislé na danom slovese
- A_V_anym3 – životné substantívum v datíve je závislé na danom slovese

Grafické znázornenie stromu pre toto sloveso je na obrázku 5.6.

Optimalizácia

Aj v prípade slovesa *odpovídat* je výsledný strom s vykonaným pruningom i bez neho rovnaký.

Pri experimentoch s boostovaním došlo k zlepšeniu iba na trénovacích dátach, je teda pravdepodobné, že pôjde opäť o pretrénovanie. Do výslednej tabuľky 6.2 je preto zapísaný výsledok bez boostingu. Porovnanie výsledkov s boostingom a bez neho, viď tabuľka 5.2

```
S2_za_minus_4 = t: 2 (5/1)
S2_za_minus_4 = f:
:...S2_prep_plus_4 = t: 1 (10)
    S2_prep_plus_4 = f:
    :...A_V_anym1 = t: 1 (4)
        A_V_anym1 = f:
        :...A_V_anym3 = t: 1 (3)
            A_V_anym3 = f: 3 (43/3)
```

Obrázok 5.6: C5.0 – strom pre sloveso odpovídat

Výsledky C5.0 pre sloveso odpovídat				
	train		test	
variant	F_μ	F_M	F_μ	F_M
bez boostingu	93.8%	92.3%	90.6%	85.2%
s boostingom	98.5%	95.8%	90.6%	85.2%
δ *inst*	1.5%	1-5%	3%	2-10%

Tabuľka 5.2: Výsledky jednotlivých metód pre sloveso přihlížet (pozn.: δ *inst* označuje, akú percentuálnu zmenu predstavuje jedna inštancia z príslušných dát, viď tiež 4.1)

5.6 Support Vector Machines

Pri experimentoch s algoritmom SVM sme na hľadanie najvhodnejších atribútov používali wrapper metódu forward-selection (viď podkapitola 5.2). Problémom pri použití tejto metódy môže byť fakt, že kombinácií atribútov s najlepším skóre môže nájsť mnoho. Z týchto rozličných kombinácií je vcelku ťažké odvodiť, ktoré atribúty sú skutočne dôležité. Vzali sme preto všetky kombinácie atribútov a spočítali, koľkokrát sa vyskytujú medzi najlepšími kombináciami. Z najfrekventovanejších sme potom vytvorili finálne modely.

5.6.1 Přihlížet

Pre sloveso *přihlížet* bol najlepší výsledok dosiahnutý pomocou forward-selection algoritmu: $F_\mu = 93.9\%$ a $F_M = 93.9\%$. Tento výsledok bol dosiahnutý 10 kombináciami atribútov, načastejšie sa vyskytujúce atribúty boli nasledujúce:

- S2_prep_plus_3 – 9/10 – nejaká predložka v datíve je syntakticky závislá na danom slovese
- S2_part_se – 4/10 – reflexívne zámeno „se“ je závislé na danom slovese
- A_total_nonanim – 4/10 – počet neživotných substantív vo vete
- A_total_anim – 4/10 – počet životných substantív vo vete
- A_V_total_nonanim – 2/10 – počet neživotných substantív vo vete závislých na danom slovese

Ďalších 8 atribútov sa vyskytlo v najúspešnejších kombináciách jeden raz. Z tých, ktoré sa vyskytli aspoň dva razy sme vytvorili model.

Optimalizácia

Model s danými atribútmi sme skúsili modifikovať použitím odlišných kernel funkcií (radial, polynomial druhého a tretieho stupňa, sigmoid, linear) a parametrov `gamma`, `cost` a `epsilon`. Najlepšie výsledky sme dosiahli s použitím radiálnej kernel funkcie s parametrami `gamma` = 0.5 a `cost` = 1.0. Mikropriemer na testovacích dátach bol na úrovni 97.0%, makropriemer 97.0%. Kompletné výsledky sa nachádzajú v súhrnnej tabuľke 6.1.

5.6.2 Odpovídat

Pre sloveso *odpovídat* sme pri hľadaní najlepšej kombinácie atribútov pomocou forward-selection algoritmu narazili na 31 kombinácií atribútov s najlepším skóre $F_\mu = 93.8\%$. Z týchto atribútov sme vybrali najčastejšie sa vyskytujúce:

- S2_prep_plus_4 – 31/31 – nejaká predložka v akuzatíve je syntakticky závislá na danom slovese
- A_V_anym3 – 28/31 – životné substantívum v datíve je závislé na danom slovese
- S2_za_minus_4 – 27/31 – predložka „za“ v akuzatíve je syntakticky závislá na slovese
- A_anym1 – 3/31 – životné substantívum v nominatíve sa nachádza vo vete s daným slovesom
- M_1_3__minus_ – 2/31 – slovo bezprostredne za nami skúmaným slovesom nemá rod (3. pozícia morfologického tagu je „-“)
- A_anym3 – 2/31 – životné substantívum v datíve sa nachádza vo vete s daným slovesom

Okrem vymenovaných atribútov bolo ďalších 25 atribútov použitých jedenkrát.

Optimalizácia

Experimentovali sme s použitím kernel funkcií raidal, polynomial druhého a tretieho stupňa, sigmoid a zmenou parametrov `gamma`, `cost`, `epsilon` a `coef0` pre polynomiálnu kernel funkciu. Najlepšie výsledky sme dosiahli s použitím kernel funkcie radial s parametrom `gamma` = 0.5 a polynomiálnej kernel funkcie druhého stupňa s parametrom `gamma` $\in (0.5, 5.0)$. Výsledky na trénovacích a testovacích dátach sa nachádzajú v súhrnnej tabuľke 6.2.

5.7 Naivný bayesovský klasifikátor

Po prvotných experimentoch s využitím atribútov vybraných algoritmom C5.0 sme aj pri NBK použili forward-selection algoritmus. Podobne ako pri SVM, i pri NBK našiel algoritmus viacero kombinácií atribútov s najlepším skóre. Z nich boli vybrané tie, ktoré sa medzi atribútmi vyskytovali najčastejšie.

5.7.1 Přihlížet

Pre sloveso *přihlížet* sa nám podarilo natrénovať NBK tak, že až 98 kombinácií atribútov dosiahlo na trénovacích dátach úspešnosť 100%. Najčastejšie sa vyskytujúce atribúty boli tieto:

- S2_prep_plus_3 – 85/98 – nejaká predložka v datíve je syntakticky závislá na danom slovese
- M__minus_2_5 – 85/98 – pád slova dve pozície pred skúmaným slovesom
- M_2_5 – 84/98 – pád slova dve pozície za skúmaným slovesom
- A_anym1 – 61/98 – životné substantívum v nominatíve sa nachádza vo vete s daným slovesom
- M__minus_1_2 – 14/98 – detailný slovný druh slova jednu pozíciu pred daným slovesom
- M__minus_2_4 – 10/98 – gramatická kategória ,,číslo“ slova nachádzajúceho sa dve pozície pred daným slovesom
- M_1_1 – 8/98 – slovný druh slova za daným slovesom
- M_1_4 – 7/98 – gramatická kategória ,,číslo“ slova na pozícii za daným slovesom

Okrem týchto 8 atribútov bolo v rozličných kombináciách použitých ďalších 92 atribútov.

Optimalizácia

Keďže táto metóda dosiahla na testovacích dátach mikro i makropriemer F-measure na úrovni 100%, žiadna ďalšia optimalizácia nedáva zmysel. Výsledky pre trénovacie i testovacie dáta sa nachádzajú v súhrnnej tabuľke 6.1.

5.7.2 Odpovídat

Pre sloveso *odpovídat* sme našli pomocou algoritmu forward-selection dokonca až 102 kombinácií atribútov, ktoré dosiahli najvyššie skóre. Z nich sa najčastejšie vyskytovali nasledujúce atribúty:

- S2_prep_plus_4 – 100/102 – nejaká predložka v akuzatíve je syntakticky závislá na danom slovese
- M__minus_1_2_M – 86/102 – slovo bezprostredne pred nami skúmaným slovesom je adjektívum odvodené od minulého prechodníka
- A_V_anym3 – 79/102 – životné substantívum v datíve je závislé na danom slovese
- M_2_1_R – 67/102 – slovo dve pozície za skúmaným slovesom je predložka
- M__minus_1_1 – 12/102 – slovný druh slova pred skúmaným slovesom

- S2_za_minus_4 – 11/102 – predložka „za“ v akuzatíve je syntakticky závislá na slovese
- M_1_3__minus_ – 10/102 – slovo bezprostredne za nami skúmaným slovesom nemá rod (3. pozícia morfologického tagu je „-“)

Okrem menovaných atribútov sa v rôznych kombináciách nachádzalo ešte ďalších 100 iných atribútov.

Optimalizácia

Pri pokuse o optimalizáciu NBK zmenou parametra `laplace`, teda zapojenia Laplacovho vyhladzovania, sme nedosiahli lepšie výsledky, než bez neho. Na testovacích dátach mali mikropriemer a makropriemer hodnoty 96.9% a 89.6%. Výsledky sú zapísané v súhrnnej tabuľke 6.2.

Kapitola 6

Analýza a porovnanie

Pri pohľade do súhrnných tabuliek s výsledkami vidíme, že najlepšie si so slovesom *přihlížet* poradila metóda NBK, pričom všetky použité metódy dosiahli výrazne lepšiu úspešnosť než slepý väčšinový klasifikátor.

přihlížet				
	train		test	
metóda	F_μ	F_M	F_μ	F_M
baseline	54.5%	35.3%	54.5%	35.3%
C5.0	90.9%	90.9%	90.9%	90.9%
rpart	93.9%	93.9%	93.9%	93.9%
SVM	93.9%	93.9%	97.0%	97.0%
NBK	95.5%	95.5%	100%	100%
δ *inst*	1.5%	1.5%	3%	3%

Tabuľka 6.1: Výsledky jednotlivých metód pre sloveso přihlížet (pozn.: δ *inst* označuje, akú percentuálnu zmenu predstavuje jedna inštancia z príslušných dát, viď tiež 4.1)

Pri slovese *odpovídat* by som ako najlepšiu metódu vybral SVM, pretože na testovacích dátach boli jej výsledky najlepšie, i keď iné metódy ju prekonali na trénovacích dátach.

6.1 Intervaly spoľahlivosti

Rozdiely medzi metódami však neboli veľmi veľké a preto je vhodné využiť tzv. techniku *bootstrapping*, aby sme zistili, v akých intervaloch sa pravdepodobne budú jednotlivé ukazovatele úspešnosti pohybovať. Táto metóda spočíva v pseudonáhodnom výbere testovacej množiny zo všetkých testovacích inštancií s opakovaním. Týmto spôsobom je viacnásobne skonštruovaná nová testovacia množina, na ktorej sa zistí úspešnosť modelov pre jednotlivé algoritmy. Z výsledkov na týchto testovacích množinách je potom vytvorený 95% interval spoľahlivosti výsledkov daného algoritmu. Tieto intervaly spoľahlivosti sme vytvorili pomocou funkcií `boot` a `boot.ci` z knižnice `boot` programu R.

odpovídat				
	train		test	
metóda	F_μ	F_M	F_μ	F_M
baseline	61.5%	25.4%	62.5%	25.6%
C5.0	93.8%	92.3%	90.6%	85.2%
rpart	89.2%	85.0%	84.4%	80.9%
SVM	92.3%	90.9%	96.9%	90.2%
NBK	87.7%	87.4%	96.9%	89.6%
δ *inst*	1.5%	1-5%	3%	2-10%

Tabuľka 6.2: Výsledky jednotlivých metód pre sloveso odpovídat (pozn.: δ *inst* označuje, akú percentuálnu zmenu predstavuje jedna inštancia z príslušných dát, viď tiež 4.1)

Na porovnanie dvoch algoritmov sa použije rozdiel úspešnosti pre dvojicu algoritmov. Ak 95% interval spoľahlivosti pre rozdiel obsahuje nulu, rozdiel v úspešnosti algoritmov nie je významný (tieto i ďalšie metódy bootstrappingu sú popísané v [10]).

Pri metódach rpart, NBK a SVM sme použili 1000-násobný pseudonáhodný výber na vytvorenie 1000 nových testovacích množín. Na týchto množinách sme potom sledovali správanie ukazovateľa F_μ. Keďže je nástroj pre algoritmus C5.0 iba s grafickým rozhraním, bootstrapping by bolo veľmi prácne uskutočniť i vyhodnotiť, preto sme, bohužiaľ, pre tento algoritmus bootstrapping nevykonali. Výsledky bootstrappingu sa nachádzajú v tabuľkách 6.3 a 6.4.

6.1.1 Přihlížet

přihlížet	
metóda	95% interval F_μ test
rpart	(84.8, 100)%
SVM	(90.9, 100)%
NBK	(100, 100)%

Tabuľka 6.3: Výsledky bootstrappingu jednotlivých metód pre sloveso přihlížet (pozn.: jedna inštancia znamená zmenu približne 3% pri F_μ)

Rozdiel medzi použitím algoritmu rpart a SVM je s 95% pravdepodobnosťou v intervale $(0, 9)\%$, rozdiel medzi SVM a NB je s 95% pravdepodobnosťou v intervale $(-9, 0)\%$ a rozdiel medzi NBK a rpart v intervale $(0, 15)\%$. Každý z týchto intervalov obsahuje 0, nemôžeme teda jednoznačne povedať, že niektorá z použitých metód je lepšia než iná.

6.1.2 Odpovídat

Rozdiely sa však ukázali pri slovese *odpovídat*, keď rozdiel medzi modelom vytvoreným algoritmom rpart a SVM leží s 95% pravdepodobnosťou v inter-

odpovídat	
metóda	95% interval F_μ test
rpart	(65.6, 93.8)%
SVM	(90.6, 100)%
NBK	(90.6, 100)%

Tabuľka 6.4: Výsledky bootstrappingu jednotlivých metód pre sloveso přihlížet (pozn.: jedna inštancia znamená zmenu približne 3% pri F_μ)

vale $(6, 28)\%$. Rovnako i rozdiel medzi rpart a NBK leží v intervale $(6, 28)\%$. Rozdiel medzi NBK a SVM pri tomto slovese sa ukázal byť nulový.

Ako najúspešnejšie sa v kontexte našich experimentov javia metódy NBK a SVM, medzi nimi sme však nenašli signifikantný rozdiel.

6.2 Analýza rámcov slovies

Na základe atribútov, ktoré si vybrali jednotlivé metódy sa pokúsime zistiť, aké rámce a významy slovies bolo potrebné od seba odlíšiť.

6.2.1 Přihlížet

Pri slovese *přihlížet* boli najčastejšie vyberanými atribútmi

- S2_prep__plus_3 – nejaká predložka v datíve je syntakticky závislá na danom slovese
- S2_part_se – reflexívne zámeno ,,se“ je závislé na danom slovese
- M_2_5 – pád slova dve pozície za skúmaným slovesom

Predložkou v datíve bude zrejme ,,k/ke“ (přihlížet k někomu/něčemu), ide teda o význam podobný ako slovné spojenie brať čosi do úvahy.

Druhým významom je niečo pasívne sledovať, prizerať sa niečomu a tento sa nespája s predložkou a zvyčajne ani so zvratným zámenom se.

Domnievam sa, že tvar so zvratným zámenom sa používa najmä v pasívnych konštrukciách ako napríklad

,,Při vývoji projektu *se přihlíželo* k možnostem bodoucího rozšíření...“,

no môže sa zriedkavejšie vyskytovať i v druhom význame, napríklad

,,Lidskému neštěstí *se* často jenom *přihlíží*.“

Slovo dve pozície za slovesom bude v prvom význame často buď predložka ,,k/ke“ alebo slovo nasledujúce za danou predložkou, preto i pád tohto slova bude datív. Naproti tomu slovo v inštrumentáli nám môže naznačovať druhý význam, napríklad v konštrukciách
,,...*přihlížet* se založenýma rukama...“

„...*přihlížet* se svým otcem...“

6.2.2 Odpovídat

Pri slovese *odpovídat* sme mali rozlišovať tri valenčné rámce. Atribúty použité vo väčšine algoritmov sú:

- S2_za_-4 – predložka „za“ v akuzatíve je syntakticky závislá na slovese
- S2_prep_+4 – nejaká predložka v akuzatíve je závislá na danom slovese
- A_V_anym3 – životné substantívum v datíve je závislé na danom slovese

Pozrime sa teda, ktoré tri významy odlišujeme. Prvý odlíšený význam sa spája s predložkou „za“ (odpovídat za něco/někoho), vo význame niesť za niečo zodpovednosť, mať čosi na starosti.

Ak sa sloveso nespája s predložkou „za“, no spája sa s inou predložkou v akuzatíve, bude to takmer výlučne predložka „na“, tým dostávame druhý význam – reagovať na niečo/niekoho.

Ďalším atribútom je životnosť/neživotnosť predmetu v datíve. Pokiaľ ma neklame cit pre jazyk, tak „odpovídat někomu“ sa používa vo význame reagovať na niekoho (teda význam číslo 2), na druhej strane „odpovídat něčemu“ má význam byť v súlade, korešpondovať s niečím.

Je zaujímavé, že tieto atribúty boli vybrané ako významné algoritmom `C5.0`, zatiaľ čo algoritmus `rpart` vyberal skôr atribúty, ktoré v iných metódach nefigurovali. Algoritmy SVM a NBK našli mnoho kombinácií atribútov s najlepším skóre, no vybratím tých najčastejšie sa opakujúcich sa dá bez problémov dospieť k malej množine skutočne dôležitých atribútov.

Kapitola 7

Záver

Našou úlohou v tomto projekte bolo otestovať a porovnať metódy strojového učenia na probléme rozpoznávania valenčných rámcov dvoch slovies – *přihlížet* a *odpovídat*. Použili sme štyri algoritmy – dva pre rozhodovacie stromy (rpart a C5.0), naivný bayesovský klasifikátor a support vector machines. Metódy NBK a SVM dosiahli lepšie výsledky než rozhodovacie stromy, pri slovese *odpovídat* bol rozdiel štatisticky významný, pri slovese *přihlížet* nie.

Najlepšie výsledky pre sloveso *přihlížet* dosiahla metóda NBK, ktorá dokázala predpovedať správne klasifikáciu na testovacích dátach s mikro- i makropriemerom F-measure na úrovni 100%.

Pre sloveso *odpovídat* dosiahla najvyššie ukazovatele metóda SVM, konkrétne $F_\mu = 96.9\%$ a $F_M = 90.2\%$.

Ukázalo sa tiež, že i jednoduchý wrapper algoritmus na výber atribútov – forward-selection – dokáže s heuristickým prístupom nájsť kombinácie atribútov s veľmi dobrými ukazovateľmi úspešnosti.

Pre sloveso *přihlížet* sa ako najvýznamnejšie javia atribúty

- nejaká predložka v datíve je syntakticky závislá na danom slovese
- reflexívne zámeno ,,se“ je závislé na danom slovese
- pád slova dve pozície za skúmaným slovesom.

Význam slovesa *odpovídat* najlepšie predpovedajú atribúty

- predložka ,,za“ v akuzatíve je syntakticky závislá na slovese
- nejaká predložka v akuzatíve je závislá na danom slovese (zrejme predložka ,,na“)
- životné substantívum v datíve je závislé na danom slovese.

Tieto atribúty boli použité pri predikcii významu takmer všetkými metódami. Bližia interpretácia významov jednotlivých slovies sa nachádza v kapitole 6.2.

Algoritmus C5.0 bol úspešný v nachádzaní atribútov, ktoré sa ukázali podstatné i v ostatných metódach, zatiaľ čo rpart bol v tomto úspešný menej. Subjektívne sa javí, že pravidlá vytvorené algoritmom C5.0 sú zmysluplnejšie ako pravidlá vytvorené algoritmom rpart, vzorka našich slovies je však malá, prílišná generalizácia teda nie je na mieste. Rpart má výhodu väčšej kontroly

nad orezávaním a predchádzaním pretrénovaniu, keďže C5.0 používa uzavretú implementáciu a naviac je to nástroj nie veľmi vhodný na dávkové spracovanie dát.

V súlade s predchádzajúcimi experimentami, spomenutými v kapitole 3, najvyššiu úspešnosť dosiahli metódy NBK a SVM.

Zaujímavé by mohlo byť porovnať výsledky viacerých algoritmov na výber atribútov, a to jednak ďalších wrapper metód i niektorých filtrovacích metód, no to už je úloha inej práce.

Ďalšie experimentovanie s väčším množstvom inštancií by tiež mohlo odhaliť zaujímavé fakty a tak priniesť odpoveď na otázku, ktorá z metód strojového učenia si s touto úlohou poradí najlepšie.

Literatúra

[1] Guyon I.; Elisseeff A. 2003. *An Introduction to Variable and Feature Selection.* in: Journal of Machine Learning Research 3 1157-1182.

[2] Hajič, J. 2004. *Complex Corpus Annotation: The Prague Dependency Treebank.* Bratislava, Slovakia. Jazykovedný ústav Ľ. Štúra, SAV.

[3] Quinlan R. 2008. *Data Mining Tools See5 and C5.0.* http://www.rulequest.com/see5-info.html, 6.12.2009.

[4] Manning Ch. D.; Schütze H. 2008. *Foundations of Statistical Natural Language Processing.* MIT Press. Cambridge, MA: May 1999.

[5] Navigli, R. 2009. *Word sense disambiguation: A survey.* ACM Comput. Surv. 41, 2, Article 10 (February 2009), 69 pages http://doi.acm.org/10.1145/1459352.1459355.

[6] Nilsson, R.; Pena, J. M.; Bjorkegren, J.; Tegner, J. 2006. *Evaluating Feature Selection for SVMs in High Dimensions.* in Lecture Notes in computer science 2006, NUMB 4212, pages 719-726, Springer-Verlag Germany. ISSN 0302-9743

[7] Ratanamahatana, Ch. "Ann"; Gunopulos, D. 2006. *Scaling up the Naive Bayesian Classifier: Using Decision Trees for Feature Selection.*

[8] Semecký, J. 2007. *Verb Valency Frames Disambiguation.* (PhD. thesis). Institute of Formal and Applied Linguistics, Charles University in Prague.

[9] Sokolova, M.; Lapalme, G. 2009 *A systematic analysis of performance measures for classification tasks.* In: Inf. Process. Manage., Vol. 45, Nr. 4 (2009), S. 427-437

[10] Venables, W. N.; Ripley, B. D. 2002 *Modern applied statistics with S.* Springer-Verlag. New York.

[11] Weston, J.; Mukherjee, S.; Chapelle, O.; Pontil, M.; Poggio, T.; Vapnik, V. 2001. *Feature Selection for SVMs.* in Advances In Neural Information Processing Systems, ISSU 13, pages 668-674, MIT Press Great Britain, ISSN 1049-5258