Postavljanje vejic v slovenščini s pomočjo strojnega učenja in izboljšanega korpusa Šolar

Anja Krajnc, Marko Robnik-Šikonja

Fakulteta za računalništvo in informatiko, Univerza v Ljubljani
Večna pot 113, 1000 Ljubljana
anja.krajnc@gmail.com, marko.robnik@fri.uni-lj.si

Povzetek

Poskušamo izboljšati trenutne pristope strojnega učenja za postavljanje vejic v slovensko besedilo. Generiramo nove atribute na podlagi slovničnih pravil za slovenski jezik, ki z dodatno informacijo omogočijo boljše učenje. Za analizo uporabimo korpus Šolar, ki je bil uporabljen v že obstoječi raziskavi, in izboljšano verzijo tega korpusa. Uporabimo vzorčenje neuravnoteženih množic ter odstranimo neinformativne atribute. Z metodo ReliefF ocenimo kakovost množice atributov. Testiramo različne klasifikacijske algoritme: naključne gozdove, metodo podpornih vektorjev, naivni Bayesov klasifikator, RBF mrežo, alternirajoče odločitveno drevo, AdaBoostM1 ter odločitveno tabelo. Najboljše rezultate dosežemo z naključnimi gozdovi, alternirajočimi odločitvenimi drevesi in odločitveno tabelo. Ugotovimo, da trenutni korpusi niso primerni za izdelavo modelov, ki bi bili splošno uporabni.

Placing comma in Slovene using machine learning and updated corpus Šolar

We improved current machine learning approaches to comma placement in Slovene language. First we generate new features based on grammar rules. This additional information improves performance of machine learning approaches. We learn from corpus Šolar and its updated version. We use undersampling of imbalanced data sets and feature subset selection with ReliefF algorithm. The classification methods we test are random forests, support vector machines, naïve Bayesian classifier, RBF network, alternating decision trees, AdaBoost.M1, and decision table. The best performing methods are random forests, alternating decision trees, and decision table. We conclude that current learning corpuses do not allow construction of generally applicable models.

1 Uvod

Pravilno postavljene vejice v besedilu ne puščajo samo dobrega vtisa pri bralcih in odsevajo verodostojnost in strokovnost besedila, pač pa omogočajo tudi lažje in enolično razumevanje vsebine stavkov, ločijo stavke znotraj povedi in nakazujejo premor v govoru. Hkrati lahko napačno postavljene vejice spremenijo pomen stavkov. Programi, ki uspešno postavljajo vejice v besedilo, so pomembni tako za pisce, ki naletijo na zadrego pri pisanju, kot tudi pri računalniški razpoznavi in obdelavi govora. Postavljanja vejic s strojnim učenjem se lotimo, ker ima slovenščina zahtevno oblikoslovno podobo in so pravila za pisanje vejic velikokrat zahtevna za razumevanje, njihova programska implementacija pa težko uresničljiva. Prav tako je strojno postavljanje vejic del zahtevnejših jezikovnih tehnologij, katerih cilj je vzdrževanje večjezičnosti.

Za postavljanje vejic v slovensko besedilo obstajata dva programa, ki delujeta na podlagi ročno napisanih pravil: Besana (Amebis d.o.o., 2015) in LanguageTool (LanguageTool skupnost, 2015). Peter Holozan se je lotil postavljanja vejic s pomočjo strojnega učenja na seznamu primerov iz korpusa Šolar (Holozan, 2012; Holozan, 2013). Strojno učenje je uporabil za problem iskanja vseh vejic ter za problem popravljanja realnih napak v besedilu, torej za iskanje odvečnih vejic. Njegove obsežne študije, ki bistveno presegajo obseg tega dela, želimo izboljšati predvsem z izboljšano predstavitvijo informacije, ki je na voljo strojnemu učenju. Uvedemo več izboljšav množice atributov in predlagamo uporabo še netestiranih uspešnih učnih algoritmov, ki so se izkazali na podobnih nalogah. Zahtevnost računskih operacij, ki se je v dosedanjih raziskavah izkazala kot omejitveni dejavnik za več učnih algoritmov, želimo zmanjšati z ocenjevanjem kakovosti atributov in s podvzorčenjem. Podrobneje o naših poskusih poročamo v (Krajnc, 2015), tukaj pa povzamemo bistvene ugotovitve.

Bistveno za uspeh napovedovanja z algoritmi strojnega učenja je, da algoritmom priskrbimo kakovostno informacijo o problemu in to v takšni obliki, da so jo sposobni izkoristiti, glede na njihove predpostavke in formalizem, ki ga uporabljajo za predstavitev napovednega modela. Izhajali smo iz korpusa Šolar, ki vsebuje popravljena besedila šolskih esejev. Ker se je med analizo izkazal za neprimernega za strojno učenje, smo preizkusili tudi izboljšano verzijo tega korpusa, ki jo je sestavil in nam jo posredoval Peter Holozan (2015). Ta korpus vsebuje približno trikrat več primerov z vejicami, prav tako je bilo veliko primerov ročno pregledanih in popravljenih. Podrobneje o zgradbi podatkovne množice poročamo v razdelku 2

Dosedanji poskusi s strojnim učenjem so bili le delno uspešni. Menimo, da je mogoče algoritmom priskrbeti še dodatno informacijo in vključiti tudi znanje, skrito v pravilih za postavljanje vejic. Za generiranje novih atributov na podlagi slovničnih pravil smo razvili orodje, ki uporabi pravila, ki jih za postavljanje manjkajočih vejic uporablja LanguageTool (2015), in jim dodali še dodatna pravila, ki jih narekuje Slovenski pravopis (Jože Toporišič in sod., 2001). Dosedanji atributni opis korpusa ima več pomanjkljivosti, ki jih analiziramo v razdelku 3 Neinformativne atribute odstranimo, nekatere pa modificiramo.

Na izboljšani podatkovni množici uporabimo algoritme, ki so že bili uporabljeni na osnovni podatkovni množici v predhodni raziskavi, to so naivni Bayesov klasifikator,

RBF mreža, alternirajoče odločitveno drevo, AdaBoostM1 in odločitvena tabela, ter uporabimo še dva algoritma, ki sta se v večini primerjalnih študij, npr. (Caruana in Niculescu-Mizil, 2006) izkazala z robustnim delovanjem, to sta algoritem naključnih gozdov ter metoda podpornih vektorjev. Za zmanjšanje potrebnih računskih virov smo preizkusili dve tehniki, ocenjevanje atributov, ki bi ga lahko potencialno uporabili za izbiro podmnožice pomembnih atributov in vzorčenje. Kakovost atributov smo ocenili z algoritmom ReliefF, ki zna upoštevati tudi pogojne odvisnosti atributov glede na razred. Pri neuravnoteženih učnih množicah, kot je v našem primeru (razmerje med učnimi primeri, ko je vejica in ko je ni, je približno 1:10), se pogosto kot koristno izkaže podvzorčenje (ang. undersampling), le-to nam istočasno zmanjša velikost učne množice, kar pohitri učenje.

V nadaljevanju najprej v 2 razdelku podrobneje opišemo podatkovne množice, obstoječe atribute in nove atribute, ki jih generiramo na podlagi pravil. Izboljšave predstavitve že uporabljene množice atributov si ogledamo v razdelku 3 V 4 razdelku opišemo ocenjevanje atributov ter podvzorčenje. Rezultate klasifikacije si ogledamo v razdelku 5 V 6 razdelku povzamemo poglavitne ugotovitve študije in podamo nekaj idej za nadaljnje delo.

2 Opis podatkovnih množic

Študijo smo začeli na podatkovnih množicah, uporabljenih v Holozan (2013), ki izhajajo iz korpusa Šolar. Te podatkovne množice in korpus imenujemo *Šolar1*. V korpusu je 11.892 pravilno postavljenih vejic ter 11.399 manjkajočih. Korpus je bil oblikoskladenjsko označen z oblikoslovnim označevalnikom in lematizatorjem Obeliks (Grčar et al., 2012) ter lematiziran in skladenjsko razčlenjen s skladenjskim razčlenjevalnikom (Dobrovoljc et al., 2012). Vsaka beseda z okoliškim oknom, ki se pojavi v korpusu, je pretvorjena v seznam atributov, dodan pa je tudi atribut, ki pove, ali tej besedi sledi vejica (Holozan, 2013). Nastala podatkovna množica je v formatu ARFF, ki je sestavljen iz glave ter podatkovnega dela in je namenjen orodju WEKA (Witten in Frank, 2005). V glavi so opisani atributi, tako da je za vsak atribut podano ime atributa in vrednosti, ki jih lahko atribut zavzame, v podatkovnem delu pa je v vsaki vrstici ena beseda, opisana z vrednostmi atributov. Korpus vsebuje 23.282 primerov z vejico (pozitivnih) in 185.875 brez vejice (negativnih), skupaj torej 209.157 besed.

Ker so začetni rezultati testiranj pokazali, da je obstoječi korpus neprimeren za strojno učenje, smo uporabili izboljšano in posodobljeno verzijo tega korpusa, ki jo je sestavil Peter Holozan. Veliko stavkov v tej novi podatkovni množici, ki jo imenujemo *Šolar2*, je ročno pregledanih in popravljenih (Holozan, 2015). Izboljšan korpus vsebuje 728.927 učnih primerov, od tega 65.356 z vejico in 663.571 brez vejice.

2.1 Opis atributov

Prvotna podatkovna množica Šolar1 je vsebovala po 67 atributov za vsako besedo. Prvi atribut je pojavnica. Ta atribut lahko zavzame toliko vrednosti, kolikor je različnih besed in njenih oblik v besedilu, npr. pojavnici človek in človeka zasedeta dve mesti v zalogi vrednosti tega atributa. Naslednji atribut je lema trenutne besede. Tudi zaloga vrednosti tega atributa je zelo velika, sestavljajo pa jo vse leme besed v besedilu. Tukaj pojavnici človek in človeka zasedeta eno vrednost in sicer osnovno obliko besede, človek. Sledi zapis MSD kode besede, zaloga vrednosti pa zavzame vse možne oblikoskladenjske oznake po oblikoskladenjskih specifikacijah JOS (Erjavec et al., 2010), s katero besedi določimo besedno vrsto, sklon, spol, število itd. Naslednji trije atributi so binarni in nosijo informacijo, ki jo priskrbi skladenjski označevalnik (Dobrovoljc et al., 2012). Zasedejo lahko eno izmed vrednosti 0 ali 1 ter sporočajo, ali trenutna beseda kaže na veznike, del povedi (obstaja povezava med trenutno besedo in neko drugo besedo v povedi) ter ali obstaja povezava bodisi na osebke, predmete ali prislovna določila. Opisanih šest atributov se ponovi za vse besede znotraj okoliškega okna. Za analizo uporabljamo okoliško okno pet besed pred in pet besed za trenutno besedo. To pomeni, da imamo skupaj enajstkrat po šest zgoraj opisanih atributov. Tem atributom sledi še razred, ki pove, ali besedi sledi vejica. Zavzema lahko dve vrednosti [je-vejica, ni-vejica].

2.2 Predstavitev pravil v atributni obliki

Obstoječim atributom smo dodali 45 novih, od tega 41 z implementacijo pravil, ki jih za postavljanje vejic uporablja orodje LanguageTool ter 4 dodatna pravila, ki jih narekuje Slovenski pravopis in ki piscem pogosto povzročajo težave. Za vsako pravilo ustvarimo nov atribut.

Oglejmo si primer implementacije pravila za vezniško besedo *ker* po pravilih, ki jih za postavljanje vejic uporablja orodje LanguageTool: kadar naletimo na besedo *ker* in beseda pred njo ni eno izmed ločil *,(;-:* ali ena izmed besed *in*, *ali*, *ter*, *a* in *temveč*, potem trenutni besedi verjetno sledi vejica, zato atribut za trenutni veznik zavzame vrednost 1. Če beseda ni veznik, ki ga iščemo, ali pa je veznik, ki ga iščemo, in ne ustreza podanemu pogoju, potem atribut za ta veznik pri tej besedi zavzame vrednost 0.

Implementirali smo še nekaj dodatnih pravil, ki bodisi piscem povzročajo največ težav bodisi zahtevajo dodatno pozornost pri pisanju. Nekatera pravila smo implementirali tako, da smo dodali pogoj pri implementaciji obstoječih pravil, ostala pa smo ustvarili kot nove samostojne atribute.

2.2.1 Členek da

Kadar se *da* v stavku rabi kot podredni veznik, pred njim najverjetneje stoji vejica. Primere povzemamo po Toporišič in sod. (2001) . Primeri uporabe: *Zeblo ga je tako, da se je ves tresel. - Daj mu nageljček, da bo tudi on vedel, da je prvi maj. - Bral je knjige, da bi spoznal svet.* Posebnost nastopi, kadar se *da* v stavku rabi kot členek. V takšnem primeru pred njim ne pišemo vejice. Primeri uporabe: *Ti da tega ne znaš? - Baje da vozi. - Iz Pirana da ste?* Obstoječemu atributu, generiranemu s pomočjo pravil za veznik *da*, smo dodali nov pogoj, kjer smo preverili, ali je MSD koda trenutne besede enaka L (oblikoskladenjska oznaka členka). Če je pogoj izpolnjen, atribut brez nadaljnjih preverjanj dobi vrednost 0. Šele ob neizpolnjevanju pogoja so se izvršila preverjanja ostalih pravil za postavitev vejice, ki ustrezajo besedi *da*, kadar se rabi kot podredni veznik. Tako program, kadar naleti na besedo *da*, preveri

ali je MSD koda besede enaka L, kar pomeni, da je beseda členek. Če je pogoj izpolnjen, atributu za ta veznik vrednost nastavi na 0. Če pogoj ni izpolnjen, kar pomeni, da beseda ni členek (veliko upov polagamo v točnost oblikoskladenjskega označevalnika), se izvršijo pogoji za ta veznik in vrednost atributa nastavi v skladu z njimi. Za pogoj, ki preverja, ali je beseda členek, ne pa za pogoj, ki preverja, ali je beseda veznik, smo se odločili, ker lahko beseda *da* nastopa tudi v vlogi drugih besednih zvez, npr. glagol v sedanjiku in tretji osebi ednine. Primer: *Nikoli ga ne da iz rok.*

2.2.2 Večbesedni vezniki

Slovenski pravopis zapoveduje, da vejic med deli večbesednega veznika ne pišemo. Primeri uporabe: *Namesto da bi se učil, je lenaril. - Lenaril je, namesto da bi se učil. - Že ko sem ga prvič videl, sem ga spregledal. - Kljub temu da je bil bolan, je šel na delo. - Rad ga imam, zato ker je tako miren.* Taki vezniški izrazi so še: tako da, toliko da, potem ko, vtem ko, še ko, brž ko, šele ko, s tem da, posebno ko, zlasti če itd. (Jože Toporišič in sod., 2001). To smo implementirali tako, da smo vsem do sedaj generiranim atributom dodali dodaten pogoj, ki pravi: če je trenutna beseda veznik, ki ustreza vsem pravilom za postavljanje vejice za ta vezniški izraz, in pred njo ne stoji beseda z MSD kodo za veznik (Vp ali Vd), potem atribut za ta veznik prejšnji besedi nastavi na 1. V nasprotnem primeru dobi atribut vrednost 0.

2.2.3 Dodaten pogoj za večbesedne veznike

S prejšnjim načinom smo poskušali čim bolj natančno najti večbesedne veznike v besedilu z uporabo MSD kod ob predpostavki, da označevalnik povsem natančno določa besedne vrste. To smo storili zaradi enostavnosti te rešitve. Ker Obeliks besedne vrste pripisuje z 98,30 % natančnostjo (Grčar et al., 2012), morda z zgoraj opisanim postopkom v oblikoslovni označevalnik polagamo preveč zaupanja. Zato implementiramo tudi izvedbo pravila brez uporabe MSD kod in brez zanašanja na označevalnik za primere, ko veznik *in* nastopa v kombinaciji z besedami *sicer*, *to*, *če* in *ko* ter tvori vezniški izraz, pred katerim verjetno stoji vejica. To smo implementirali tako, da smo za trenutno besedo preverili, ali je enaka besedi *in*, nato preverili, ali je naslednja beseda enaka eni izmed zgoraj naštetih besed, in če sta pogoja izpolnjena, prejšnji besedi (besedi na položaju pred trenutno besedo) atribut za ta veznik nastavili na 1, v nasprotnem primeru pa na 0.

3 Izboljšave predstavitve atributov

Predstavitev nekaterih atributov smo spremenili, da bi algoritmi strojnega učenja bolje izkoristili v njih vsebovano informacijo. Nekatere atribute, ki strojnemu učenju ne koristijo, smo izločili iz podatkovne množice.

3.1 Obravnava pojavnic in lem

V dosedanjih poskusih strojnega učenja na problemu vejic so bili atributi sestavljeni tako, da sta dve besedi, ki imata isto oblikoskladenjsko oznako, obravnavani kot dve različni vrednosti atributa. To ni smiselno, saj v slovenščini (ne)obstoj vejice ni odvisen od besede same kot oznake pojma, pač pa od njene besedne vrste (ponekod tudi od nekaterih oblikoskladenjskih lastnosti besede) in povezav z drugimi besedami v povedi. Zato je bolj smiselno, da posameznih besed ne obravnavamo kot ločenih vrednosti atributov. S tem se izognemo drugačni obravnavi besed iste besedne vrste (ali celo iste oblikoskladenjske oznake), ki določajo enako vez z drugimi besedami in v stavku nosijo isti pomen. S tem premislekom atribute, ki določajo posamezne besede, odstranimo. Enako kot je nesmiselno definirati atribut glede na samo besedo, velja tudi za leme. Tudi atribute, ki opisujejo leme besed, odstranimo iz množice atributov.

3.2 Obravnava oblikoskladenjskih oznak

Vsaka beseda iz korpusa Šolar1 je bila do sedaj v atributih predstavljena z eno od 930 različnih oblikoskladenjskih oznak. Samostalnik lahko zavzame eno izmed 78 vrednosti, zaimek eno izmed 451 vrednosti, pridevnik ima 168 oznak in glagol 127 različnih oznak. Z bogatejšim in večjim korpusom bi bile te številke še večje, saj bi nastopalo še več možnih oblik. Pri tem se pojavi vprašanje, ali je to smiselno. V slovenščini vejice postavljamo glede na besedne vrste in povezave med besedami v povedi, ne pa glede na vse možne kombinacije lastnosti, vidov in oblik teh besednih vrst. Povezave in pravila za postavljanje vejic je težko razbrati, če za stvari, ki bi nam morale podati isto informacijo, uporabljamo več deset različnih oznak. Po tem premisleku smo neinformativne atribute, ki opisujejo MSD kode in so lahko zavzeli eno izmed 930 vrednosti, spremenili na dva načina. Tako dobimo dve podatkovni množici, ki se razlikujeta v številu in vrednosti tistih atributov, ki nam povedo oblikoskladenjske lastnosti trenutne besede in ostalih besed znotraj okna.

3.2.1 Zapis MSD kode s po enim atributom za vsako besedo znotraj okna

Prvi način spremembe atributov, ki nosijo podatek o MSD kodi, vpliva na enajst atributov znotraj okna. Na primer, vse različne oznake samostalnika, ki so kombinacije vrste, spola, števila, sklona in živosti, ki pri samostalniku tvorijo mnogo različnih oblikoskladenjskih oznak, nadomestimo z oznako S, ki bo strojnemu učenju povedala najpomembneje - naleteli smo na samostalnik. S tem se izognemo situacijam, kjer algoritem strojnega učenja nima dovolj podatkov o neki oznaki, ki se redko pojavi (ali pa se npr. pojavi samo v testni množici), lahko pa bi se naučil ukrepanja pri isti besedni vrsti v drugem spolu (in/ali številu, sklonu,...) in s tem z drugačno oznako. Tudi pri pridevniku vse možne oznake, ki določajo, da je beseda pridevnik in hkrati njegovo vrsto, stopnjo, spol, število, sklon in določnost (po oblikoskladenjskih oznakah JOS jih je 279, v našem korpusu pa 168), zamenjamo s skupno oznako za pridevnike (P). Enako storimo, kadar naletimo na prislov (R), zaimek (Z), številnik (K), predlog (D) in glagol (G). MSD oznako za kategorijo Neuvrščeno, kamor spadajo razne neznane besede, tipkarske napake in besede, ki jih je program napačno tokeniziral, nadomestimo z N.

Obdržimo oznake za veznik. Po oblikoskladenjskih oznakah JOS (Erjavec et al., 2010) veznik označimo s črko V ter dodatno črko, ki določa, ali je veznik podredni ali

priredni. Tako besedo, kadar ugotovimo, da je veznik, označimo z oznako Vd ali Vp. Ker je stavljenje vejic velikokrat odvisno od vrste veznika, obdržimo obe oznaki. Prav tako obdržimo oznake za medmet (M), okrajšavo (O) in členek (L), saj so pri njih oznake dolge en znak.

3.2.2 Zapis MSD kode s po 9 atributi za vsako besedo znotraj okna

S prvim načinom izboljšanja MSD oznak smo atribute, ki opisujejo oblikoskladenjske oznake, zamenjali s prvo črko oznake (razen pri veznikih, kjer smo obdržali prvi dve črki). S tem smo obdržali samo podatke o besedni vrsti. Z drugim načinom vključimo vse podatke, ki jih nosi posamezna MSD koda. S tem želimo zagotoviti potencialno več informacije za strojno učenje, tvegamo pa morebitno pretirano prilagajanje podatkom.

Najdaljšo oblikoskladenjsko oznako ima zaimek, kjer je MSD koda sestavljena iz 9 znakov, zato iz osnovne MSD kode generiramo devet atributov. V dosedanji predstavitvi MSD koda določa en atribut za vsako besedo, to je 11 atributov za vse besede znotraj okna, posamezna vrednost vsakega izmed atributov je dolga največ 9 znakov, vseh vrednosti pa je 930. Z novo implementacijo vsak atribut, ki nosi podatek o MSD kodi, spremenimo v devet novih, tako da vsak izmed novih atributov dobi vrednost enega od znakov MSD kode. Kadar je dolžina MSD kode manjša od 9 znakov, atributom, ki določajo preostale znake, nastavimo vrednost *, ki je znak za manjkajočo vrednost. Ker spreminjamo 11 atributov, torej po en atribut za vsako besedo znotraj okna, to pomeni, da enajst osnovnih atributov nadomestimo z 99 novimi. Spremenjena podatkovna množica se tako poveča iz 90 na 178 atributov za opis trenutne besede.

3.2.3 Zapis MSD kode s po 38 atributi za vsako besedo znotraj okna

MSD oznake bi lahko opisali še na tretji način, ki bi strojnemu učenju morebiti podal nekaj več informacije o povezavah med deli MSD kode za posamezno besedno vrsto. Kot prvi atribut bi obdržali prvo črko MSD oznake, ki nam pove besedno vrsto. Ta atribut bi bil edini skupen vsem besednim vrstam in bi zasedal eno izmed dvanajstih vrednosti: S za samostalnik, G za glagol, P za pridevnik, R za prislov, Z za zaimek, K za števnik, D za predlog, V za veznik, L za členek, M za medmet, O za okrajšavo ter N za neuvrščeno. Temu atributu bi sledili ločeni atributi za vsako posamezno besedno vrsto in sicer za vsako besedno vrsto toliko atributov, kolikor dolge so njene MSD oznake. Če bi začeli s samostalnikom, pri katerem lahko njegova MSD oznaka zasede šest mest, od tega prvi znak določa besedno vrsto in smo njegovo vrednost že zapisali v prvi atribut, bi to pomenilo, da bi bili atributi na mestih od drugega do šestega rezervirani za podatke o samostalniku. Pri vseh ostalih besednih vrstah bi bili atributi na teh mestih nastavljeni na vrednost *. Temu bi sledilo 7 atributov za oznake glagola, 6 za pridevnik, 2 za oznake prislova, 8 za zaimek, 6 za števnik, po 1 za predlog, veznik in neuvrščeno. MSD oznake medmetov in okrajšav so dolge po en znak, ki nam pove besedno vrsto, to pa že vsebuje prvi atribut, zato za ti dve besedni vrsti dodatnih atributov ne bi definirali. MSD oznake bi torej opisali s po 38 novimi atributi za vsako besedo znotraj okna. To bi skupaj pomenilo 418 atributov, ki bi nosili informacije o MSD oznakah. Implementacijo in testiranje tretjega načina puščamo za nadaljnje delo, saj zahteva več računskih kapacitet.

4 Ocenjevanje atributov in podvzorčenje

Zanima nas pomembnost atributov, tj. koliko informacije posamezen atribut vsebuje pri klasifikaciji za dani učni problem. Kvaliteto atributov ocenjujemo z mero za ocenjevanje atributov ReliefF (Robnik-Šikonja in Kononenko, 2003). ReliefF je široko uporabljan algoritem za ocenjevanje atributov, ki zaznava tudi pogojne odvisnosti med atributi, uspešno obravnava šumne in neznane vrednosti ter deluje tudi za večrazredne probleme.

Ocenjevanje atributov smo pognali na podatkovnih množicah, ki izhajajo iz korpusa Šolar1 in Šolar2, na vsakem po dve množici. Par podatkovnih množic se med seboj razlikuje pri atributih, ki podajajo informacijo o oblikoskladenjski oznaki. Prva množica iz para vsebuje 90 atributov in ima vsako oblikoskladenjsko oznako opisano z enim atributom. Znotraj okna je tako 11 atributov, ki opisujejo oblikoskladenjske oznake, zato ti množici imenujemo Šolar1-11 in Šolar2-11. Drugi dve množici na vsakem korpusu vsebujeta 178 atributov in imata vsako oblikoskladenjsko oznako opisano s po devetimi atributi (99 atributov za opis oblikoskladenjskih oznak in zato poimenovanje Šolar1-99 in Šolar2-99). Zanima nas, kako pomembni so atributi in koliko informacije nosijo oblikoskladenjske oznake, če jih opišemo bodisi z enim bodisi z devetimi atributi. Ker se zaključki glede ocen atributov z mero ReliefF ne razlikujejo med korpusom Šolar1 in Šolar2, predstavimo v tabeli 1 le rezultate za korpus Šolar2 in sicer po 15 najbolje ocenjenih atributov. Rezultati množice Šolar2-11 so na levi strani in množice Šolar2-99 na desni strani tabele 1.

Za Šolar2-11 so najbolje ocenjeni atributi o MSD kodah besed na mestih 0, 1, -1, -2, in 2. Na desni strani tabele 1 se za Šolar2-99 pokaže, da je najpomembnejša beseda na mestu 0, ki z različnimi oznakami zavzame kar štiri od petih najvišjih mest (msd0-5, msd0-3, msd0-4, msd0-1). Iz tega sklepamo, da nosi drugi način opisa MSD kod več koristne informacije. Kljub temu pa je treba upoštevati, da sta korpusa Šolar1 in Šolar2 zelo specifična in da vsebujeta več med seboj podobnih besedil, tako da rezultatov ne moremo preveč posploševati.

Rezultate algoritma ReliefF bi lahko uporabili za izbor podmnožice pomembnih atributov, s čimer bi lahko omejili trenutno dokaj visoko porabo računskih virov. Zaradi pomanjkanja prostora poskusov s to možnostjo, ki koristi predvsem manj zmogljivim algoritmom strojnega učenja, nismo vključili v članek.

Za neuravnotežene učne množice, kot je naš korpus Šolar, kjer je razmerje med mesti z vejico in brez nje približno 1:10, pogosto za izboljšanje delovanja učnih algoritmov koristi uravnoteženje učne množice. Ena najbolj uveljavljenih tehnik uravnoteženja, ki istočasno tudi zmanjša zahtevnost računanja, je podvzorčenje (ang. undersampling). Le-to smo preizkusili tudi na našem problemu, a ker rezultati bistveno ne odstopajo od tistih brez uravnoteženja, o tem v nadaljevanju ne poročamo, bi pa bilo tehniko smiselno uporabiti za določanje optimalnih parametrov, ki zahtevajo mnogo ciklov učenja, in na bodočih, večjih korpusih.

	Šolar2-11		Šolar2-99	
rang	ocena	atribut	ocena	atribut
1	0,355	msd0	0,357	msd0-5
2	0,271	msd1	0,353	msd0-3
3	0,237	msd-1	0,340	msd0-4
4	0,161	msd-2	0,337	msd-1-1
5	0,132	msd2	0,320	msd0-1
6	0,132	msd-3	0,292	msd-1-2
7	0,120	msd-4	0,261	msd0-2
8	0,106	msd-5	0,257	je_vez1
9	0,099	zac_modrega0	0,233	msd-1-4
10	0,099	je_vez1	0,215	msd-1-5
11	0,082	zac_modrega1	0,214	msd-1-3
12	0,082	msd3	0,195	msd1-1
13	0,063	zac_rdecega1	0,189	zac_modrega1
14	0,061	je_vez0	0,187	msd-2-1
15	0,060	msd4	0,165	msd1-2

Tabela 1: Rezultat ocenjevanja atributov z mero ReliefF na množicah Šolar2. Na levi strani imamo za vsako MSD oznako po en atribut (Šolar2-11), na desni pa po 9 atributov (Šolar2-99).

5 Rezultati klasifikacije

Testirali smo več različnih klasifikacijskih algoritmov, ki so že bili uporabljeni v prejšnjih raziskavah. Naivni Bayesov klasifikator (NaiveBayes) je enostaven verjetnostni klasifikator, ki predpostavlja, da so atributi med seboj pogojno neodvisni. Odločitvena tabela (DecisionTable) izbere pomembne atribute, potem pa primere klasificira glede na njim najbolj podoben primer iz učne množice. RBF mreža (ang. Radial Basis Function Network, RBFnetwork) je nevronska mreža, ki klasificira glede na razdaljo primerov do središč Gaussovskih jeder, ki jih vsebujejo nevroni. Alternirajoče odločitveno drevo (ang. Alternating Decision Tree, ADTree) je posebne vrste odločitveno drevo, ki implementira boosting. AdaBoostM1 prav tako uporablja idejo boostinga, kar pomeni, da med učenjem napačno klasificiranim učnim primerom povečuje uteži, zato postanejo ti pomembnejši in se učenje osredotoča nanje. Dodatno smo uporabili dva nova algoritma, ki pri večini primerjalnih študij, npr. (Caruana in Niculescu-Mizil, 2006), dosegata visoko klasifikacijsko točnost, to sta metoda podpornih vektorjev (SMO) in metoda naključnih gozdov (RandomForest). Pri SMO lahko, za razliko od večine drugih algoritmov, vključimo vse razpoložljive atribute, saj metoda sama kombinira atribute, vendar je tudi zato časovno zahtevna. Metoda RandomForest vrača skupinsko napoved množice randomiziranih odločitvenih dreves in s tem zmanjša napako zaradi variance napovedi.

Testiranje smo najprej opravili na originalnem korpusu Šolar1. Rezultati testiranj so potrdili, da je korpus neprimeren za strojno učenje. Odvzem neinformativnih atributov je poslabšal rezultate in najboljše rezultate smo dobili z algoritmom odločitvena tabela, kjer ne gre toliko za učenje v smislu posploševanja, ampak si zapomnimo le natančno okolico besed, kjer stoji vejica. Po natančnejšem pregledu korpusa smo ugotovili, da je poleg veliko stavkov, ki ne vsebujejo vejice, vsebuje tudi veliko napačno postavljenih vejic. Sestavlja ga mnogo podobnih besedil. Prav tako je poln nepravilno tvorjenih stavkov z mnogimi pogovornimi in tujimi besedami.

Testiranje smo zato osredotočili na podatkovne množice izboljšanega korpusa Šolar2, kjer je primerov z vejicami skoraj trikrat več. V tabeli 2 vidimo rezultate za različne statistike: natančnost, priklic in F1 posebej za učne primere, ko vejica je in ko je ni, ter dodatno še klasifikacijska točnost in AUC, ki sporoča informacijo o pravilnosti rangiranja oz. napovedanih verjetnosti. Zaradi velike časovne zahtevnosti metode podpornih vektorjev in odločitvena tabela smo morali pri teh algoritmih uporabiti zmanjšane podatkovne množice. V tem primeru smo uporabili naključno izbrano desetino množice Šolar2, kar pomeni, da smo uporabili 72.892 primerov. Pri vseh ostalih algoritmih smo uporabili Šolar2 originalne velikosti, to je 728.927 primerov. Pri metodi podpornih vektorjev smo testirali linearno in kvadratno jedro. Pri vseh algoritmih smo uporabili privzete parametre in 10-kratno prečno preverjanje. Zaradi velikih množic so razlike med metodami skoraj povsod statistično značilne.

V tabeli 1 vidimo, da za primere, kadar ni vejice, najboljše rezultate pri meri F1, ki je harmonična sredina med natančnostjo in priklicem, dobimo z odločitveno tabelo (F1=0,977), z naključnimi gozdovi in obema inačicama boostinga (AdaBoostM1 in ADTree). Razlik v rezultatih med množicami, ki različno definirajo atribute za opis MSD kod, praktično ni.

Za primer, kadar je vejica, se z F1=0,71 najbolje odreže odločitvena tabela, sledi ji algoritem naključnih gozdov. Podobno velja za klasifikacijsko točnost, kjer ponovno izstopata odločitvena tabela (95,7 %) in metoda naključnih gozdov. Pri AUC najboljši rezultat doseže metoda naključnih gozdov (97,3 %) na podatkovni množici Šolar2-99, ostale metode precej zaostajajo.

6 Zaključki

Testirali smo več pristopov strojnega učenja za postavljanje vejic v slovenščini. Modificirali smo podatkovne množice, ki so bile uporabljene v preteklih raziskavah, in uporabili nekatere nove algoritme strojnega učenja. Originalne podatkovne množice smo spremenili tako, da smo jim najprej odstranili neinformativne atribute, ki so oteževali strojno učenje in povečevali časovno zahtevnost ter porabo virov. Nato smo dodali nove atribute, generirane na podlagi pravil, ki jih za postavljanje vejic uporablja orodje LanguageTool, in na podlagi pravopisnih pravil za slovenski jezik. Na dva načina smo preoblikovali atribute za zapis oblikoskladenjskih oznak. Pri prvem načinu smo za zapis atributa uporabili le besedno vrsto besed znotraj okoliškega okna. Pri drugem načinu smo obdržali celotno oblikoskladenjsko oznako tako, da smo jo razdelili na 9 novih atributov, kjer je vsak atribut nosil po en znak oznake. Poskusili smo izboljšati rezultate s prilagojenim podvzorčenjem, vendar za zdaj še ne dosežemo enake klasifikacijske točnosti.

Najbolj uspešne metode strojnega učenja so algoritmi naključna drevesa, alternirajoče odločitveno drevo ter odločitvena tabela. Predvidevamo, da dober rezultat odločitvene tabele kaže, da izboljšan in posodobljen korpus Šolar2 še vedno ni najbolj primeren za strojno učenje, saj je stopnja posplošitve pri tem algoritmu nizka. Rezul-

		Ni vejice			Je vejica			Točnost [%]	AUC
		Natančnost	Priklic	F1	Natančnost	Priklic	F1		
Šolar2-99	NaiveBayes	0,979	0,840	0,904	0,333	0,813	0,473	83,746	0,905
	RBFNetwork	0,927	0,985	0,955	0,591	0,217	0,318	91,634	0,868
	ADTree	0,951	0,995	0,972	0,898	0,482	0,638	94,866	0,925
	AdaBoostM1	0,951	0,970	0,961	0,620	0,489	0,547	92,735	0,882
	RandomForest	0,956	0,996	0,976	0,925	0,536	0,579	95,455	0,973
Šolar2-11	NaiveBayes	0,983	0,769	0,863	0,269	0,861	0,410	77,754	0,903
	RBFNetwork	0,922	0,992	0,956	0,654	0,147	0,240	91,655	0,870
	ADTree	0,946	0,996	0,971	0,916	0,426	0,581	94,502	0,913
	AdaBoostM1	0,952	0,970	0,961	0,621	0,499	0,553	92,775	0,867
	RandomForest	0,957	0,995	0,975	0,913	0,542	0,680	95,428	0,943
1/10	DecisionTable	0,958	0,995	0,976	0,918	0,653	0,698	95,589	0,939
Šolar2-99	SMO-linearno	0,955	0,996	0,975	0,928	0,529	0,674	95,366	0,762
	SMO-kvadratno	0,927	1,000	0,962	0,996	0,210	0,347	92,848	0,605
1/10	DecisionTable	0,959	0,995	0,977	0,920	0,577	0,709	95,720	0,940
Šolar2-11	SMO-linearno	0,954	0,997	0,975	0,942	0,520	0,670	95,367	0,758
	SMO-kvadratno	0,943	0,992	0,967	0,834	0,392	0,534	93,799	0,692

Tabela 2: Rezultati klasifikacije na korpusu Šolar2.

tati pri najboljših algoritmih so podobni pri obeh podatkovnih množicah, kjer smo različno generirali atribute o MSD oznakah. To zelo verjetno pomeni, da ob uporabi primernega korpusa zadostuje, če za opis oblikoskladenjskih oznak uporabimo le besedno vrsto (razen pri veznikih, kjer dodamo še informacijo ali gre za priredni ali podredni veznik).

Rezultati za priklic, natančnost in mero F1 na korpusu Šolar2 z izboljšanimi in za strojno učenje primernejšimi atributi so podobni rezultatom na korpusu Šolar1 z atributi, ki opisujejo konkretne besede, leme in celotne MSD oznake (Holozan, 2013). Menimo, da je s tem postavljen dober temelj za uporabo strojnega učenja tudi na morebitnih prihodnjih izboljšanih korpusih in v orodjih za avtomatsko postavljanje vejice.

Zanimivo bi bilo videti, kako se strojno učenje odreže, če podatkovnim množicam dodamo atribute, generirane glede na vsa pravila za postavljanje vejic v slovenščini, a je to zaradi včasih ne dovolj jasnih in nedvoumnih definicij težko implementirati. Vejica poleg tega nosi tudi semantično informacijo, zato brez semantičnih atributov ne moremo pričakovati bistvenih izboljšav za ta del problema. Zanimalo bi nas tudi, kako bi se obnesla implementacija tretjega načina definiranja MSD oznak, ki smo ga opisali v razdelku 3.2.3 in zahteva več računskih zmogljivosti.

Za kvalitetno procesiranje naravnega jezika so izjemno pomembne jezikovne tehnologije, kot so lematizator, označevalnik in skladenjski razčlenjevalnik. Bistven je tudi kvaliteten korpus, ki mora biti sestavljen iz dobrih in (večkrat) lektoriranih besedil. Pri teh komponentah so izboljšave mogoče in potrebne.

7 Zahvala

Zahvaljujemo se Petru Holozanu za nasvete, podatkovne množice in dostop do izboljšanega korpusa Šolar.

8 Literatura

Amebis d.o.o., 2015. *BesAna - slovnični pregledovalnik*. http://besana.amebis.si/ [Dostop: 06/06/2015].

Rich Caruana in Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. V: *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, str. 161–168, New York, NY, USA. ACM.

Kaja Dobrovoljc, Simon Krek in Jan Rupnik. 2012. Skladenjski razčlenjevalnik za slovenščino. V: *Zbornik 8. konference Jezikovne tehnologije*, Institut Jožef Stefan, Ljubljana.

Tomaž Erjavec, Darja Fišer, Simon Krek in Nina Ledinek. 2010. The JOS linguistically tagged corpus of Slovene. V: *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, Malta.

Miha Grčar, Simon Krek in Kaja Dobrovoljc. 2012. Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. V: *Zbornik 8. konference Jezikovne tehnologije*, Institut Jožef Stefan, Ljubljana.

Peter Holozan. 2012. Kako dobro programi popravljajo vejice v slovenščini. V: *Zbornik 8. konference Jezikovne tehnologije*, Institut Jožef Stefan, Ljubljana.

Peter Holozan. 2013. Uporaba strojnega učenja za postavljanje vejic v slovenščini. *Uporabna informatika*, 21(4).

Peter Holozan. 2015. Izboljšani korpus Šolar. Osebna komunikacija.

Jože Toporišič in sod. 2001. *Slovenski pravopis*. Slovenska akademija znanosti in umetnosti.

Anja Krajnc. 2015. Postavljanje vejic v slovenščini s pomočjo strojnega učenja. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko. Diplomsko delo.

LanguageTool skupnost, 2015. *LanguageTool Style and Grammar Check*. http://www.languagetool.org [Dostop: 06/06/2015].

Marko Robnik-Šikonja in Igor Kononenko. 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning Journal*, 53:23–69.

Ian H Witten in Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.