Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anja Krajnc

Postavljanje vejic v slovenščini s pomočjo strojnega učenja

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Marko Robnik-Šikonja

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri postavljanju vejic nastane pri pisanju v slovenščini precej napak, zato bi želeli analizirati problem in narediti korak k uporabnemu pripomočku za njihovo popravljanje. Po zgledu drugih jezikov in prvih poskusov v slovenščini se problema lotite s strojnim učenjem. Preskusite več algoritmov strojnega učenja in izberite zanje primerne parametre. Uporabite obstoječe učne množice, ki so nastale na podlagi korpusa Šolar in jih obogatite z bolj informativnimi atributi. Ker so učne množice neuravnotežene in vsebujejo veliko množico atributov, poskusite uporabiti tehnike uravnoteženja in izbire podmnožice pomembnih atributov. Analizirajte rezultate in predlagajte potencialno uspešne pristope.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Anja Krajnc sem avtorica diplomskega dela z naslovom:

Postavljanje vejic v slovenščini s pomočjo strojnega učenja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom prof. dr. Marka Robnika-Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18. junija 2015 Podpis avtorja:

Kazalo

Povzetek

Cilj diplomske naloge se je naučiti postavljanja vejic s strojnim učenjem. Naš pristop temelji na generiranju novih atributov na podlagi slovničnih pravil za slovenski jezik, ki z dodatno informacijo omogočijo boljše učenje, tj. višjo natančnost in priklic. Osredotočili smo se na postavljanje vseh vejic v besedilu. Izhajali smo iz že obstoječe raziskave za postavljanje vejic v slovenščini, ki smo jo dopolnili z drugačnimi metodami učenja, drugačnimi parametri, vzorčenjem neuravnoteženih množic ter z dodatnimi informativnimi atributi. Za analizo smo uporabili korpus Šolar in izboljšano verzijo tega korpusa. Za modeliranje smo uporabili sistem za strojno učenje WEKA. Najboljše rezultate smo dosegli z algoritmi naključna drevesa, alternirajoče odločitveno drevo ter odločitvena tabela.

Ključne besede: procesiranje naravnega jezika, obdelava jezika, slovenski jezik, vejica, ločila, jezikovne tehnologije, naključni gozdovi, SVM, prečno preverjanje, podvzorčenje, strojno učenje.

Abstract

We aim to learn comma placing using machine learning. Our approach is based on adding new attributes created from grammatical rules for the Slovenian language, which provides more information and thus enable better learning, i.e., higher precision and recall. We focus on placing all the commas in the text. We modify an existing research with additional learning methods, different parameters, undersampling and knowledge based attributes. We use corpus Šolar and improved corpus Šolar for testing and machine learning toolkit WEKA. Best results were achieved with random forests, alternating decision tree and decision table models.

Keywords: natural language processing, language manipulation, Slovenian language, comma, punctuation mark, language technologies, random forest, SVM, cross-validation, undersampling, machine learning.

Poglavje 1

Uvod

Kmalu po nastanku prvih elektronskih računalnikov se je pojavila potreba po računalniški simulaciji človekovega obnašanja, tj. oponašanja človekovega načina reševanja problemov. V poplavi velikih količin podatkov, s katerimi se srečujemo na vsakem koraku, si je danes nemogoče zamisliti svet brez strojnega učenja, ki omogoča, da računalniki inteligentno rešujejo kompleksne probleme, saj zahteve in količina podatkov nemalokrat presegajo človekove kognitivne sposobnosti. Z razmahom števila področij, ki so že in še bodo odvisna od podpore računalnikov, pa se veča tudi potreba po obdelavi naravnega jezika, saj želimo, da bi se človek in računalnik naučila čimbolj naravnega komuniciranja.

Digitalna revolucija korenito spreminja komunikacijske navade in družbo nasploh, z njo pa se večajo tudi potrebe po zahtevnejših jezikovnih tehnologijah, npr. pomenski interpretaciji besedila, povzemanju razprav, avtomatskem odgovarjanju na vprašanja, detekciji mnenjskih trendov, zaznavanju čustvenih odzivov itd. Eden izmed pomembnejših ciljev jezikovnih tehnologij je vzdrževanje politike večjezičnosti s prevajanjem besedil in tolmačenjem pri govorni komunikaciji in s tem rušenje jezikovnih meja in povezovanje ne samo Evrope, pač pa celega sveta. Kakor je Gutenbergov izum tiskarskega stroja omogočil razmah količine izmenjanih informacij preko tiskane besede, vendar pa je pahnil v pozabo in izumrtje nekatere manjšinske jezike, ki so bili

omejeni le na prenašanje z govorjeno besedo, moramo v izogib temu, da se v digitalni revoluciji in ob razmahu interneta isto zgodi s slovenskim jezikom, ki ga po svetu govori ali razume samo 2,5 milijona ljudi, veliko pozornosti in razvoja namenjati jezikovnim tehnologijam za slovenščino.

Ker ima slovenščina zahtevno oblikoslovno podobo (oblikoskladenjske oznake vsebujejo kar 2000 slovničnih oznak), veliko dvoumnosti, kar je že samo po sebi velik izziv, prosti besedni red, množico različnih besednih oblik ipd., zahteva njena obdelava veliko napora. Pri tem so pomembni označevanje, razčlenjevanje ter lematizacija in hkrati dostopnost obsežnih korpusov, ki so temeljna infrastruktura za raziskovanje in procesiranje naravnega jezika.

Pravilno postavljene vejice v besedilu ne puščajo samo dobrega vtisa pri bralcih ter odsevajo verodostojnost in strokovnost besedila, pač pa tudi omogočajo lažje razumevanje vsebine besedila in nakazujejo premor v govoru. Za uspešno postavljanje vejic mora pisec podrobno poznati nemalo pravil, ki so zaradi notranje organiziranosti pravil in kategorij v pravilih zahtevna za razumevanje že maternim govorcem. Zaradi tega programi, ki uspešno postavljajo vejice v besedilo, niso uporabni le za pisce, ki naletijo na zadrego pri pisanju, pač pa so pomembni tudi pri računalniški razpoznavi in obdelavi govora. Človeka oz. pisca besedil lahko učimo pisanja vejic z raznimi interaktivnimi priročniki, kakršen je npr. Slogovni priročnik v okviru projekta Sporazumevanje v slovenskem jeziku [4]. Na ta način se naučimo, da si pri pravilnem postavljanju vejic pomagamo glede na tipične besedilne položaje ter izraze, ki so za njih značilni. Za strojno učenje je ta problem zato zelo zahteven.

Za postavljanje vejic v slovensko besedilo obstajata dva programa (Besana [5] in LanguageTool [16]), oba delujeta na podlagi ročno napisanih pravil. Peter Holozan se je lotil preizkušanja postavljanja vejic s pomočjo strojnega učenja na seznamu primerov iz korpusa Šolar [8] [9]. Strojno učenje je uporabil za problem iskanja vseh vejic ter za problem popravljanja realnih napak v besedilu, torej za iskanje odvečnih vejic. V tem diplomskem delu želimo izboljšati rezultate strojnega učenja na isti množici primerov

tako, da generiramo nove atribute na podlagi slovničnih pravil. Razvili smo orodje za generiranje atributov z implementacijo pravil, ki jih za postavljanje manjkajočih vejic v tekstu uporablja LanguageTool [16] in dodali še dodatna pravila, ki jih narekuje Slovenski pravopis [6]. Rezultate želimo izboljšati tudi s spremembo atributov obstoječe množice tako, da odstranimo nekatere neinformativne atribute, nekatere druge pa modificiramo. Rezultate smo primerjali z dosedanjimi raziskavami. Preizkusili smo tudi izboljšan korpus Šolar, ki vsebuje približno trikrat več primerov z vejicami. Veliko primerov je bilo ročno pregledanih in popravljenih.

1.1 Pregled vsebine

Diplomsko delo smo razdelili na devet poglavij. Problem vejic v slovenščini smo podrobneje opisali v drugem poglavju. V tretjem poglavju smo si ogledali jezikovne tehnologije, ki jih uporabljamo za pripravo podatkov za nadaljnjo procesiranje. Algoritme strojnega učenja smo opisali v četrtem poglavju. Obstoječo množico podatkov, spremembe množice in pravila, s katerimi smo generirali nove atribute kot pomoč pri strojnem učenju, smo opisali v petem poglavju. V šestem poglavju prikažemo, kako mera za ocenjevanje atributov ReliefF oceni atribute originalne in spremenjene podatkovne množice. V sedmem poglavju opravimo testiranje, tako opišemo prečno preverjanje in podvzorčenje. V osmem poglavju analiziramo rezultate. Diplomsko delo sklenemo z devetim poglavjem, kjer pregledamo narejeno, naredimo zaključke in predlagamo nekaj možnosti izboljšav in idej za nadaljnje delo.

Poglavje 2

Problem vejic v slovenščini

Ločila so enodelna ali dvodelna nečrkovna pisna (grafična) znamenja [12]. Podrobneje si bomo ogledali vejico, ki zaradi veliko kompleksnih pravil, ki zahtevajo podrobno poznavanje hierarhije in razdelitve jezikovnih enot, piscem povzroča največ težav. Prikazali bomo njeno skladenjsko in neskladenjsko rabo ter ju za lažje razumevanje podkrepili s primeri. Ogledali si bomo najpogostejše težave pri pisanju vejic. Opisali bomo nekatera pravila in posebnosti pri pisanju vejic.

2.1 Na kratko o vejici

Vejica je enodelno levostično nekončno ločilo, ki spada pod skladenjsko rabo ločil, kar pomeni, da zaznamuje tonski potek, premore, vrste stavkov, povedi ipd. Rabi se bodisi na koncu povedi bodisi na koncu dela povedi. Vejico nekoliko redkeje rabimo tudi neskladenjsko. Ločila, ki spadajo v neskladenjsko rabo, nam zaznamujejo okrajšave besed in besedila, vrstilnost številnikov, kratnost prislovov zapisanih s številkami ipd. Vejica se neskladenjsko rabi v primerih, ko jo pri zapisu števil uporabljamo kot decimalno vejico ali namesto presledka pri označevanju milijonic, prav tako se neskladenjsko rabi pri zapisu imen in večbesednih poimenovanj, kadar nakazuje njihov spremenjen vrstni red [6]. Podrobneje si bomo skladenjsko in neskladenjsko rabo vejice

pogledali v nadaljevanju.

2.1.1 Skladenjska raba vejice

Skladenjska raba vejice se pojavlja:

- med enakovrednimi deli proste ali zložene povedi

  Teh delov je več vrst: nestavčni enakovredni deli, pristavčni, stavčni enakovredni ter nestavčno-stavčni enakovredni deli.

  Nestavčne enakovredne dele delimo na vezalne (to je naštevanje, npr. *Večja slovenska mesta so Ljubljana, Maribor, Celje, Kranj, Nova Gorica, Koper.*), stopnjevalne (npr. *Ne le ti, tudi tvoj brat naj pride.*), ločne (npr. *Bi mleko, sladkor?*), protivne (npr. *Staro, vendar zanimivo knjigo je našel.*), vzročne in posledične (npr. *Neizkušena, kajti mlada, je odšla v svet.*) ter pojasnjevalne dele (*Popoldanski delavnik je dvakrat na teden, in sicer v torek in petek.*). Zanimivejša posebnost pri nestavčnih enakovrednih delih je ta, da za levim pridevnikom ne pišemo vejice, kadar le-ta določa celo preostalo besedno zvezo (npr. *dolgi kodrasti lasje*).

  Pri pristavčnih delih velja pravilo, da je vejica na obeh straneh (npr. *Slovensko narodno gledališče, Opera in balet, ima na sporedu tri nove predstave.*)

  Podobno kot nestavčne enakovredne dele ločimo tudi stavčne enakovredne dele na vezalne (npr. *Na nebu zvezde sevajo, na vasi fantje pevajo.*), stopnjevalne (npr. *Ni niti upočasnil, kaj šele da bi ustavil.*), ločne (npr. *Zdaj je jokal, zdaj se je smejal.*), protivne (npr. *Vse ima, vendar mu ni dovolj.*), vzročne (npr. *Ne hodi po progi, je smrtno nevarno.*), posledične (npr. *Večerilo se je, zato so gnali črede domov.*) ter pojasnjevalne (npr. *Iz samostana je zvon zavabil, to molijo nune.*). Pojavljajo pa se tudi v bibliografskih zapisih pri naštevanju enot (npr. *Milko Kos, Srednjeveška Ljubljana, Topografski opis, 1955, 79 strani*).

Nestavčno stavčni enakovredni deli (npr. *Previdno, toda ne da bi se obotavljal, smo šli za njimi.*) imajo nekaj posebnosti. Naletimo na pravila, ki zapovedujejo, kdaj vejice **ne** pišemo. Kadar imamo skladenjsko enakovredne dele, vejice ne pišemo pred nekaterimi prirednimi vezniki, to so vezalni *in*, *pa*, *ter*, prav tako ločni *ali*, *oziroma*, *bodi(si)* in pred drugim delom dvodelnih vezniških zvez *ne - ne*, *niti - niti*, *tako - kakor*. Vendar pa pred vsemi temi vezniki pišemo vejico, kadar to zahteva vrinjeni ali vmesni stavek oz. dostavek. V nasprotju z ločnimi *ali* in stopnjevalnima *ne* oz. *niti* pa se vejica lahko piše pri naštevalnih *ali*, *ne* oz. *niti* [12].

- med nadrednim in odvisnim stavkom pišemo vejico, če je nadredni stavek pred odvisnim in prav tako v obratnem primeru, če je odvisni stavek pred nadrednim (npr. *Največja sreča za človeka je, če je zdrav. - Novica, da bomo kmalu rešeni, nam je vlila novih moči.*). Tudi tu naletimo na posebosti. Ena izmed nih je, da kadar beseda *da* nastopa kot členek, pred njim ne pišemo vejice. Prav tako vejico opuščamo, kadar skupaj nastopata dve podredni besedi (npr. *Le piši materi, da če ne bo denarja, ne boš več stanoval pri nas.*) Prav tako vejice ne pišemo med deli večbesednega veznika, taki vezniški izrazi so: *tako da*, *toliko da*, *potem ko*, *vtem ko*, *še ko*, *brž ko*, *šele ko*, *s tem da*, *posebno ko*, *zlasti če*, *namesto da*, *že ko*, *zato ker*, *medtem ko*, *zato da* ipd. [12].

- med polstavkom in drugim delom povedi, kjer je jedro polstavka lahko deležje, deležnik, pridevnik ali samostalnik (npr. *Želeč jim lahko noč, so se poslovili. - Ne misleč na posledice, je vse povedal. - Iz moje mladosti, polne ponižanja, se svetijo tiho tvoje oči.*). Posebno pozornost zahtevajo samostalniške zveze, nastale iz odvisnika, saj jih ne ločimo od okolja z vejico (npr. *Ob slovesni podelitvi bralnih značk osnovnošolcem je govoril tudi šolski ravnatelj.*). Vejico smemo napisati ali izpustiti, kar nam je ljubše, v primeru, ko lahko nedoločniški polstavek razvijemo v osebno glagolsko obliko (npr. *Postati zdravnik je njegova želja.* in prav

tako *Postati zdravnik, je njegova želja.*) [12].

- med izpostavkom in preostalim delom povedi (npr. *Sin, ta ti bo šele zagodel! - Vprašaj ga, vsevedneža!*) ter med dostavkom (npr. *Prinesi nam tudi kruha, črnega. - Pridi jutri, sam, točno.*) in preostalim delom povedi [12].

- med pastavki in drugim besedilom, in sicer pri zvalnikih in ogovorih (npr. *Gospa, kar po domače.*), pri samostojno rabljenih medmetih in medmetnih zvezah (npr. *Dober dan, kako ste kaj?*) ter pri samostojnih členkih (npr. *Da, vse je čisto. - Pač, še ena rešitev je. - No, pa vstopite!*) [12].

2.1.2 Neskladenjska raba vejice

Neskladenjsko se vejica rabi pri navajanju literature (npr. *Bajt, D. 1994: Pišem, torej sem. Maribor: Obzorja*), pri pisanju števil kot znamenje med celim številom in decimalkami (npr. *2,1456*) in pri pisanju datumov, kjer kraj in datum ločimo z vejico (npr. *Ljubljana, 16.4.2014*). Prav tako vejico neskladenjsko rabimo pri pisanju voznih redov (npr. *Litija 9:06, Zidani Most 9:40*), pozdravov (npr. *Spoštovana gospa Marjeta,...*) in podpisov (npr. *Lepo vas pozdravljam,...*) ter pri alinejnem naštevanju, kadar naštevanje zaključimo s piko. Med neskladenjsko rabo spada tudi pisanje vejic, kadar z njimi nakazujemo, da smo spremenili pravilno zaporedje sestavin, npr. pri abecednem seznamu (npr. *kruh, črni*) ter pri pisanju imen in večbesednih poimenovanj v seznamih ali stvarnih kazalih, kjer jim iz tehničnih (ali bibliografskih ipd.) spremenimo vrstni red (npr. *Molan, Ivan*) [12] [17].

2.2 Težave pri pisanju vejic

Pod okriljem projekta Sporazumevanje v slovenskem jeziku [4] je bila izvedena raziskava, kjer se je znotraj aktivnosti Slogovni priročnik [15] poskušalo zaznati najbolj pogoste zadrege uporabnikov pri pisanju besedil.

Izkazalo se je, da imajo uporabniki spleta težave pri postavljanju t.i. polstavčnih vejic (vejica v premem govoru, decimalna vejica, povezava vejice z drugimi ločili, tehnična raba vejice, vejica pri polstavkih in nedoločniških polstavkih) in pri pristavčnih vejicah, npr. pri pisanju dneva in datuma in ob nazivih ter imenih z okrajšavami.

Težave se pojavljajo pri vrinjenih stavkih, prav tako pri naštevanju različnih besednih enot (npr. vejica pri ponavljanju iste besede in vejica pred okrajšavami in izrazi, ki končujejo naštevanje) in pri pisanju vejice ob izpuščeni osebni glagolski obliki.

Zadrega se pogosto pojavi pri pisanju vejic v različnih stavčnih razmerjih (povezovanje stavčnega člena in istovrstnega odvisnika in pri povezovanju enakovrednih stavkov, neenakovrednih stavkov) ter kadar uporabniki naletijo na veznike ter besede v vezniški vlogi.

Veliko napak se pojavlja pri vejici med večbesednimi vezniki in vezniškimi izrazi (vejica med večbesednimi vezniki in vezniškimi izrazi ter pomensko-razlikovalna vejica med deli večbesednega veznika) ter pri drugih posebnih izrazih. Podrobna analiza zadreg uporabnikov presega temo te diplomske naloge in je dosegljiva na spletni strani portala Slogovni priročnik [15].

Znotraj zadreg pri vejici in veznikih ter besedah v vezniški vlogi so najpogostejše zadrege pri vezniku a, podrednem in prav tako prirednem vezniku ali, prav tako pri veznikih in ne, in sicer, in tako, in to, in tudi, (in) torej, in zato, ki, kot/kakor, na primer (da), niti, od ... do, oziroma, pa tudi, razen, tako ... kot (tudi), tako da, ter, toliko in zlasti.

Nekatere izmed teh veznikov oz. besed v vezniški vlogi bomo opisali (in implementirali) v poglavju 5.2, saj jih za pravilno postavljanje vejic uporablja orodje LanguageTool, nekatere ostale pa v poglavju 5.3, kjer si bomo podrobneje ogledali implementacijo nekaterih pravil iz Slovenskega pravopisa.

Poglavje 3

Jezikovne tehnologije

Obdelava naravnega jezika je interdisciplinarno področje, pri katerem je potrebno znanje računalniških strokovnjakov in jezikoslovcev, a tudi matematikov, filozofov, psiholingvistov in nevrologov [14]. Pri obdelavi naravnega jezika si želimo, da bi računalnik besedilo razumel podobno kot človek, zato potrebujemo tehnologije, s katerimi pridobimo čim več podatkov. Zanimajo nas besedne vrste, vloge besed v stavku itd. V tem poglavju bomo opisali postopke označevanja, razčlenjevanja in lematizacije besed. Ogledali si bomo besedilne korpuse, ki služijo kot temeljna infrastruktura pri raziskovanju in obdelavi naravnih jezikov.

3.1 Označevanje

Oblikoslovno označevanje je postopek, s katerim besedam v besedilu določimo njihove skladenjske oznake, npr. samostalnik, glagol, predmet, medmet itd. Ker v slovenskem jeziku poleg dvojine in samostalnikov v šestih sklonih in treh spolih poznamo tudi pridevnike, ki poleg sklona, spola in števila lahko izražajo tudi stopnjo ter določnost, je oblikoslovno označevanje slovenščine zahteven postopek, saj se mora označevalnik odločati med skoraj 2000 slovničnimi oznakami. Poleg naštetega je v slovenskem jeziku veliko dvoumnih besed (npr. klòp - majhen zajedalec, klóp - lesena, kamnita pri-

prava za sedenje, klop - klopčič), prostega besednega vrstnega reda (stavčne člene je mogoče najti takorekoč na vseh mestih v stavku, z njimi pa poudarjamo različne elemente [14]) in množica različnih besednih oblik.

Projekt JOS je z namenom spodbujanja razvoja jezikovnih tehnologij za slovenski jezik razvil označene korpuse slovenskega jezika (jos100k in jos1M), priporočila za oblikoslovno označevanje (oblikoskladenjske specifikacije), dva označena korpusa in dva spletna servisa [3]. Naloga priporočil za oblikoslovno označevanje je za vsako besedno vrsto definirati kategorijo z njenimi atributi in vrednostmi. Tako vsako besedo razvrsti v eno izmed opcij: samostalnik, glagol, pridevnik, prislov, zaimek, števnik, predlog, veznik, členek, medmet, okrajšava, neuvrščeno.

S pomočjo označevalnika smo za vsako besedo iz uporabljenega korpusa dobili njeno MSD kodo oz. oblikoskladenjsko oznako. Oznako uporabimo kot atribut oz. več atributov za strojno učenje (podrobneje bomo atribute predstavili v 5.1.1). Atributi označujejo tudi besede pred in za trenutno besedo. Uporabljamo privzeto velikost okna, to je pet besed pred trenutno in pet besed za trenutno, zato smo s pomočjo označevalnika dobili vrednosti enajstih MSD kod (za trenutno besedo ter za vse besede znotraj okoliškega okna).

Kot bomo opisali v nadaljevanju, prva črka MSD kode določa besedno vrsto trenutne besede, vse morebitne nadaljnje črke pa njene lastnosti. Najkrajši je zapis členkov, medmetov in okrajšav, saj so opisani le z eno črko. Videli bomo, da V določa veznik, D predlog in L členek. Ostale črke, ki določajo besedne vrste, so še S za samostalnik, G za glagol, P za pridevnik, R za prislov, Z za zaimek, K za števnik, M za medmet, O za okrajšavo ter N za neuvrščeno.

Ogledali si bomo kategorije za predlog, veznik in členek, ki jih potrebujemo pri implementaciji pravil v poglavju 5.2 in 5.3, opisi ostalih kategorij pa so dosegljivi na [3].

3.1.1 Veznik

Kadar oblikoskladenjski označevalnik za slovenščino naleti na veznik, ga označi s črko V, doda pa mu še oznako v, če je veznik priredni in d, če je veznik podredni. Tako je npr. za veznik in, kadar je uporabljen priredno, njegova oblikoslovna oznaka enaka Vp, za podredni veznik *čeprav* pa se uporabi oznaka Vd.

oznaka	**besedna vrsta**	**lastnosti**	**primeri**
Vp	veznik	priredni	in, pa, ali, saj, ter, zato, vendar, a, namreč, oziroma
Vd	veznik	podredni	da, ki, kot, ko, če, ker, kjer, čeprav, kakor, naj

Tabela 3.1: Oznake za veznik po priporočilih za oblikoslovno označevanje JOS [2].

3.1.2 Členek

Oblikoskladenjski označevalnik dodeli zaznanemu členku kodo L.

oznaka	**besedna vrsta**	**lastnosti**	**primeri**
L	členek	/	tudi, ne, še, že, le, naj, samo, prav, seveda, predvsem

Tabela 3.2: Oznake za členek po priporočilih za oblikoslovno označevanje JOS [2].

3.1.3 Predlog

Oblikoskladenjski označevalnik predlog označi z dvema črkama. Prva je D, druga pa ena izmed črk *i* ,*r*, *d*, *t*, *m* ali *o*, odvisno od tega, kateri sklon določa

samostalniški in/ali pridevniški besedi, s katero nastopa v povedi. Če določa imenovalnik, je druga črka kode *i*, za rodilnik *r*, dajalnik *d*, tožilnik *t*, mestnik *m* in orodnik *o*. Nekateri predlogi imajo skupno lemo s prislovi, to so: *čez*, *blizu*, *glede*, *križem*, *mimo*, *nasproti*, *okoli*, *okrog*, *poleg*, *povrh*, *povrhu*, *proti*, *skozi*, *širom*, *tostran*, *vštric*, *za*, *znotraj*, *zraven* in *zunaj* [3]. V teh primerih moramo biti pozorni na morebitne napake pri avtomatskem označevanju.

oznaka	**besedna vrsta**	**lastnosti**	**primeri**
Di	predlog	imenovalnik	po
Dr	predlog	rodilnik	iz, od, do, zaradi, brez, s/z, poleg, glede, prek
Dd	predlog	dajalnik	k, proti, kljub, h/k, navkljub, blizu, navzlic, nasproti
Dt	predlog	tožilnik	za, na, v, čez, po, skozi, med, pod, zoper, ob
Dm	predlog	mestnik	v, na, po, pri, o, ob, nad, pod
Do	predlog	orodnik	s/z, med, pred, pod, za, nad

Tabela 3.3: Oznake za predlog po priporočilih za oblikoslovno označevanje JOS [3].

3.2 Razčlenjevanje

Skladenjsko razčlenjevanje je eden izmed temeljnih postopkov analize besedil, ki omogoča razvoj kompleksnejših jezikovnih tehnologij, npr. prepoznavo govora, strojno prevajanje, avtomatsko povzemanje razprav itd. Z njim besedam pripišemo skladenjska razmerja glede na druge besede v povedi. Korpus, ki smo ga uporabili, je bil skladenjsko razčlenjen s skladenjskim razčlenjevalnikom Obeliks, ki je prosto dostopen na [2]. To je računalniški program, ki za avtomatizirano skladenjsko razčlenjevanje uporablja sistem odvisnostnih drevesnic, razvit v okviru projekta JOS [2]. Ta skladenjske

odnose v povedi predstavi z drevesno strukturo, kjer lahko vsaka povezava zavzame enega izmed desetih tipov povezav, razdeljenih v tri skupine povezav. Prva skupina so povezave prvega nivoja in označujejo razmerja znotraj besednih zvez. V to skupino spadajo tipi, ki povezujejo jedro in določilo besednih zvez, dele zloženega povedka, jedra v prirednih zvezah znotraj stavka, besede ali ločila v vezniški vlogi in nepolnopomenske besede, ki imajo močno tendenco po sopojavljanju. Druga skupina povezav so povezave drugega nivoja in označujejo stavčne člene. Tu naletimo na štiri tipe povezav: osebek stavka, predmet stavka, prislovno določilo lastnosti ter ostala prislovna določila. Tretja skupina povezav se uporablja za povezovanje vseh ostalih struktur. To so hierarhično najvišje pojavnice, skladenjsko manj predvidljive in oddaljene strukture, vrinki ter ločila [2]. Povezavam drugega nivoja bomo rekli tudi rdeče povezave, povezavam tretjega pa modre povezave.

S pomočjo skladenjskega razčlenjevalnika za vsako besedo dobimo vrednosti treh atributov za povezave trenutne besede in po tri atribute za vsako besedo znotraj okoliškega okna. Z našim okoliškim oknom tako dobimo vrednosti za 33 atributov. Vsak trojček atributov nam pove, ali obstaja povezava prvega, drugega in tretjega nivoja (atribut dobi vrednost 1) ali ne (atribut dobi vrednost 0). Atribute podrobneje opišemo v poglavju 5.1.1.

3.3 Lematizacija

Lematizacija (tudi *geslenje*) je proces, s katerim besede, zapisane v različnih sklonih, oblikah in spolih nadomestimo z njihovimi osnovnimi (slovarskimi) oblikami. Osnovna oblika besede se imenuje lema. Vsem različnim skladenjskim oblikam neke besede pripada ista lema. Npr. besedam *hodim*, *hodiš*, *hodi*, *hodiva* itd. pripada lema *hoditi*.

V jezikih, ki niso tako morfološko bogati kot slovenščina, se namesto lematizacije pogosto uporablja krnjenje besed. To je postopek, s katerim besedi, ki ni v osnovni obliki, odrežemo končnico, da ostane le krn besede. Pri obdelavi slovenščine se priporoča lematizacija, saj jezik vsebuje veliko

besed, kjer krn besede ne poda dovolj informacije o besedi in s tem otežuje nadaljnjo obdelavo.

Kot bomo videli v poglavju 5.1.1, s pomočjo lematizacije dobimo vrednosti enajstih atributov, ki jih uporabimo kot pomoč pri strojnem učenju. Prvi atribut nam pove lemo trenutne besede, s privzetim okoliškim oknom pa dobimo še vrednosti desetih atributov, ki nam povedo leme petih predhodnjih in petih naslednjih besed v stavku.

Poglavje 4

Opis algoritmov strojnega učenja

Opišemo algoritme, ki jih uporabljamo za iskanje čimbolj ustrezne hipoteze za dano predznanje in vhodne podatke. Na izboljšani podatkovni množici uporabimo algoritme, ki so že bili uporabljeni na osnovni podatkovni množici v predhodni raziskavi ter uporabimo dva nova algoritma, to sta algoritem naključni gozdovi ter metoda podpornih vektorjev. Opišemo metodo naključnih gozdov (Random Forest), ki je ansambel odločitvenih dreves, in metodo podpornih vektorjev (SVM). Opišemo algoritme naivni Bayesov klasifikator, RBF mreža, alternirajoče odločitveno drevo, AdaBoostM1 ter odločitvena tabela.

4.1 Naključni gozdovi

Metodo naključnih gozdov je razvil Leo Breiman [18] in se uporablja za izboljšanje napovedne točnosti drevesnih modelov. Ta je primerljiva z najboljšimi algoritmi kot sta boosting in SVM. Slaba stran naključnih gozdov je, da je zaradi množice sto in več dreves razlaga odločitev otežena, zato so potrebne posebne vizualizacijske tehnike.

4.1.1 Osnovni princip metode naključnih gozdov

Posamezen gozd sestoji iz množice navadnih odločitvenih dreves. Ta v osnovi v vozliščih izbirajo najboljši atribut iz množice atributov. Ko gradimo naključna drevesa, običajna odločitvena drevesa nadgradimo tako, da jim dodajamo več naključnosti. To implementiramo tako, da v vozliščih izbiramo najboljši atribut iz naključno izbrane podmnožice vseh atributov [13]. Učne primere za posamezna drevesa izbiramo naključno s strmenskim vzorčenjem. S tem iz navadnih odločitvenih dreves ustvarimo različna (naključna) drevesa in zato dobimo robusten in zanesljiv klasifikator. Klasificiranje novih primerov poteka tako, da ima vsako izmed dreves za vsak primer na voljo en glas. Nameni ga razredu, kamor bi klasificiralo trenutni primer. Iz množice vseh glasov dobimo verjetnostno distribucijo po vseh razredih [11].

4.2 Metoda podpornih vektorjev

Lastnost metode podpornih vektorjev (SVM) je, da za razliko od večine algoritmov, ki težijo k čim manjšemu naboru pomembnih atributov, pri tej metodi vključimo vse razpoložljive atribute in jih uporabimo za napovedovanje odvisne spremenljivke. Pomemben je predvsem način kombiniranja atributov in ne izbira atributov. Za metodo podpornih vektorjev se odločimo, ker dosega visoko točnost napovedi in je primerna za učenje na velikih množicah primerov z velikim številom atributov, prav tako pa so namenjene razločevanju dveh razredov (v našem primeru je-vejica, ni-vejica). Slabi strani metode podpornih vektorjev sta težavna interpretacija naučenega in razlaga posamezne odločitve [11].

4.2.1 Osnovni princip metode podpornih vektorjev

V danem prostoru atributov iščemo optimalno hiperravnino, ki bo med seboj ločila oba razreda (je-vejica, ni-vejica). Optimalna hiperravnina je takšna hiperravnina, ki je od najbližjih primerov vsakega razreda oddaljena z enako

in maksimalno razdaljo. Prav tako mora upoštevati omejitve, ki jih vsak učni primer zahteva za svojo pravilno klasifikacijo. Tistim primerom, ki so najbližji optimalni hiperravnini, pravimo podporni vektorji, njihovo število pa določa kompleksnost rešitve. V praksi se izkaže, da je teh vektorjev od 3-5% vseh učnih primerov, kar pomeni, da je kompleksnost rešitve relativno majhna [11].

4.3 Ostali algoritmi

Naivni Bayesov klasifikator

Naivni Bayesov klasifikator je enostaven verjetnostni klasifikator, ki predpostavlja, da so si atributi med seboj pogojno neodvisni.

RBF mreža

RBF mreža (ang. Radial Basis function network, okrajšava RBFNetwork) je nevronska mreža, ki klasificira glede na razdaljo primerov do središč Gaussovskih jeder, ki jih vsebujejo nevroni.

Alternirajoče odločitveno drevo

Alternirajoče odločitveno drevo (ang. Alternating decision tree, okrajšava ADTree) je posebne vrste odločitveno drevo, ki implementira *boosting*.

AdaBoostM1

AdaBoostM1 je za klasifikacijo zasnovana metoda, ki uporablja idejo *boosting*. Primerkom določi uteži, s katerimi jim določi pomembnost. Primeri z večjimi utežmi postanejo pomembnejši in s tem prioritetni za pravilno klasifikacijo [10].

Odločitvena tabela

Odločitvena tabela (ang. Decision Table) primerja podniz primera za klasifikacijo z metodo najbližjih sosedov in ga razvrsti v isti razred kot njemu najbližje primere [10].

Poglavje 5

Opis podatkov

Ogledamo si množico podatkov, iz katere smo izhajali pri naši analizi. Opišemo korake, ki so bili potrebni, da smo iz korpusa dobili format, ki je bil ustrezen za nadaljnje analize. Podrobno opišemo sestavo formata, v katerem so informacije o besedah in njenih atributih. Navedemo atribute, ki smo jih imeli na voljo, ter informacijo, ki jo vsebujejo. Opišemo postopek dodajanja novih atributov na podlagi slovničnih pravil. Podrobneje si ogledamo pravila, ki jih uporablja orodje LanguageTool za postavljanje vejic, in kako smo iz njih generirali atribute v pomoč strojnemu učenju. Opišemo še nekaj dodatnih implementacij pravil za vejice, ki piscem besedil pogosto povzročajo težave. Opišemo pomanjkljivosti obstoječe množice podatkov in naše popravke.

5.1 Obstoječa množica podatkov

Izhajali smo iz množice podatkov, ki je posodobljena verzija korpusa uporabljenega v [8], in sicer podkorpus z napačno postavljenimi vejicami. Velikost korpusa je 209.156 besed, v njem je 11.892 pravilno postavljenih vejic ter 11.399 manjkajočih. Korpus je bil predelan v korpus s pravilno postavljenimi vejicami [8]. Korpus je bil oblikosladenjsko označen, lematiziran in skladenjsko razčlenjen. Korpus je bil označen z oblikoslovnim označevalnikom in lematizatorjem Obeliks [17] ter skladenjsko razčlenjen s

skladenjskim razčlenjevalnikom [1], oba sta bila razvita pod okriljem projekta Sporazumevanje v slovenskem jeziku [4]. Vsaka beseda z okoliškim oknom, ki se pojavi v korpusu, je pretvorjena v seznam atributov, dodan pa je bil tudi atribut, ali tej besedi sledi vejica [9]. Rezultat označevanja je v formatu ARFF, ki je sestavljen iz glave ter podatkovnega dela in je namenjen orodju WEKA [7]. 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 atributno opisana beseda.

5.1.1 Atributi

Poskuse smo opravili na treh različno generiranih datotekah, v vseh pa so besede zapisane z enakim številom atributov. Vse so vsebovale celoten MSD (opisano v poglavju 3.1) ter atribute s podatki o skladnji. Prva datoteka je bila oblikoslovno označena ter skladenjsko razčlenjena z uporabo besedila s pravilno postavljenimi vejicami. Ker to ni realna situacija, saj v realnih primerih vnaprej ne vemo, kje morajo stati vejice, je bila druga datoteka generirana z uporabo besedila, ki je bilo označeno brez vejic. Tretja datoteka je bila ustvarjena z uporabo učnega korpusa brez vejic, iz katerega so bili naučeni lematizator, oblikoslovni označevalnik in skladenjski razčlenjevalnik [9]. Kako je označevanje in razčlenjevanje z vejicami oz. brez njih vplivalo na uspešnost testiranja, smo preizkusili v poglavju 7.3. V nadaljevanju bomo opisali atribute, ki smo jih definirali za vsako besedo.

Prvi atribut je beseda, na kakršno naletimo v besedilu, ne da bi jo kakorkoli spreminjali. Ta atribut lahko zavzame toliko vrednosti, kolikor je različnih besed in njenih oblik v besedilu, npr. *človek* in *človeka* zasedeta dve mesti v zalogi vrednosti tega atributa. Naslednji atribut je lema trenutne besede (glej poglavje 3.3). Tudi zaloga teh vrednosti je lahko zelo velika, sestavljajo pa jo vse leme nastopajočih besed v besedilu. Tukaj *človek* in *človeka* zasedeta eno vrednost in sicer z osnovno obliko besede, *človek*. Sledi zapis MSD kode besede, zaloga vrednosti pa zavzame vse možne oblikoskladenjske oznake po oblikoskladenjskih specifikacijah JOS [3], s katero besedi

določimo besedno vrsto, sklon, spol, število itd [9]. Naslednji trije atributi so binarni (lahko zasedejo eno izmed vrednosti 0 ali 1) ter nam sporočajo, ali trenutna beseda kaže na del povedi (obstaja povezava med trenutno besedo in neko drugo besedo v povedi), ali obstaja povezava bodisi na osebke, predmete ali prislovna določila (t.i. modra povezava) in ali trenutna beseda kaže na veznike (t.i. rdeča povezava). Vseh šest atributov ponovimo za vse besede znotraj okoliškega okna. Za analizo uporabljamo privzeto okoliško okno, tj. pet besed pred in pet besed za trenutno besedo. To pomeni, da sledi petkrat po šest zgoraj opisanih atributov za pet besed pred trenutno besedo in petkrat po šest istih atributov za pet naslednjih besed od trenutne. Tako imamo za vsako besedo 66 atributov, sledi pa jim še razred, ki pove, ali besedi sledi vejica. Zavzema lahko dve vrednosti [je-vejica, ni-vejica]. Opisanim 67 atributom, ki so bili uporabljeni v [8], smo dodali še 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.

Podrobneje si atribute ogledamo v tabeli 5.1. V njej imamo podane atribute za besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje*. Prvi stolpec nam pove zaporedno številko atributa. Ker pri programiranju naslavljamo od 0, do posameznega atributa dostopamo s klicem atributa na položaju *zap.št*-1. Drugi stolpec poda opis atributa in tretji stolpec vrednost atributa za zgoraj navedeno besedo z njenim okoliškim oknom. Znak * pomeni, da beseda ne obstaja in z njo tudi ne vrednosti atributov za to besedo.

Zap. št.	Opis atributa	Primer
1	trenutna beseda	*vidno*
2	lema trenutne besede	*vidno*
3	MSD koda trenutne besede	*Rsn*
4	obstaja povezava na trenutno besedo	*1*
5	obstaja modra povezava na trenutno besedo	*0*
6	obstaja rdeča povezava na trenutno besedo	*0*
7	prejšnja beseda	*že*
8	lema prejšnje besede	*že*
9	MSD koda prejšnje besede	*L*
10	obstaja povezava na prejšnjo besedo	*1*
11	obstaja modra povezava na prejšnjo besedo	*0*
12	obstaja rdeča povezava na prejšnjo besedo	*0*
13	predprejšnja beseda	*je*
14	lema predprejšnje besede	*biti*
15	MSD koda predprejšnje besede	*Gp-ste-n*
16	obstaja povezava na predprejšnjo besedo	*0*
17	obstaja modra povezava na predprejšnjo besedo	*0*
18	obstaja rdeča povezava na predprejšnjo besedo	*0*
19	beseda tri mesta pred trenutno	*Tu*
20	lema besede tri mesta pred trenutno	*tu*
21	MSD koda besede na položaju tri mesta pred trenutno	*Rsn*
22	obstaja povezava na besedo tri mesta pred trenutno	*1*
23	obstaja modra povezava na besedo tri mesta pred trenutno	*0*
24	obstaja rdeča povezava na besedo tri mesta pred trenutno	*0*
25	beseda štiri mesta pred trenutno	*
26	lema besede štiri mesta pred trenutno	*
27	MSD koda besede štiri mesta pred trenutno	*
28	obstaja povezava na besedo štiri mesta pred trenutno	*

29	obstaja modra povezava na besedo štiri mesta pred trenutno	*
30	obstaja rdeča povezava na besedo štiri mesta pred trenutno	*
31	beseda pet mest pred trenutno	*
32	lema besede pet mest pred trenutno	*
33	MSD koda besede pet mest pred trenutno	*
34	obstaja povezava na besedo pet mest pred trenutno	*
35	obstaja modra povezava na besedo pet mest pred trenutno	*
36	obstaja rdeča povezava na besedo pet mest pred trenutno	*
37	beseda eno mesto za trenutno	*kako*
38	lema besede eno mesto za trenutno	*kako*
39	MSD koda besede eno mesto za trenutno	*Rsn*
40	obstaja povezava na besedo eno mesto za trenutno	*1*
41	obstaja modra povezava na besedo eno mesto za trenutno	*1*
42	obstaja rdeča povezava na besedo eno mesto za trenutno	*0*
43	beseda dve mesti za trenutno	*odločno*
44	lema besede dve mesti za trenutno	*odločno*
45	MSD koda besede dve mesti za trenutno	*Rsn*
46	obstaja povezava na besedo dve mesti za trenutno	*0*
47	obstaja modra povezava na besedo dve mesti za trenutno	*0*
48	obstaja rdeča povezava na besedo dve mesti za trenutno	*0*
49	beseda tri mesta za trenutno	*je*
50	lema besede tri mesta za trenutno	*biti*
51	MSD koda besede tri mesta za trenutno	*Gp-ste-n*
52	obstaja povezava na besedo tri mesta za trenutno	*0*
53	obstaja modra povezava na besedo tri mesta za trenutno	*0*
54	obstaja rdeča povezava na besedo tri mesta za trenutno	*0*

55	beseda štiri mesta za trenutno	*prepričana*
56	lema besede štiri mesta za trenutno	*prepričan*
57	MSD koda besede štiri mesta za trenutno	*Pdnzei*
58	obstaja povezava na besedo štiri mesta za trenutno	*0*
59	obstaja modra povezava na besedo štiri mesta za trenutno	*0*
60	obstaja rdeča povezava na besedo štiri mesta za trenutno	*0*
61	beseda pet mest za trenutno	*v*
62	lema besede pet mest za trenutno	*v*
63	MSD koda besede pet mest za trenutno	*Dt*
64	obstaja povezava na besedo pet mest za trenutno	*0*
65	obstaja modra povezava na besedo pet mest za trenutno	*1*
66	obstaja rdeča povezava na besedo pet mest za trenutno	*0*
67	razredni atribut, ki pove ali trenutni besedi sledi vejica	*je-vejica*

Tabela 5.1: Vrednosti atributov za besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje.*

5.2 Implementacija pravil iz orodja LanguageTool

LanguageTool je odprtokodni program, ki zna poleg preverjanja sloga in slovnice besedila tudi postavljati vejice s pomočjo ročno napisanih pravil. Napačno postavljenih vejic zaenkrat ne popravlja. Podpira množico jezikov, med njimi slovenščino, angleščino, nemščino, francoščino, poljščino itd. Za problem postavljanja vejic v slovenščini ima implementiranih 41 pravil, ki so zapisana v formatu XML [8]. Vseh 41 pravil smo uporabili v našem programu in z njimi generirali nove atribute v pomoč strojnemu učenju tako, da smo dodali 41 atributov (za vsako pravilo en atribut), ki lahko zasedejo vrednosti 0 ali 1. Vrednost smo atributu nastavili na podlagi pravil. V nadaljevanju opišemo pravila in njihovo implementacijo.

5.2.1 Ampak, saj, kajne, kajti, kot, toda, ampak tudi

Pravilo za veznike oz. vezniške besede[1] *ampak*, *saj*, *kajne*, *kajti*, *kot* in *toda* je v programu LanguageTool implementirano sledeče: če v besedilu pred omenjenim veznikom oz. vezniško besedo ni ločil *,(;–:*, potem pred tem veznikom verjetno manjka vejica. To pravilo smo realizirali v program tako, da kadar smo naleteli na omenjeni veznik in je bil ta veznik neprva beseda v stavku, smo v primeru, da prejšnja beseda ni bila ločilo, prejšnji besedi dodali atribut, ki lahko zavzame vrednosti [0,1], in ga nastavili na 1.

Za *ampak tudi* velja enak postopek kot za *ampak*, le da med ločili upoštevamo še -, trenutna beseda mora biti enaka *ampak*, naslednja beseda od trenutne pa *tudi*. Preveriti moramo tudi, da trenutna beseda ni prva ali zadnja v stavku, kajti kot prva na besedi pred njo ne more biti vejice. Tudi kot zadnja ne sme biti, saj ji mora slediti še vsaj *tudi*, v realnosti pa še najmanj ena beseda (ki ni pika).

[1] vezniška beseda - *beseda, ki kaže na razmerje med dvema deloma sporočila[19]*

5.2.2 Temveč tudi, tj.

Pri vezniških izrazih *temveč tudi* ter *tj.* velja enako pravilo kot zgoraj (pred vezniškim izrazom ne sme stati katero izmed ločil *,(;-:*), le da smo ga tu realizirali nekoliko drugače. Pozorni smo bili na besedo pred prejšnjo, ta ni smela biti enaka zgoraj naštetim ločilom, prejšnja beseda pa je morala zadostovati pogoju, da je enaka *temveč* oz. *tj*, trenutna pa besedi *tudi* oz. .(piki). Kot smo opisali v poglavju 5.1.1, je vejica atribut na besedi pred vejico, ostala ločila pa obravnavamo enako kot besede.

Velja poudariti, da kadarkoli govorimo o besedi, imamo v mislih njeno lemo, torej podatek o osnovni slovarski obliki besede, ki se nahaja v datoteki s končnico ARFF na drugem mestu vrstice, če je ena vrstica podatek za eno besedo in to tudi upoštevamo pri generiranju atributov. Podrobno sestavo datoteke opisujemo v razdelku 5.1.1.

5.2.3 Kam, kdaj, kjer, vse dokler, zaradi

Podobno kot zgoraj realiziramo tudi pravila za vezniške besede *kam*, *kdaj*, *kjer*, *vse dokler* in *zaradi*. Pri dvobesedni vezniški besedi *vse dokler* smo zopet pozorni na besedo pred prejšnjo, pravilo za vse naštete vezniške izraze pa pravi: če v besedilu najdemo eno izmed zgoraj naštetih vezniških besed, pred katero ni ločil *,(;–:* in tudi ne besed *in*, *ali* in *ter*, potem je velika verjetnost, da mora biti na besedi pred to vezniško besedo (oz. na besedi pred prejšnjo, če imamo vezniško besedo, ki je dolga dve besedi) atribut nastavljen na 1 (je vejica).

5.2.4 Predlog in kaj, predlog in kar, predlog in kateri

Pred vezniškimi izrazi *predlog+kaj*, *predlog+kar* ter *predlog+kateri* ne smejo stati ločila *,(;-:*, pri prvih dveh tudi ne besede *in*, *ali*, *ter*, pri zadnjem pa poleg tega preverjamo še, da pred njim ne stoji množica besed: *brez*, *v*, *na*, *pred*, *nad*, *pri*, *ob*, *s*, *za*, *po*, *iz*, *o*, *skozi*, *pod*, *izza*, *okoli*, *vzdolž*, *do*, *proti*, *med*, *brez*, *od* in *zaradi*.

Vse naštete dvodelne vezniške izraze, ki kot drugo besedo vsebujejo predlog, obravnavamo podobno kot zgoraj. Predlog pri prvem delu vezniških izrazov predlog+*kaj*, predlog+*kar* in tudi predlog+*kateri* lahko zavzema vrednosti *v*, *na*, *pred*, *nad*, *ob*, *za*, *skozi*, *pod...* Da nam ni potrebno preverjati vseh možnosti, ki se lahko pojavijo kot drugi del teh dvodelnih vezniških izrazov, smo uporabili naslednji pogoj: pri vezniški besedi, kjer imamo opravka s predlogom na prvem mestu, pogledamo drugi podatek v vrstici za prejšnjo besedo, ki nam sporoča MSD kodo in če je ta enaka *Di*, *Dr*, *Dd*, *Dt*, *Dm* ali *Do* (oblikoslovne oznake za predloge v različnih sklonih, opisano v poglavju 3.1) za to besedo, potem prejšnja beseda spada med predloge in naš pogoj je, seveda v kombinaciji z zgoraj omenjenimi pogoji, izpolnjen. V takšnem primeru verjetno manjka vejica, zato atribut za ta veznik nastavimo na vrednost 1.

5.2.5 To je, zatorej, ko, kolikor

Podobno dodajamo atribute tudi za vezniška izraza *to je* in *zatorej*. Pri *to je* smo pozorni na predprejšnjo besedo in tudi na to, ali je izpolnjen pogoj, da je trenutna beseda *je* in prejšnja enaka *to*, pri *zatorej* pa smo pozorni na prejšnjo besedo. Pri obeh velja, da predprejšnja (pri *to je*) oz. prejšnja (pri *zatorej*) ne sme biti enaka ločilom *,(;-:* ali vezniku *in*. Če je pogoj izpolnjen, storimo tako kot doslej: atributoma, ki ustrezata pogoju za vezniška izraz *to je* in *zatorej* nastavimo na 1, v nasprotnem primeru pa na 0.

Tudi kadar naletimo na izraza *ko* ter *kolikor*, ravnamo podobno, le da smo tu pozorni na nekoliko več besed, ki ne smejo stati pred njima. Za oba velja, da beseda pred njima ne sme biti enaka ločilom *,(;-:*, pri *ko* dodatno ne sme biti ujemanja med prejšnjo besedo in besedami *in*, *ali*, *ter*, *le*, *samo*, *tudi*, *potem*, *zlasti*, *celo*, *medtem*, *ampak*, *a*, *vendar*, *toda*, *vsakič* in *takoj*, pri vezniški besedi *kolikor* pa prejšnja beseda ne sme biti enaka besedam *in*, *ali*, *ter*, *le*, *samo*, *tudi*, *potem*, *zlasti*, *celo*, *razen*, *v* in *na*.

5.2.6 Ki, ne da bi, kakor, kamor, kakšen, kakršen

Vezniška izraza *ki* in *ne da bi* si v programu LanguageTool delita pravilo. Beseda pred trenutnim veznikom ne sme biti enaka besedam *in*, *ali*, *ter* in pred veznikom ne sme biti ločil *,(;–:*. Dodatno pred *ne da bi* ne sme stati spodnji ali zgornji narekovaj, pred *ki* pa pravilo pravi, da ne sme biti tropičja. *Ne da bi* ima tri besede, zato moramo biti pozorni, da ne preverjamo prejšnje besede, ampak dve pred njo. Če je trenutna beseda *bi*, mora biti prejšnja enaka *da* in predprejšnja enaka *ne*. Ena beseda pred to pa mora ustrezati pravilu. Če je to izpolnjeno, ima ta atribut vrednost 1, v nasprotnem primeru pa je njegova vrednost 0.

Tudi pred *kakor* in *kamor* ne sme biti ločil *,(;–:*, prav tako ne besed *in*, *ali*, *ter*, *le*, *samo*, *tudi*, *potem*, *zlasti*, *celo* in *razen*. Če so pogoji izpolnjeni, lahko sklepamo, da pred njima verjetno stoji vejica, zato atribut dobi vrednost 1.

Za *kakšen* in *kakršen* obstaja podoben pogoj. Da je pogoj *je-vejica* (atribut nastavljen na 1) na besedi pred veznikom izpolnjen, ta beseda ne sme biti ločilo ali enaka besedam *in*, *ali*, *ter*, *za*, *po*, *ob*, *na*, *pred*, *nad*, *pri*, *s*, *v* in *iz*. Tako kadar naletimo na besedo *kakšen* in prejšnja beseda ustreza napisanemu pogoju, atribut za ta pogoj dobi vrednost 1, v nasprotnem primeru 0 in prav tako pravilo velja tudi, kadar naletimo na besedo *kakršen*. Če beseda ni enaka besedi *kakršen*, trenutni besedi atribut za ta veznik nastavimo vrednost 0. Prav tako ta atribut nastavimo na 0, če je trenutna beseda enaka besedi *kakršen*, vendar pa ne zadostuje pogoju.

5.2.7 Kadar, kaj, kar, ker, kje, dokler, kako, preden

Pri vezniških izrazih *kadar*, *kaj* in *kar* smo zopet pozorni na prejšnjo besedo, ki ne sme biti katero izmed ločil *,(;–:*. Dodatno pri *kadar* prejšnja beseda ne sme biti enaka besedam *in*, *ali*, *ter*, *le*, *samo*, *tudi*, *potem*, *zlasti*, *celo* in *razen*, pri vezniškem izrazu *kaj* besedam *že*, *še*, *v*, *na*, *pred*, *nad*, *ob*, *za*, *skozi*, *pod*, *in*, *ali*, *nič* in *ter*, pri vezniškem izrazu *kar* pa besedam *že*, *še*, *v*, *na*, *pred*, *na*d, *ob*, *za*, *skozi*, *pod*, *in*, *a*, *ali* in *ter*.

Kadar se beseda ujema s katerim izmed veznikom *ker*, *kje*, *dokler*, *kako* in *preden*, se prejšnja beseda ne sme ujemati z ločili *,(;-:* ter besedami *in*, *ali*, *ter*. Dodatno pri *ker* prejšnja beseda ne sme biti enaka *a*, *temveč*, *ampak*, *vendar* in *toda*. Pri *kje* ne sme biti ujemanja z besedo *iz*, pri *dokler* z besedo *vse*, pri *kako* z besedama *še* in *a* ter pri *preden* z besedami *tik*, *še* in *a*. 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.

5.2.8 Torej, v kolikor, zato, če, kateri, da

Pri vezniških izrazih *torej*, *v kolikor*, *zato*, *če*, *kateri* in *da* ni novih posebnosti. Zopet smo pri dvodelnih vezniških izrazih pozorni na besedo pred prejšnjo, pri enodelnih pa na prejšnjo besedo (besedo pred trenutno). Da je atribut za iskani veznik lahko enak 1, pred vezniškimi izrazi ne sme stati katero izmed ločil *,(;-:*, dodatno pri vezniku *da* ne sme biti spodnjega in zgornjega narekovaja. Nadalje pred *torej* ne sme biti besed *in*, *nam*, *jim*, *ji*, *mu*, *nama*, *vama*, *njima*, *jima*, *vam*, *meni*, *mi*, *tebi*, *ti*, pred *v kolikor* ne besed *in*, *ali*, *ter*, *le*, *samo*, *tudi*, *potem*, *zlasti*, *celo* in *razen*. Pred vezniškimi besedami *zato*, *če*, *kateri* in *da* poleg naštetih ločil pred seboj ne smejo imeti besed *in*, *ali* in *ter*, dodatno veznik *zato* ne sme imeti besed *samo*, *le*, *zgolj*, *predvsem*, *lahko*, *ampak*, *glavnem*, *morda*. *Če* pa ne sme imeti besed *le*, *samo*, *tudi*, *da*, *zlasti*, *razen*, *ampak*, *a* in *kot*. Pred *kateri* ne sme stati *v*, *na*, *pred*, *nad*, *pri*, *ob*, *s*, *za*, *po*, *iz*, *o*, *skozi*, *pod*, *izza*, *okoli*, *vzdolž*, *do*, *od*, *proti*, *med*, *zaradi*, *h* (ob predpostavki, da zna uporabnik pravilno uporabljati predloga k/h) ter pred *da* ne sme stati katera izmed besed *tako*, *le*, *samo*, *kot*, *kakor*, *češ*, *ko*, *lahko*, *temveč*, *ne*, *ampak*, *nam*, *jim*, *vam*, *mu*, *mi*, *ji*, *jima*, *vama*, *namesto* in *a*.

V tabeli 5.2 imamo kot primer podane atribute, ki smo jih generirali s pomočjo pravil, ki jih za postavljanje vejic uporablja orodje LanguageTool. Uporabili smo isti stavek kot pri tabeli 5.1, če bi atribute dodajali v originalno množico. V prvem stolpcu imamo podano zaporedno številko atributa. V drugem stolpcu imamo zapisane vezniške izraze. Če trenutna beseda ustreza

pravilom za vezniški izraz, v tretjem stolpcu v isti vrstici najdemo vrednost 1, v nasprotnem primeru pa vrednost 0.

Zap. št.	**Opis atributa**	**Primer**
68	trenutna beseda ustreza pravilom za vezniško besedo *ampak*	*0*
69	trenutna beseda ustreza pravilom za vezniško besedo *ampak tudi*	*0*
70	trenutna beseda ustreza pravilom za vezniško besedo *da*	*0*
71	trenutna beseda ustreza pravilom za vezniško besedo *dokler*	*0*
72	trenutna beseda ustreza pravilom za vezniško besedo *kadar*	*0*
73	trenutna beseda ustreza pravilom za vezniško besedo *kaj*	*0*
74	trenutna beseda ustreza pravilom za vezniško besedo *kajne*	*0*
75	trenutna beseda ustreza pravilom za vezniško besedo *kajti*	*0*
76	trenutna beseda ustreza pravilom za vezniško besedo *kako*	*1*
77	trenutna beseda ustreza pravilom za vezniško besedo *kakor*	*0*
78	trenutna beseda ustreza pravilom za vezniško besedo *kakršen*	*0*
79	trenutna beseda ustreza pravilom za vezniško besedo *kakšen*	*0*
80	trenutna beseda ustreza pravilom za vezniško besedo *kam*	*0*
81	trenutna beseda ustreza pravilom za vezniško besedo *kamor*	*0*
82	trenutna beseda ustreza pravilom za vezniško besedo *kar*	*0*

83	trenutna beseda ustreza pravilom za vezniško besedo *kateri*	*0*
84	trenutna beseda ustreza pravilom za vezniško besedo *kdaj*	*0*
85	trenutna beseda ustreza pravilom za vezniško besedo *ker*	*0*
86	trenutna beseda ustreza pravilom za vezniško besedo *ki*	*0*
87	trenutna beseda ustreza pravilom za vezniško besedo *kje*	*0*
88	trenutna beseda ustreza pravilom za vezniško besedo *kjer*	*0*
89	trenutna beseda ustreza pravilom za vezniško besedo *ko*	*0*
90	trenutna beseda ustreza pravilom za vezniško besedo *kolikor*	*0*
91	trenutna beseda ustreza pravilom za vezniško besedo *kot*	*0*
92	trenutna beseda ustreza pravilom za vezniško besedo *ne da bi*	*0*
93	trenutna beseda ustreza pravilom za vezniško besedo *preden*	*0*
94	trenutna beseda ustreza pravilom za vezniško besedo *predlog+kaj*	*0*
95	trenutna beseda ustreza pravilom za vezniško besedo *predlog+kar*	*0*
96	trenutna beseda ustreza pravilom za vezniško besedo *predlog+kateri*	*0*
97	trenutna beseda ustreza pravilom za vezniško besedo *saj*	*0*
98	trenutna beseda ustreza pravilom za vezniško besedo *temveč tudi*	*0*
99	trenutna beseda ustreza pravilom za vezniško besedo *tj.*	*0*
100	trenutna beseda ustreza pravilom za vezniško besedo *to je*	*0*
101	trenutna beseda ustreza pravilom za vezniško besedo *toda*	*0*

102	trenutna beseda ustreza pravilom za vezniško besedo *torej*	*0*
103	trenutna beseda ustreza pravilom za vezniško besedo *v kolikor*	*0*
104	trenutna beseda ustreza pravilom za vezniško besedo *vse dokler*	*0*
105	trenutna beseda ustreza pravilom za vezniško besedo *zaradi*	*0*
106	trenutna beseda ustreza pravilom za vezniško besedo *zato*	*0*
107	trenutna beseda ustreza pravilom za vezniško besedo *zatorej*	*0*
108	trenutna beseda ustreza pravilom za vezniško besedo *če*	*0*

Tabela 5.2: Pregled dodanih atributov (od zaporedne št. 68 do 108) za besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje.*

5.3 Dodatna pravila, ki povzročajo zadrege

Implementirali smo še nekaj dodatnih pravil, ki bodisi uporabnikom povzročajo največ težav (opisali smo jih v poglavju 2.2) bodisi zahtevajo dodatno pozornost pri pisanju (opisano v poglavju 2.1). Nekatera pravila smo implementirali tako, da smo dodali pogoj pri implementaciji obstoječih pravil, ki so opisana v poglavju 5.2, ostala pa smo ustvarili kot nove samostojne atribute.

5.3.1 Členek *da*

V poglavju 5.2.8 smo opisali pravilo za postavljanje vejice, kadar naletimo na podredni veznik *da*. 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 veš? - Baje da pije. - Iz Blatnega Dola da ste?* [6]

Obstoječemu atributu, generiranemu za veznik *da*, ki smo ga opisali v 5.2.8, smo dodali nov pogoj, kjer smo preverili, ali je MSD koda trenutne besede enaka L (označevanje členkov smo opisali v poglavju 3.1.3). Če je bil pogoj izpolnjen, je atribut brez nadaljnjih preverjanj dobil vrednost 0. Šele ob neizpolnjevanju pogoja (oz. izpolnjevanje negiranega 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, 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.*

5.3.2 Večbesedni vezniki

Na nekaj večbesednih veznikov smo naleteli že v poglavju 5.2, npr. *ampak tudi*, *temveč tudi*, *vse dokler* itd. 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. [6]

To smo implementirali tako, da smo vsem do sedaj generiranim atributom (opisali smo jih v 5.2) 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 Vp ali Vd (ali pa prva črka MSD kode ni V), potem atribut za ta veznik prejšnji besedo nastavi na 1. V nasprotnem primeru se pogoj ne izpolne in atribut dobi vrednost 0.

S to implementacijo smo vplivali na vse veznike iz poglavja 5.2 in preprečili, da se postavi vejica med vezniki, ki so del enega večbesednega vezniškega izraza.

5.3.3 In sicer, in to, in če, in ko

V prejšnjem podpoglavju 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, ker bi bilo iskanje vseh večbesednih veznikov in implementacija vsakega posebej časovno potratna. Ker to ni realna situacija (Obeliks besedne vrste pripisuje z 98,30% natančnostjo [1]), morda z zgoraj opisanim postopkom v oblikoslovni označevalnik polagamo preveliko zaupanje. Zato v tem razdelku opišemo izvedbo pravila brez uporabe MSD kod in brez zanašanja na označevalnik.

V primerih, ko veznik *in* nastopa v kombinaciji z besedami *sicer*, *to*, *če* in *ko*, 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 *trenutna beseda-1*) atribut za ta veznik nastavimo na 1, v nasprotnem primeru pa na 0.

Zap. št.	**Opis atributa**	**Primer**
109	trenutna beseda ustreza pravilom za vezniško besedo *in sicer*	*0*
110	trenutna beseda ustreza pravilom za vezniško besedo *in to*	*0*
111	trenutna beseda ustreza pravilom za vezniško besedo *in če*	*0*
112	trenutna beseda ustreza pravilom za vezniško besedo *in ko*	*0*

Tabela 5.3: Pregled dodatnih atributov (od zaporedne št. 109 do 112) za besedo *vidno* v stavku *Tu je že vidno, kako odločno je prepričana v svoje dejanje.*

5.4 Izboljšava obstoječe množice podatkov

V razdelku 5.1.1 smo opisali obstoječe atribute. Tukaj se sprašujemo o smiselnosti takšne implementacije. Za nekatere atribute predlagamo spremembe, ki bodo dale strojnemu učenju več informacije. Nekatere atribute, ki otežujejo strojno učenje, izločimo iz podatkovne množice.

5.4.1 Beseda kot atribut

Omenili smo, da so nekateri atributi besede z zalogo vrednosti vseh besed, ki se pojavijo v besedilu. Iz tega sledi, 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 same besede 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 podajamo kot atribute. S tem se izognemo drugačni obravnavi besed z isto besedno vrsto (ali celo oblikoskladenjsko oznako), ki določajo enako vez z drugimi besedami in v stavku nosijo isti pomen. Npr. če bi naleteli na primer, kjer bi s strojnim učenjem ugotovili, da pred besedo *ekran* stoji vejica, če pred njo stoji beseda *tipkovnica* (npr. pri naštevanju komponent računalnika), bi s predlagano izbiro atributov računalnik pri strojnem učenju lažje ugotovil, da isto pravilo velja tudi za besedi *tiskalnik* in *miška*. Z obstoječimi atributi bi ti dve povezavi obravnavali kot popolnoma različni. S tem premislekom atribute za vseh enajst vrednosti (trenutna beseda in pet besed pred njo ter pet besed za njo) spremenimo v oznake. V tabeli 5.1 so to atributi na mestih 1, 7, 13, 19, 25, 31, 37, 43, 49, 55 in 61.

5.4.2 Lema kot atribut

Enako kot je nesmiselno definirati atribut glede na samo besedo, saj za strojno učenje ne pomeni drugega kot nek skupek črk, ki se lahko pojavi večkrat v besedilu ali pa tudi ne, je enak premislek tudi pri definiranju osnovne oblike

besede (leme). Tudi ta atribut prestrukturiramo v oznako. Spremenili smo dodatnih 11 atributov. Te atribute v tabeli 5.1 najdemo na mestih 2, 8, 14, 20, 26, 32, 38, 44, 50, 56 in 62.

5.4.3 Oblikoskladenjska oznaka kot atribut

Tudi tretji atribut, definiran v obstoječi množici podatkov, ima svoje pomanjkljivosti. S trenutno definirano množico podatkov lahko vsako besedo v našem besedilu razvrstimo v 930 oblikoskladenjskih oznak. Samostalnik lahko zavzame eno izmed 78 vrednosti, zaimek pa kar eno izmed 451 vrednosti. Pridevnik lahko razvrstimo v 168 oznak in glagol v 127. Z bogatejšim in večjim korpusom bi bile te številke še večje. 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. Npr. glagoli *bosta*, *nista* in *bova* zavzemajo tri različne oblikoskladenjske oznake, strojnemu učenju pa želimo povedati, da naj ukrepa pri vseh enako. Pri postavljanju vejic nas zanima, ali je beseda samostalnik, ne pa tudi njegova vrsta, spol, število, sklon in živost. Npr. stavek *"Rad imam brata, sestro, bratranca in sestrično."* bi moral strojnemu učenju povedati, da obstaja povezava s stavkom *"Na obisk grem k bratu, sestri, bratrancu in sestrični."* Vendar je povezave in pravila za postavljanje vejic težko razbrati, če za stvari, ki bi nam morale podati isto informacijo, uporabljamo več deset različnih oznak.

Po tem premisleku bomo neinformativni atribut, ki opisuje MSD kodo in lahko sedaj zavzame 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 besed znotraj okna. Ostalih atributov ne bomo več spreminjali.

Opis MSD kode z enim atributom

Prva implementacija sprememb bo vplivala na enajst atributov. To so vsi atributi s podatkom o MSD kodi za trenutno besedo in za besede znotraj

okna. V tabeli 5.1 so to atributi z zaporednimi številkami 3, 9, 15, 21, 27, 33, 39, 45, 51, 57 ter 63.

Vse različne oznake samostalnika, ki so kombinacije vrste, spola, števila, sklona in živosti, ki pri samostalniku tvorijo mnogo različnih oblikoskladenjskih oznak, nadomestimo z eno, ki bo strojnemu učenju povedala najpomembneje - naleteli smo na samostalnik. Vse možne oznake, ki določajo samostalnik, npr. *Somei* (besedna vrsta samostalnik, vrste občno ime, moškega spola, v ednini in sklonu imenovalnik), *Somer* (samostalnik vrste občno ime, moškega spola, v ednini in sklonu rodilnik), *Somed* (samostalnik vrste občno ime, moškega spola, v ednini in sklonu dajalnik), *Sometn* (samostalnik vrste občno ime, moškega spola v ednini in sklonu tožilnik ter brez živosti), *Sometd* (samostalnik vrste občno ime, moškega spola, v ednini, sklon tožilnik in z živostjo) in vse ostale, nadomestimo z oznako S (samostalnik). S tem se izognemo situacijam, kjer algoritem strojnega učenja ne bi vedel kaj storiti z oznako, ki se redko pojavi (ali pa se npr. pojavi samo v testni množici), naučen pa bi bil ukrepanja pri isti besedni vrsti v drugem spolu (in/ali številu, sklonu,...) in s tem z drugačno oznako.

Enako storimo z ostalimi besednimi vrstami. Pri glagolu nas ne zanima vrsta, vid, oblika, oseba, število, spol in nikalnost, zanima nas le to, ali je beseda glagol. Zato vse oznake, ki določajo glagol (glagol določa kar 156 različnih oznak po oblikoskladenjskih oznakah JOS), zamenjamo z novo oznako, s črko G.

Tudi pri pridevniku vse 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 z novo, unikatno oznako za vse pridevnike, črko S.

Prislov lahko po Priporočilih za oblikoslovno označevanje JOS razvrstimo v razmeroma malo oznak, teh je 4. Kljub temu bomo tudi tu vse oznake zamenjali s črko R, ker tudi prislov na obstoj vejice ne vpliva s svojo vrsto ali stopnjo, pač pa s svojo besedno vrsto in vplivom (povezavami) na druge besede v povedi. Besedno vrsto, kadar naletimo na prislov, sporočimo s črko

R.

Enako storimo za zaimek, številnik in predlog. Vse oznake, ki določajo zaimek ter njegovo vrsto, osebo, spol, število, sklon, število svojine, spol svojine ter naslonskost, zamenjamo z oznako Z. Pri številniku vse oznake, s katerimi označimo številnik z njegovim zapisom, vrsto, spolom, številom, sklonom in določnostjo, zamenjamo z oznako K. Predlog različne oznake opišejo s kombinacijo črke D in dodatne črke, ki določa njegov sklon. Tudi te oznake nadomestimo s črko D. Enako bomo storili še za kategorijo Neuvrščeno, kamor spadajo razne neznane besede, tipkarske napake in besede, ki jih je program napačno tokeniziral. Vse besede, ki spadajo pod to kategorijo, nadomestimo z N.

Obdržali bomo oznake za veznik. Kot smo opisali v poglavju 3.1.1, 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, bomo obdržali obe oznaki. Prav tako obdržimo oznake za medmet, okrajšavo in členek. Prvega, tako kot prej, označimo s črko M, drugo s črko O in tretjega z L.

Z naštetimi popravki smo za atribute, opisane v tabeli 5.1, dobili novo množico vrednosti, ki jo lahko zavzame beseda, ki se pojavi v uporabljenem korpusu. Spremenili smo množico vrednosti atributom na mestih 3, 9, 15, 21, 27, 33, 39, 45, 51, 57 in 63. Sedaj vsaka beseda v korpusu ne zavzame več ene izmed 930 različnih oznak, pač pa eno izmed 13 oznak. Z zmanjšanim številom oznak želimo izboljšati strojno učenje. Manjše število oznak za strojno učenje pomeni lažje iskanje povezav pri istih besednih vrstah (sedaj vse iste besedne vrste označujemo z isto oznako) in s tem tudi povezav z drugimi besednimi vrstami ter zato boljše posploševanje.

V tabeli 5.1 vidimo, kako s popravljeno množico atributov opišemo besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje.* Opazimo, da se je število atributov zmanjšalo, prav tako vidimo, da so oblikoskladenjske oznake zamenjane z novimi oznakami. To dodatno definiranim

atributom (poglavji 5.2 in 5.3) spremeni zaporedne številke atributov. Zaporedna številka vsakega atributa se zmanjša za 22 mest. To je toliko, kolikor atributov smo odstranili iz obstoječe množice podatkov.

Zap. št.	**Opis atributa**	**Primer**
1	MSD koda trenutne besede	*R*
2	obstaja povezava na trenutno besedo	*1*
3	obstaja modra povezava na trenutno besedo	*0*
4	obstaja rdeča povezava na trenutno besedo	*0*
5	MSD koda prejšnje besede	*L*
6	obstaja povezava na prejšnjo besedo	*1*
7	obstaja modra povezava na prejšnjo besedo	*0*
8	obstaja rdeča povezava na prejšnjo besedo	*0*
9	MSD koda predprejšnje besede	*G*
10	obstaja povezava na predprejšnjo besedo	*0*
11	obstaja modra povezava na predprejšnjo besedo	*0*
12	obstaja rdeča povezava na predprejšnjo besedo	*0*
13	MSD koda besede na položaju tri mesta pred trenutno	*R*
14	obstaja povezava na besedo tri mesta pred trenutno	*1*
15	obstaja modra povezava na besedo tri mesta pred trenutno	*0*
16	obstaja rdeča povezava na besedo tri mesta pred trenutno	*0*
17	MSD koda besede štiri mesta pred trenutno	*
18	obstaja povezava na besedo štiri mesta pred trenutno	*
19	obstaja modra povezava na besedo štiri mesta pred trenutno	*
20	obstaja rdeča povezava na besedo štiri mesta pred trenutno	*
21	MSD koda besede pet mest pred trenutno	*
22	obstaja povezava na besedo pet mest pred trenutno	*

23	obstaja modra povezava na besedo pet mest pred trenutno	*
24	obstaja rdeča povezava na besedo pet mest pred trenutno	*
25	MSD koda besede eno mesto za trenutno	*R*
26	obstaja povezava na besedo eno mesto za trenutno	*1*
27	obstaja modra povezava na besedo eno mesto za trenutno	*1*
28	obstaja rdeča povezava na besedo eno mesto za trenutno	*0*
29	MSD koda besede dve mesti za trenutno	*R*
30	obstaja povezava na besedo dve mesti za trenutno	*0*
31	obstaja modra povezava na besedo dve mesti za trenutno	*0*
32	obstaja rdeča povezava na besedo dve mesti za trenutno	*0*
33	MSD koda besede tri mesta za trenutno	*G*
34	obstaja povezava na besedo tri mesta za trenutno	*0*
35	obstaja modra povezava na besedo tri mesta za trenutno	*0*
36	obstaja rdeča povezava na besedo tri mesta za trenutno	*0*
37	MSD koda besede štiri mesta za trenutno	*P*
38	obstaja povezava na besedo štiri mesta za trenutno	*0*
39	obstaja modra povezava na besedo štiri mesta za trenutno	*0*
40	obstaja rdeča povezava na besedo štiri mesta za trenutno	*0*
41	MSD koda besede pet mest za trenutno	*D*
42	obstaja povezava na besedo pet mest za trenutno	*0*
43	obstaja modra povezava na besedo pet mest za trenutno	*1*
44	obstaja rdeča povezava na besedo pet mest za trenutno	*0*
45	razredni atribut, ki pove ali trenutni besedi sledi vejica	*je-vejica*

Tabela 5.4: Opis redefiniranih atributov za besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje.*

Opis MSD kode z devetimi atributi

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 pravzaprav obdržali samo podatke o besedni vrsti. Z drugim načinom bomo vključili vse podatke, ki jih nosi posamezna MSD koda. S tem želimo zagotoviti več informacije za strojno učenje. Najdaljšo oblikoskladenjsko oznako ima zaimek, kjer je MSD koda sestavljena iz 9 znakov. MSD koda glagola in številnika je dolga 8 znakov, pridevnika 7, samostalnika 6 itd. Ker je najdaljša oznaka dolga 9 znakov, bomo iz osnovne MSD kode generirali devet atributov. V osnovni podatkovni množici MSD koda določi 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. Oglejmo si atribut z zaporedno številko 15 v tabeli 5.1. Vrednost MSD kode je *Gp-ste-n* in to je vrednost enega atributa. Z novo implementacijo ta atribut spremenimo v devet novih tako, da vsak izmed novih atributov dobi vrednost enega od znakov MSD kode. Tako za konkretni primer dobimo devet novih atributov z vrednostmi: *G,p,-,s,t,e,-,n,**. Kadar je dolžina MSD kode manjša od 9 znakov, atributom, ki določajo preostale znake, nastavimo vrednost ***. Ker spreminjamo 11 atributov, torej po en atribut za vsako besedo znotraj okna, to pomeni, da enajst osnovnih atributov nadomestimo z 99 novimi. Naša podatkovna množica se tako poveča iz 90 na 178 atributov za opis trenutne besede.

V tabeli 5.5 vidimo, kako se atribut z zaporedno številko 13 iz tabele 5.4 preslika v novih devet atributov. Ker je oblikoskladenjska oznaka krajša od devetih znakov, skladno z dosedanjim označevanjem manjkajočih podatkov preostali znak nadomestimo z znakom *.

Zap. št.	Opis atributa	Primer
13	prvi atribut določa besedno vrsto	*G*
14	drugi atribut pri samostalniku, glagolu, pridevniku, prislovu, zaimku, vezniku in besedam, ki spadajo pod neuvrščeno, določa vrsto, pri številniku zapis ter pri predlogu pa sklon	*p*
15	tretji atribut pri samostalniku določa spol, pri glagolu vid, pri pridevniku in prislovu stopnjo, pri zaimku osebo in pri številniku vrsto	-
16	četrti atribut pri samostalniku določa število, pri glagolu obliko, pri pridevniku, zaimku in številniku pa spol	*s*
17	peti atribut pri samostalniku določa sklon, pri glagolu osebo, pri pridevniku, zaimku in številniku število	*t*
18	šesti atribut pri samostalniku določa živost, pri glagolu število, pri pridevniku, zaimku in številniku sklon	*e*
19	sedmi atribut pri glagolu določa spol, pri pridevniku določnost, pri zaimku število svojine, pri številniku določnost	-
20	osmi atribut pri glagolu določa nikalnost, pri zaimku pa spol svojine	*n*
21	deveti atribut pri zaimku določa naslonskost	*

Tabela 5.5: Nove vrednosti atributa z zaporedno številko 13 iz tabele 5.4 za besedo *vidno* iz stavka *Tu je že vidno, kako odločno je prepričana v svoje dejanje.* MSD kodo opisujemo s po devetimi atributi za vsako besedo znotraj okna. Ta atribut se nanaša na predprejšnjo besedo.

Opis MSD kode z osemintridesetimi atributi

Implementirali smo dva različna postopka opisa MSD oznak z atributi. S prvim načinom smo uporabili le prvo črko MSD oznake, ki nam pove besedno vrsto, le pri veznikih smo uporabili dve črki, da smo obdržali tudi podatek o vrsti veznika. Z drugim načinom smo obdržali vse podatke, ki jih nosi posamezna MSD oznaka tako, da smo za vsako MSD oznako definirali devet atributov, kjer je vsak nosil po eno črko oznake. 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. To bi bil edini atribut, ki bi nosil podatek o posamezni besedni vrsti in bi bil skupen vsem besednim vrstam ne glede na samo besedno vrsto, oz. njegova vrednost ne bi bila nikoli enaka *. Zasedal bi lahko 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. Atributu o besedni vrsti bi sledili vsi atributi znotraj izbrane besedne vrste, to so vsi znaki MSD oznake razen prvega. 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 *. Če bi nato sledilo definiranje glagola, kjer lahko njegova MSD oznaka zasede največ sedem mest brez podatka o besedni vrsti (tega smo zapisali v prvi atribut), bi to pomenilo, da so mesta od sedmega pa do vključno trinajstega rezervirana za vrednosti atributov, če je trenutna beseda samostalnik. Če trenutna beseda ne bi bila samostalnik, bi imeli atributi na teh mestih vrednost *. Sledilo bi šest atributov, ki bi bili različni od *, kadar bi bila trenutna beseda pridevnik, nato dve mesti za vrednosti prislova, sledilo bi osem mest za zaimek, šest za števnik, po eno za predlog, veznik in neuvrščeno. MSD oznake medmetov in okrajšav so dolge po en znak, ki nam pove besedno vrsto, to pa že uporabimo

pri določanju vrednosti prvega atributa, zato za ti dve besedni vrsti dodatnih atributov ne bi definirali. To bi pomenilo, da bi atribute o MSD oznakah, ki smo jih uporabljali doslej, zamenjali z 38 novimi atributi. Pri vseh bi prvi atribut določal besedno vrsto, ostali pa bi bili enaki *, razen če bi trenutna beseda ustrezala besedni vrsti, ki določa atribute na tistih mestih. Znotraj okoliškega okna smo s prvim načinom opisa MSD oznake imeli 9 atributov za opis oblikoskladenjskih oznak, z drugim smo imeli 99 atributov za (bolj informativen) opis oblikoskladenjskih oznak, s tretjim načinom pa bi to pomenilo 418 atributov, ki bi nosili informacije o MSD oznakah.

Tretjega načina nismo implementirali, saj bi bila potrebna še večja računska moč, kot jo imamo na voljo, idejo pa puščamo za nadaljnje delo.

Poglavje 6

Izbira podmnožice atributov

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. Ogledali si bomo kvaliteto najpomembnejših atributov dveh korpusov, pri vsakem znotraj dveh podatkovnih množic, kjer je razlika le pri atributih, ki podajajo informacijo o oblikoskladenjski oznaki. To podajamo bodisi z 11 bodisi z 99 atributi.

6.1 ReliefF

Iz obstoječe podatkovne množice s 67 atributi smo v razdelku 5.4 nekaj atributov izločili. V razdelkih 5.2 in 5.3 smo dodali 45 novih atributov, ki smo jih generirali na podlagi pravil. Posodobljeni podatkovni množici smo na dva načina modificirali atribute, ki opisujejo oblikoskladenjske oznake. Tako smo dobili dve novi podatkovni množici. Prva, ki vsebuje 90 atributov, ima vsako oblikoskladenjsko oznako opisano z največ enim atributom, znotraj okna je to 11 atibutov, ki opisujejo oblikoskladenjske oznake. Druga množica vsebuje 178 atributov in ima vsako oblikoskladenjsko oznako opisano z devetimi atributi, to za vsako besedo z njenim okoliškim oknom nanese 99 atributov. Zanima nas, kako pomembni so atributi, predvsem pa nas zanima, koliko informacije nosijo oblikoskladenjske oznake, če jih opišemo bodisi z enim bo-

disi z devetimi atributi. Ocenili smo kvaliteto atributov za obe podatkovni množici z mero za ocenjevanje atributov ReliefF.

Na voljo imamo dva korpusa, kjer je drugi izboljšana in nadgrajena verzija prvega. Kadar se bomo sklicevali na podatkovne množice prvega korpusa, uporabimo oznako Šolar1, kadar bomo govorili o izboljšani verziji tega korpusa pa Šolar2. Algoritem za ocenjevanje atributov najprej poženemo na prvem korpusu. Ker nam mera za ocenjevanje atributov poda tudi vpogled v kvaliteto samega korpusa, ne le njegovih atributov, z mero za ocenjevanje atributov ocenimo tudi atribute originalne podatkovne množice. Ocenili bomo tudi atribute obeh podatkovnih množic drugega korpusa, kjer se razlikuje število atributov za opis MSD kode.

V poglavju 7 bomo vsa testiranja opravili na dveh podatkovnih množicah, kjer so atributi generirani glede na besedilo prvega korpusa. Testiranja opravimo tudi na podatkovnih množicah, ki so rezultat obdelave drugega korpusa. Zaradi velike časovne zahtevnosti bomo nekatera izmed testiranj opravili na zmanjšanih množicah podatkov.

6.2 Kakovost atributov

Pri vsaki izmed podatkovnih množic si bomo ogledali prvo tretjino najbolje ocenjenih atributov. V tabeli 6.1 vidimo ocenjene atribute originalne podatkovne množice. To je podatkovna množica, ki za opis besede uporablja 67 atributov. To so vsi atributi, preden smo množici odstranili tiste s podatki o oblikah in lemah, prav tako preden smo dodali atribute, ki smo jih generirali s pomočjo slovničnih pravil ter pravil, ki jih za postavljanje vejic uporablja orodje LanguageTool in preden smo spremenili vrednosti tistih atributov, ki nosijo informacije o MSD kodi. Med najbolje ocenjenimi atributi so samo štirje atributi takšni, za katere smo pričakovali, da nosijo največ informacije za pravilno postavljanje vejic. Opazimo, da je najbolje ocenjen atribut z MSD kodo, nato atributa z oblikami in lemami. Atribut *msd0*, ki nosi podatek o MSD kodi, lahko zasede eno izmed 930 vrednosti, atribut *obl0*, ki

nosi podatek o obliki besede, lahko zavzame kar 23082 različnih vrednosti, atribut *lem0* pa 10951. Ti atributi so zelo verjetno dobro ocenjeni le za to specifično učno množico oz. korpus. Najboljša mesta zasedajo po našem mnenju neinformativni atributi. To nam pove, da je uporabljeni korpus morda neprimeren za strojno učenje. Zaradi tega lahko pričakujemo, da se bodo rezultati klasifikacije na izboljšanih množicah atributov poslabšali.

Zap. št.	Ocena	Ime atributa	Zap. št.	Ocena	Ime atributa
1	0.278	msd0	12	0.093	msd-2
2	0.273	lem0	13	0.091	msd2
3	0.251	je_vez1	14	0.082	lem-2
4	0.243	obl0	15	0.077	obl-2
5	0.213	lem1	16	0.076	je_vez0
6	0.197	obl1	17	0.072	msd-3
7	0.177	msd1	18	0.072	zac_modrega0
8	0.134	msd-1	19	0.066	obl2
9	0.109	lem-1	20	0.066	lem2
10	0.103	zac_modrega1	21	0.064	lem-3
11	0.102	obl-1	22	0.0604	obl-3

Tabela 6.1: Rezultat ocenjevanja atributov z mero ReliefF originalne podatkovne množice.

V tabelah 6.2 in 6.3 prikažemo rezultate ocenjevanja atributov z mero ReliefF spremenjene podatkovne množice, na kateri smo napravili vse spremembe, opisane v poglavju 5. V prvi tabeli imamo podan rezultat ocenjevanja atributov, ko smo atribute o oblikoskladenjski oznaki opisali na prvi način (opisano v razdelku 5.4.3): originalen atribut s podatkom o MSD kodi smo spremenili tako, da smo obdržali le prvo črko oznake, razen pri veznikih, kjer smo obdržali dve črki. S tem smo obdržali enako število atributov za

opis MSD kode kot v originalni podatkovni množici. Ta podatkovna množica sestoji iz 90 atributov, od tega jih enajst uporabi za opis MSD kode, zato jo bomo označili z imenom Šolar1-11. V tabeli 6.3 imamo rezultate ocenjevanja atributov druge podatkovne množice, označili jo bomo z imenom Šolar1-99, saj imamo 99 atributov za opis MSD kod trenutne besede in besed znotraj okna, vseh atributov pa je 178.

Zap. št.	Ocena	Ime atributa	Zap. št.	Ocena	Ime atributa
1	0.301	msd0	16	0.057	je_vez-1
2	0.249	msd1	17	0.055	zac_rdecega-1
3	0.196	msd-1	18	0.052	msd4
4	0.134	msd-2	19	0.048	je_vez2
5	0.132	je_vez1	20	0.047	zac_rdecega-2
6	0.125	msd2	21	0.041	msd5
7	0.111	msd-3	22	0.040	zac_modrega-1
8	0.103	msd-4	23	0.039	zac_rdecega-3
9	0.090	zac_modrega1	24	0.038	je_vez-2
10	0.083	msd-5	25	0.038	zac_rdecega2
11	0.078	zac_modrega0	26	0.037	je_vez-4
12	0.0741	msd3	27	0.036	je_da
13	0.071	je_vez0	28	0.036	zac_rdecega-4
14	0.065	zac_rdecega1	29	0.033	je_vez-3
15	0.057	zac_rdecega0	30	0.032	zac_modrega2

Tabela 6.2: Rezultat ocenjevanja atributov z mero ReliefF spremenjene podatkovne množice Šolar1-11.

Če primerjamo obe tabeli, vidimo, da so v obeh podatkovnih množicah najpomembnejši atributi z MSD kodo, pri obeh sta MSD kodi trenutne besede

in prejšnje besede v samem vrhu pomembnih atributov. V tabeli 6.3 vidimo, da je pri upoštevanju MSD kode trenutne besede najbolj informativen tretji znak te kode, nato pa prvi znak (to je podatek o besedni vrsti). Pri MSD kodi prejšnje besede je najpomembnejši prvi znak MSD kode, sledi pa mu drugi znak in nato četrti. Npr. za besedo *pojdi* iz stavka *"Le pojdi, zdaj."*, kjer ima trenutna beseda MSD oznako *Ggvvde*, prejšnja *L* in naslednja *Rsn*, bi to pomenilo, da strojnemu učenju pri postavljanju vejic najbolj pomaga tretji znak MSD oznake trenutne besede, ki nam v našem primeru pove glagolski vid (ta je lahko dovršni, nedovršni in dvovidski [3]), nato prva črka MSD kode trenutne besede, ki nam pove besedno vrsto in zatem prva črka MSD kode prejšnje besede, ki nam pove besedno vrsto prejšnje besede. Sledi peta črka MSD kode trenutne besede. V našem primeru nam pove, v katerem številu (ednina, množina, dvojina) se nahaja trenutna beseda, ki je glagol itd. Med prvo tretjino najpomembnejših atributov se znajde tudi atribut, ki smo ga opisali v razdelku 5.2.8 in nam pove, ali besede ustreza pogojem za besedo *da*.

To, da največ informacije za strojno učenje nosi atribut, ki je tretji znak MSD kode in v našem primeru sporoča glagolski vid ter da je četrti najpomembnejši tisti, ki določa peti znak MSD kode in nam pove število, za naše razumevanje postavljanja vejic v slovenščini ni logično in iz tega skledamo, da korpus ne odraža dejanskih zakonitosti vejice v slovenščini. Tretji znak MSD kode pri samostalniku določa spol, pri glagolu vid, pri pridevniku in prislovu stopnjo, pri zaimku osebo, pri številniku vrsto, pri predlogu, vezniku, členku, medmetu in okrajšavi pa manjkajoči znak. Težko si predstavljamo, da bi bil obstoj vejice odvisen od naštetih lastnosti besednih vrst.

Kvaliteta atributov pri uporabi izboljšanega korpusa

V tabelah 6.4 in 6.5 si ogledamo rezultate ocenjevanja atributov z mero ReliefF podatkovnih množic, ki smo jih zgradili z uporabo izboljšanega korpusa. Predstavimo tretjino najpomembnejših atributov. Mero za ocenjevanje atributov smo pri množici Šolar2-11 pognali na vseh 728927 primerih. Zaradi

velike časovne zahtevnosti (ocenjevanje atributov množice Šolar2-11 je trajalo teden dni) smo pri množici Šolar2-99 za ocenjevanje atributov uporabili desetino primerov velikosti množice, to je 72892 primerov.

Če primerjamo na enak način definirane atribute za opis oblikoskladenjskih oznak pri uporabi dveh različnih korpusov, vidimo, da so v tabeli 6.4 ocene atributov podobne ocenam istih atributov na slabšem korpusu, ki smo jih navedli v tabeli 6.2. Množici najbolje ocenjenih atributov se ne razlikujeta dosti. Razen manjših sprememb v vrstnem redu atributov, vidimo, da so se med najboljšimi atributi pojavili trije novi, dva pa sta bila odstranjena. Novi atributi so *zac_rdecega-5*, *zac_modrega-5* in *zac_modrega-2*. Atributa, ki ju ne najdemo več v tretjini najpomembnejših sta *je_vez3* ter *zac_modrega2*. Vidimo, da so z uporabo novega korpusa pomembnejše povezave med bolj odmaknjenimi besedami. Kar deset atributov v obeh podatkovnih množicah zaseda ista mesta med prvo tretjino najpomembnejših, to so *msd0*, *msd1*, *msd-1*, *msd-2*, *msd3*, *je_vez-1*, *zac_rdecega-1*, *zac_rdecega-2*, *zac_rdecega2* ter *je_da*. Atributi *je_vez1*, *zac_modrega1*, *zac_rdecega0*, *zac_modrega-1* ter *je_vez-2* so se na lestvici pomaknili za nekaj mest nižje (so manj pomembni), atributa *msd4* in *zac_rdecega-4* pa višje (so ocenjeni kot pomembnejši). Vsa mesta atributov v prvi tretjini najpomembnejših so pri uporabi izboljšanega korpusa dobila nekoliko boljšo oceno, npr *msd0* s starim korpusom dobi oceno 0,301, z novim korpusom pa 0,346.

Ob primerjavi tabele 6.5 s tabelo 6.3 vidimo, da pri večini atributov ni veliko sprememb. Večina atrbutov ostane na podobnih mestih kot prej, atributi *msd-1-2*, *msd0-2*, *je_vez1*, *msd-1-4*, *msd-1-5*, *zac_modrega1*, *msd2-3*, *msd3-3* obdržijo svoje mesto na lestvici najboljših. Atributa *msd-2-3* in *msd0-7* se pomakneta nekaj mest nižje. Štirje atributi zapustijo seznam, to so *msd0-8*, *msd-1-8*, *zac_rdecega0* in *je_vez0*, nadomestijo jih *msd-3-5*, *msd5-2*, *msd-3-4*, *msd-3-2*. Z novim korpusom je nekoliko več poudarka na MSD oznakah besed znotraj okoliškega okna na račun povezav med besedami. Zanimivo je tudi, da so še vedno pomembni atributi, ki določajo neprve znake MSD kode.

Zap. št.	Ocena	Ime atributa	Zap. št.	Ocena	Ime atributa
1	0.348	msd0-3	31	0.085	msd2-3
2	0.319	msd0-1	32	0.085	msd2-5
3	0.310	msd-1-1	33	0.084	msd-1-7
4	0.310	msd0-5	34	0.082	msd-2-6
5	0.305	msd0-4	35	0.080	msd3-1
6	0.272	msd-1-2	36	0.078	msd0-8
7	0.270	msd0-2	37	0.077	je_vez2
8	0.243	je_vez1	38	0.076	zac_rdecega2
9	0.207	msd-1-4	39	0.075	msd3-2
10	0.184	msd-1-5	40	0.073	je_da
11	0.1832	msd1-1	41	0.071	msd4-1
12	0.180	msd-1-3	42	0.068	zac_modrega2
13	0.171	zac_modrega1	43	0.066	msd-3-1
14	0.157	msd1-2	44	0.065	msd-1-8
15	0.154	msd-2-1	45	0.065	msd4-2
16	0.1518	msd0-6	46	0.065	msd3-5
17	0.122	msd0-7	47	0.065	msd3-4
18	0.122	msd1-4	48	0.064	msd2-6
19	0.119	msd1-3	49	0.063	msd-2-7
20	0.115	msd-1-6	50	0.063	msd-2-8
21	0.114	msd-2-4	51	0.061	zac_modrega0
22	0.112	msd1-5	52	0.061	msd5-1
23	0.109	msd2-1	53	0.061	msd3-3
24	0.107	msd-2-2	54	0.060	zac_rdecega0
25	0.103	msd-2-5	55	0.060	je_vez0
26	0.102	zac_rdecega1	56	0.057	msd4-4
27	0.099	msd2-2	57	0.056	msd3-6
28	0.091	msd1-6	58	0.054	msd4-5
29	0.089	msd-2-3	59	0.054	msd4-3
30	0.087	msd2-4	60	0.054	msd1-7

Tabela 6.3: Rezultat ocenjevanja atributov z mero ReliefF spremenjene podatkovne množice Šolar1-99.

Zap. št.	Ocena	Ime atributa	Zap. št.	Ocena	Ime atributa
1	0.357	msd0-5	31	0.103	msd2-3
2	0.353	msd0-3	32	0.100	msd3-1
3	0.340	msd0-4	33	0.097	zac_rdecega2
4	0.337	msd-1-1	34	0.096	zac_modrega2
5	0.320	msd0-1	35	0.095	msd-2-3
6	0.292	msd-1-2	36	0.094	msd4-1
7	0.261	msd0-2	37	0.093	msd3-5
8	0.257	je_vez1	38	0.092	msd3-4
9	0.233	msd-1-4	39	0.089	msd3-2
10	0.215	msd-1-5	40	0.088	msd3-3
11	0.214	msd-1-3	41	0.083	msd0-7
12	0.195	msd1-1	42	0.079	msd-1-7
13	0.189	zac_modrega1	43	0.079	msd4-2
14	0.187	msd-2-1	44	0.078	msd-3-1
15	0.165	msd1-2	45	0.078	zac_modrega0
16	0.145	msd1-4	46	0.077	msd-2-6
17	0.143	msd1-5	47	0.076	je_da
18	0.139	msd0-6	48	0.075	msd4-4
19	0.133	msd1-3	49	0.074	msd5-1
20	0.129	msd2-1	50	0.073	msd2-6
21	0.121	zac_rdecega1	51	0.072	msd4-3
22	0.116	msd-2-4	52	0.071	msd4-5
23	0.113	msd2-2	53	0.070	msd3-6
24	0.109	msd2-5	54	0.065	msd-3-5
25	0.109	msd1-6	55	0.062	msd1-7
26	0.109	msd-1-6	56	0.061	msd-2-8
27	0.108	msd-2-5	57	0.061	msd5-2
28	0.107	msd-2-2	58	0.060	msd-2-7
29	0.105	msd2-4	59	0.060	msd-3-4
30	0.103	je_vez2	60	0.059	msd-3-2

Tabela 6.4: Rezultat ocenjevanja atributov z mero ReliefF desetine spremenjene podatkovne množice Šolar2-99 na izboljšanem korpusu.

Zap. št.	Ocena	Ime atributa	Zap. št.	Ocena	Ime atributa
1	0.355	msd0	16	0.055	je_vez-1
2	0.271	msd1	17	0.054	zac_rdecega-1
3	0.237	msd-1	18	0.054	zac_rdecega0
4	0.161	msd-2	19	0.051	msd5
5	0.132	msd2	20	0.045	zac_rdecega-2
6	0.132	msd-3	21	0.039	zac_rdecega-3
7	0.120	msd-4	22	0.035	zac_rdecega-4
8	0.106	msd-5	23	0.035	je_vez2
9	0.099	zac_modrega0	24	0.035	zac_rdecega-5
10	0.099	je_vez1	25	0.035	zac_rdecega2
11	0.082	zac_modrega1	26	0.034	zac_modrega-1
12	0.082	msd3	27	0.034	je_da
13	0.063	zac_rdecega1	28	0.033	je_vez-2
14	0.061	je_vez0	29	0.033	zac_modrega-5
15	0.060	msd4	30	0.032	zac_modrega-2

Tabela 6.5: Rezultat ocenjevanja atributov z mero ReliefF spremenjene podatkovne množice Šolar2-11 na izboljšanem korpusu.

Poglavje 7

Klasifikacija vejic

V tem poglavju opišemo testiranje klasifikacijskih modelov. Opišemo postopek, ki smo ga uporabili, da smo iz novih podatkovnih množic dobili uravnotežene podmnožice. Z njim uravnotežimo po dve spremenjeni podatkovni množici za vsak korpus, to sta podatkovni množici z novo zalogo vrednosti oblikoskladenjskih oznak ter brez neinformativnih atributov, da dobimo uravnotežene podmnožice. Testiranje opravimo z različnimi algoritmi na osnovni podatkovni množici obeh korpusov in za vsak korpus na dveh spremenjenih podatkovnih množicah ter na njunih uravnoteženih podmnožicah.

7.1 Prečno preverjanje

Testiramo po tri podatkovne množice za vsak korpus: osnovno, na kateri temeljijo predhodne raziskave [9], ter dve spremenjeni, ki smo jima odstranili neinformativne atribute ter spremenili zalogo vrednosti oblikoskladenjskih oznak. Prvi spremenjeni podatkovni množici smo atribute s podatki o MSD kodah spremenili tako, da sestojijo le iz prve črke oblikoskladenjske oznake, ta nam pove besedno vrsto (razen pri vezniku, kjer smo obdržali tudi podatek o tem, ali je veznik priredni ali podredni), da bi se s tem izognili veliki množici neinformativnih atributov. Drugi podatkovni množici smo oblikoskladenjske oznake spremenili v 9 atributov. Pri vseh teh podatkovnih množicah upora-

bimo desetkratno prečno preverjanje, ki je del programskega paketa Weka.

7.2 Implementacija prečnega preverjanja in podvzorčenja

Da bi čimbolj uravnotežili podatkovno množico, smo implementirati učenje, s katerim smo želeli združiti prednosti prečnega preverjanja in podvzorčenja (ang. undersampling). Podatkovno množico smo razdelili na deset približno enako močnih podmnožic, ki smo jim naključno dodali enako velik vzorec primerov brez vejic.

Podatkovno množico smo razdelili na pozitivne in negativne primere. Pozitivni primeri so tisti, ki imajo razred pozitiven (je-vejica), negativni pa razred ni-vejice. Zatem smo pozitivne primere naključno razbili na 10 enakih delov. Za vsakega izmed desetih delov smo vzeli vse vse preostale pozitivne primere (razen trenutnega dela), ter jim dodali enako število naključno izbranih negativnih primerov. Rezultat je deset učnih množic. Generirali smo še deset pripadajočih testnih množic, ki smo jih ustvarili tako, da smo vzeli trenutni del (desetino) pozitivnih primerov (del, ki smo ga izključili iz učne množice) in mu dodali vse negativne primere, ki jih nismo uporabili v pripadajoči učni množici. Učni algoritem pokličemo na vsaki učni množici ter nato testiramo na pripadajoči testni množici. Rezultate povprečimo in primerjamo z rezultati testiranj neuravnoteženih podatkovnih množic (tako z originalno kot spremenjenimi) pri uporabi različnih klasifikacijskih algoritmov.

Programersko rešitev prikazuje algoritem 1, kjer smo postopek realizirali tako, da smo najprej v dve podatkovni strukturi (seznama) ločili primere, ki imajo zadnji atribut pri besedi nastavljen na *je-vejica* od primerov, ki imajo zadnji atribut nastavljen na *ni-vejice*. Iz podatkovne strukture s pozitivnimi primeri smo ustvarili deset novih podatkovnih struktur enake velikosti, ki jih bomo uporabili za učenje (*učna[i]*, pri čemer je i=1,2,3...10) in deset struktur, ki jih bomo uporabili za testiranje (*testna[i]*, pri čemer je i=1,2,3...10). V vsako testno podatkovno strukturo smo zapisali po desetino vseh primerov

iz podatkovne množice s pozitivnimi primeri. Prva tako dobi prvo desetino pozitivnih primerov, druga drugo desetino itd. Za vsak primer iz seznama pozitivnih primerov velja, da ga zapišemo v i-to testno množico, če zadosti pogoju, da je zaporedna številka primera (oz. besede) večja ali enaka i, pomnoženem z desetino velikosti podatkovne strukture s pozitivnimi primeri in hkrati manjša od vsote zmnožka i*desetina števila pozitivnih primerov in desetine števila pozitivnih primerov. S tem zagotovimo, da gre primer iz prve desetine v prvo testno množico, iz druge desetine v drugo itd. Če ne zadosti pogoju, primer zapišemo v pripadajočo učno množico (*učna[i]*). Nato z metodo random.shuffle naključno premešamo množico negativnih primerov. V učno množico zapisujemo negativne primere dokler jih je manj ali enako številu pozitivnih primerov, ki se že nahajajo v tej učni množici. To število smo deklarirali ob branju vsake vrstice, nato pa ob vsakem pisanju pozitivnega primera v učno množico povečevali njegovo vrednost za 1. Ko smo končali s pisanjem pozitivnih primerov, smo to število uporabili kot števec negativnih primerov, ki jih moramo zapisati v pripadajočo učno množico. Vse ostale negativne primere zapišemo v pripadajočo testno množico. S tem smo zagotovili, da je vsaka učna množica dobila natanko toliko negativnih primerov kot pozitivnih, vsaka testna pa je dobila vse preostale negativne. Pri tem so pozitivni primeri izbrani uravnoteženo, negativni pa naključno.

Konkretno to za naše podatkovne množice, zgrajene iz korpusa Šolar1, pomeni, da smo 209157 primerov (oz. besed s pripadajočimi atributi) razdelili na 23282 pozitivnih primerov in 185875 negativnih, nato pa z zgoraj opisano implementacijo učenja dobili deset učnih množic s po 20954 pozitivnih primerov (to so vsi pozitivni, razen desetina, ki smo jo zapisali v testno množico) in prav toliko negativnih. Dobili pa smo tudi deset testnih množic s po 2328 pozitivnih in 164921 negativnih primerov. S tem smo iz razmerja cca 1:8 med pozitivnimi in negativnimi primeri dobili učne množice z enakim številom pozitivnih in negativnih primerov ter testne množice z razmerjem cca 1:70 med pozitivnimi in negativnimi primeri.

Za podatkovne množice iz korpusa Šolar2 smo iz 728927 primerov, od tega

65356 pozitivnih in 663571 negativnih primerov, z implementacijo učenja dobili deset učnih množic s po 58821 pozitivnih in enakim številom negativnih primerov ter deset testnih množic s po 6535 pozitivnih in 604750 negativnih primerov. S tem smo iz razmerja cca 1:10 dobili razmerje med pozitivnimi in negativnimi cca 1:92. Ker se lahko zaradi tega priklic zelo poslabša, bomo testiranje opravili tudi na testnih množicah, pri katerih bomo obdržali osnovno razmerje med pozitivnimi in negativnimi primeri. To storimo tako, da v testni množici s 6535 pozitivnimi in 604750 negativnimi primeri odstranimo toliko negativnih primerov, da obdržimo osnovno razmerje. To pomeni, da vsem testnim množicam odstranimo 538399 negativnih primerov, da nam ostane 66351 negativnih primerov. Razmerje tako ostane cca 1:10.

Algorithm 1 Uravnoteženje učne množice

```
desetina ← vseh_pozitivnih_primerov/10
for id=0 to 9 do
  stevec_dodanih_pozitivnih ← 0
  for all elements in seznam_pozitivnih do
    if (position_of_element >= id*desetina) and (position_of_element <
    id*desetina+desetina) then
      add element to testna_mnozica
    else
      add element to ucna_mnozica
      stevec_dodanih_pozitivnih ← +1
    end if
  end for
  shuffle elements in seznam_negativnih
  for all elements in seznam_negativnih do
    if position_of_element <= stevec_dodanih_pozitivnih then
      add element to ucna_mnozica
    else
      add element to testna_mnozica
    end if
  end for
end for
```

Poglavje 8

Rezultati

Testiranje smo najprej opravili na korpusu Šolar1, ki je vseboval 209157 besed. Od tega je bilo 23282 pozitivnih in 185875 negativnih primerov. Rezultati testiranj so nam potrdili, da je korpus neprimeren za strojno učenje, zato smo testiranje ponovili na izboljšanem korpusu Šolar2, ki je vseboval 728927 primerov, od tega 65356 pozitivnih in 663571 negativnih.

8.1 Rezultati korpusa Šolar1

V tabeli 8.1 vidimo rezultate testiranj. Vidimo, da se je v primeru, ko ni vejice, najbolje odrezal algoritem odločitvena tabela in sicer na originalni podatkovni množici z mero F1=0,971. Skoraj enako dober rezultat (F1=0,97) smo dosegli z istim algoritmom na podatkovnih množicah Šolar1-MSD11 in Šolar1-MSD99. Vidimo, da tu ni nobene razlike pri rezultatu med podatkovnima množicama, ki različno definirata atribute za opis MSD oznak. Naslednji najboljši rezultat smo dosegli z algoritmom naključni gozdovi na podatkovni množici Šolar1-MSD11 z mero F1=0,968. Isti algoritem na podatkovni množici s podrobneje definirano MSD oznako (Šolar1-MSD99) ima mero F1=0,966. Vidimo, da tudi v tem primeru ni veliko razlike med množicama. Najboljšo natančnost (0,9986) smo dosegli z algoritmom naključni gozdovi na uravnoteženi podatkovni množici Šolar1-MSD99, najboljši

priklic (0,995) pa z metodo podpornih vektorjev na neuravnoteženi podatkovni množici Šolar1-MSD11.

V primeru, ko je vejica, se je prav tako najbolje odrezal algoritem odločitvena tabela in prav tako na originalni podatkovni množici (F1=0,723). Sledi rezultat istega algoritma na podatkovnih množicah Šolar1-MSD11 z mero F1=0,715 in Šolar1-MSD99 z mero F1=0,709. Najboljšo natančnost (0,92) smo dosegli z naključnimi gozdovi na Šolar1-MSD11, sledi metoda podpornih vektorjev (0,918) na isti množici in nato zopet metoda podpornih vektorjev, tokrat na ŠOLAR1-MSD99. Najboljši priklic (0,9112) smo dosegli z naivnim Bayesovim klasifikatorjem na uravnoteženem Šolar1-MSD11. Sledijo naključni gozdovi (0,908) na uravnoteženi Šolar1-MSD99 in nato RBF mreža (0,9034) na uravnoteženi Šolar1-MSD11.

Najboljšo klasifikacijsko točnost (Accuracy) smo zopet dosegli pri odločitveni tabeli (94,7408%) na osnovni podatkovni množici Šolar1-osnovni. Sledita rezultata istega algoritma na spremenjeni podatkovni množici Šolar1-MSD11 (94,6337%) in na podatkovni množici Šolar1-MSD99 (94,5123%), nato pa algoritem naključni gozdovi (94,0772%) na podatkovni množici Šolar1-MSD11. Najboljši AUC je dosegel algoritem naključni gozdovi (0,963) na podatkovni množici Šolar1-MSD99, sledi rezultat istega algoritma (0,955) na podatkovni množici z manj atributi za opis MSD kode, to je Šolar1-MSD11, tretji najboljši rezultat za AUC pa je dosegla metoda odločitvena tabela na Šolar1-MSD99 z AUC=0,942.

Dejstvo, da je odvzem neinformativnih atributov poslabšal rezultate in tudi to, da se od vseh algoritmov, ki smo jih uporabili, najbolje odreže odločitvena tabela, nam potrdi, da je korpus slab in neprimeren za strojno učenje. Po natančnejšem pregledu korpusa ugotovimo, da je poleg veliko stavkov, ki ne vsebujejo nobene vejice, korpus vsebuje tudi veliko napačno postavljenih vejic. Sestavlja ga veliko podobnih besedil. Prav tako je poln nepravilno tvorjenih stavkov z veliko pogovornih in iz tujine prevzetih besed.

Uravnoteženje množic ni uspelo, rezultati so se poslabšali pri vseh algoritmih in pri obeh podatkovnih množicah. Razlog za poslabšanje je morda

to, da smo z generiranjem novih testnih množic spremenili razmerje med pozitivnimi in negativnimi primeri. V originalni množici je bilo to razmerje cca 1:8, v novih učnih 1:1, v novih testnih pa cca 1:71. Ker je korpus neprimeren za strojno učenje, testiranja z ohranjenim razmerjem med pozitivnimi in negativnimi primeri nismo preizkusili. Smo pa to preizkusili na izboljšanem korpusu v nadaljevanju.

Zanimivo je, da različna izbira opisa MSD kod le malo spremeni rezultate. Za primer, kadar ni vejice, se odločitvena tabela in SMO nekoliko bolje odrežeta z večjim številom atributov za opis MSD kode, vsi ostali pa z manjšim. V primeru, kadar je vejica, se naivni Bayesov klasifikator, RBF mreža, odločitvena tabela in naključni gozdovi bolje odrežejo z manjšim številom atributov o MSD kodi, ADTree, AdaBoostM1 in SMO pa z večjim.

		Ni vejice			Je vejica			Accuracy [%]	AUC
		Natančnost	Priklic	F1	Natančnost	Priklic	F1		
Šolar1 - osnovni	NaiveBayes	0,971	0,917	0,943	0,542	0,785	0,641	90,2117	0,931
	RBFNetwork	0,954	0,947	0,951	0,601	0,639	0,619	91,2616	0,868
	ADTree	0,947	0,982	0,964	0,794	0,563	0,659	93,5097	0,904
	AdaBoostM1	0,947	0,976	0,961	0,745	0,566	0,643	93,0153	0,826
	DecisionTable	0,954	0,989	0,971	0,873	0,617	0,723	94,7408	0,89
Šolar1 - MSD11	NaiveBayes	0,975	0,851	0,909	0,41	0,825	0,548	84,8511	0,906
	RBFNetwork	0,923	0,976	0,949	0,646	0,347	0,452	90,6128	0,881
	ADTree	0,942	0,985	0,963	0,812	0,519	0,633	93,3103	0,907
	AdaBoostM1	0,948	0,946	0,947	0,576	0,582	0,579	90,5679	0,856
	DecisionTable	0,952	0,989	0,97	0,874	0,605	0,715	94,6337	0,943
	RandomForest	0,942	0,994	0,968	0,92	0,512	0,658	94,0772	0,955
	SMO	0,939	0,995	0,966	0,918	0,484	0,634	93,7731	0,739
Šolar1 - MSD11 - uravnoteženo	NaiveBayes	0,998	0,6603	0,7947	0,0365	0,9112	0,0702	66,38827	0,9055
	RBFNetwork	0,998	0,6375	0,888	0,0346	0,9034	0,0665	64,12426	0,8721
	ADTree	0,9977	0,811	0,9036	0,0605	0,8606	0,1131	81,16486	0,9085
	AdaBoostM1	0,9963	0,8528	0,9189	0,0722	0,777	0,1314	85,17402	0,8947
	DecisionTable	0,9979	0,8662	0,9271	0,0833	0,858	0,1517	86,59347	0,9377
	RandomForest	0,9977	0,869	0,9289	0,0841	0,8488	0,1525	86,88069	0,9315
	SMO	0,9972	0,8914	0,9353	0,0901	0,8396	0,1631	87,99306	0,86
Šolar1 - MSD99	NaiveBayes	0,972	0,85	0,907	0,403	0,805	0,537	84,5365	0,908
	RBFNetwork	0,903	0,986	0,943	0,584	0,158	0,249	89,374	0,864
	ADTree	0,942	0,986	0,964	0,823	0,52	0,637	93,4069	0,924
	AdaBoostM1	0,945	0,958	0,951	0,622	0,555	0,587	91,2884	0,875
	DecisionTable	0,952	0,988	0,97	0,866	0,6	0,709	94,5123	0,942
	RandomForest	0,942	0,992	0,966	0,89	0,511	0,649	93,8491	0,963
	SMO	0,94	0,994	0,966	0,914	0,496	0,643	93,8754	0,745
Šolar1 - MSD99 -uravnoteženo	NaiveBayes	0,997	0,8091	0,8936	0,0588	0,8436	0,1098	81,95738	0,9078
	RBFNetwork	0,9971	0,741	0,8492	0,0467	0,8635	0,0884	74,26707	0,8834
	ADTree	0,997	0,8523	0,9183	0,0769	0,8153	0,1398	85,1671	0,922
	AdaBoostM1	0,996	9,014	0,9463	0,1792	0,6685	0,2273	89,91653	0,9111
	DecisionTable	0,989	0,8452	0,9155	0,0756	0,8924	0,1394	84,58838	0,9362
	RandomForest	0,9986	0,8801	0,9356	0,0965	0,908	0,1746	88,04742	1,8581
	SMO	0,998	0,8776	0,934	0,1695	0,7946	0,1644	87,76007	0,8849

Tabela 8.1: Rezultati testiranj podatkovnih množic, zgrajenih s korpusom Šolar1.

8.2 Rezultati korpusa Šolar2

Oglejmo si še rezultate testiranj podatkovnih množic posodobljenega korpusa, kjer je primerov z vejicami skoraj trikrat več. Veliko stavkov iz korpusa Šolar1 je Peter Holozan [8] ročno pregledal in popravil. V tabeli 8.2 vidimo rezultate teh testiranj. Najboljši rezultati so podčrtani. Zaradi velike časovne zahtevnosti algoritmov SMO in odločitvena tabela smo morali pri teh algoritmih uporabiti zmanjšane podatkovne množice. Uporabili smo naključno izbrano desetino množice Šolar2, kar pomeni, da smo uporabili 72892 primerov. Pri vseh ostalih algoritmih smo uporabili Šolar2 originalne velikosti, to je 728927 primerov. Pri SMO smo uporabili različne parametre. Poizkusili smo, kakšen rezultat dobimo z izbiro linearnega jedra in kakšen rezultat z izbiro kvadratnega jedra. Pri vseh ostalih algoritmih smo uporabili privzete parametre.

V tabeli 8.2 vidimo, da za primer, kadar ni vejice, najboljši rezultat pri meri F1 (0,976) dobimo z naključnimi gozdovi na podatkovni množici, kjer za opis MSD kode uporabljamo po 9 znakov za vsako besedo, to je Šolar2-MSD99. Skoraj enako dober rezultat (0,975) dobimo z istim algoritmom na množici Šolar2-MSD11. Sledi metoda ADTree z mero F1=0,972 na množici Šolar2-MSD99 in mero F1=0,971 na množici Šolar2-MSD11. Najboljšo natančnost (0,999) dobimo z naključnimi gozdovi pri uravnoteženi množici Šolar2-MSD99, prav tako se tudi vsi ostali algoritmi pri vseh podatkovnih množicah odrežejo skoraj enako dobro. Najboljši priklic dobimo zopet z metodama naključni gozdovi (0,996) na množici Šolar2-MSD99 ter ADTree (0,996) na Šolar2-MSD11. Sledita jima rezultata istih algoritmov, ADtree (0,995) na množici Šolar2-MSD99 ter naključni gozdovi (0,995) na Šolar2-MSD11. Vidimo, da so za primer, kadar ni vejice, minimalne razlike v rezultatih pri množicah, ki različno definirajo atribute za opis MSD kod.

Za primer, kadar je vejica, se z mero F1=0,68 najbolje odreže algoritem naključni gozdovi na množici Šolar2-MSD11. Sledita mu rezultata algoritmov ADtree (0,638) in naključni gozdovi (0,579) na Šolar2-MSD99. Ostali

algoritmi imajo občutno slabše rezultate za mero F1. Najboljšo natančnost dosežemo z naključnimi gozdovi (0,925) na Šolar2-MSD99, nato z ADTree (0,916) na Šolar2-MSD11 in naključnimi gozdovi (0,913) na isti množici. Najboljši priklic dobimo z uporabo naivnega Bayesovega klasifikatorja (0,9042) na uravnoteženi množici Šolar2-MSD11. Sledi mu rezultat naključnih gozdov (0,8982) na uravnoteženi Šolar2-MSD99.

Najboljšo klasifikacijsko točnost (95,4554%) dosežemo z uporabo naključnih gozdov na podatkovni množici Šolar2-MSD99. Sledi mu rezultat istega algoritma (95,4281%) na množici Šolar2-MSD11. Nato sledita rezultata algoritma ADTree (94,8664% in 94,5016%) na obeh omenjenih množicah. Najboljši AUC (0,973) dobimo z naključnimi gozdovi na Šolar2-MSD99, naslednje tri najboljše rezultate (0,9573, 0,9467 in 0,943) dobimo z uporabo istega algoritma na vseh preostalih množicah. Vidimo, da najboljše rezultate za oba primera, kadar je vejica in kadar je ni, dobimo z uporabo algoritmov naključni gozdovi in ADTree.

Znotraj uravnoteženih podatkovnih množic, kjer smo obdržali prvotno razmerje med pozitivnimi in negativnimi primeri, se je pri množici ŠOLAR2-MSD99 za primer *je-vejica* najbolje odrezal algoritem naključni gozdovi z mero F1=0,6198. Vidimo, da je ohranitev prvotnega razmerja med pozitivnimi in negativnimi primeri bistveno izboljšala rezultate, saj smo dosegli višjo natančnost. Za primer ni-vejice se je na isti množici zopet najbolje odrezal algoritem naključni gozdovi z mero F1=0,9192. Na množici ŠOLAR2-MSD11, po tem ko smo jo uravnotežili in obdržali razmerje cca 1:10 med pozitivnimi in negativnimi primeri, se je za primer, kadar je vejica in prav tako za primer, kadar ni vejice, najbolje odrezal algoritem odločitvena tabela z mero F1=0,5755 za prvi primer in z mero F1=0,9351 za drugi primer. Če med seboj primerjamo meri F1 pri istem postopku uravnoteževanja in prav tako pri enakem razmerju, vendar pri različnem številu atributov za opis oblikoskladenjskih oznak, vidimo, da se je pri uravnoteženih množicah z obdržanim razmerjem najbolje odrezala množica, kjer smo uporabili 99 atributov za opis MSD oznake za primer je-vejica in množica z 11 atributi za MSD oznake za

primer ni-vejice.

Pri testiranju na množicah, ki smo jih zmanjšali na desetino originalne velikosti, vidimo, da najboljši rezultat dobimo z algoritmom odločitvena tabela. Za primer, kjer ni vejice, je to pri meri F1=0,977 na podatkovni množici, kjer MSD oznako opisujemo z 11 atributi. Za primer, kadar je vejica, pa najboljši rezultat dobimo z mero F1=0,709 pri istem algoritmu in isti podatkovni množici, torej množici, kjer oblikoskladenjsko oznako opisujemo z 11 atributi. Če primerjamo rezultata pri uporabi metode podpornih vektorjev z različnimi algoritmi, vidimo, da se bolje odreže izbira linearnega jedra. Najboljšemu rezultatu, ki smo ga dobili pri testiranju na zmanjšanih množicah, se najbolj približa algoritem, s katerim smo dobili najboljši rezultat na originalni velikosti množic, to je algoritem naključni gozdovi.

		Ni vejice			Je vejica			Accuracy [%]	AUC
		Natančnost	Priklic	F1	Natančnost	Priklic	F1		
Šolar2 - MSD99	NaiveBayes	0,979	0,84	0,904	0,333	0,813	0,473	83,7456	0,905
	RBFNetwork	0,927	0,985	0,955	0,591	0,217	0,318	91,6339	0,868
	ADTree	0,951	0,995	0,972	0,898	0,482	0,638	94,8664	0,925
	AdaBoostM1	0,951	0,97	0,961	0,62	0,489	0,547	92,7351	0,882
	RandomForest	0,956	0,996	0,976	0,925	0,536	0,579	95,4554	0,973
Šolar2 - MSD99	NaiveBayes	0,998	0,7944	0,8848	0,0429	0,85	0,0814	79,51062	0,905
- uravnoteženo	RBFNetwork	0,9978	0,7597	0,8615	0,038	0,8441	0,0725	76,06046	0,8749
	ADTree	0,998	0,829	0,906	0,0515	0,8584	0,0971	82,91366	0,9249
	AdaBoostM1	0,9976	0,8351	0,9085	0,0529	0,8005	0,0985	83,490518	0,908
	RandomForest	0,999	0,887	0,9397	0,0793	0,8982	0,1457	88,71698	0,9573
Šolar2 - MSD99	NaiveBayes	0,9817	0,7956	0,8791	0,2906	0,85	0,4332	80,05091	0,9054
- uravnoteženo z obdržanim razmerjem	RBFNetwork	0,9805	0,7603	0,8553	0,2634	0,8441	0,3994	76,78716	0,8752
	ADTree	0,9483	0,8296	0,8817	0,3927	0,8584	0,5203	83,42836	0,9251
	AdaBoostM1	0,924	0,8353	0,8651	0,3841	0,8005	0,4916	81,51685	0,9083
	DecisionTable	0,9565	0,8785	0,9136	0,4726	0,879	0,6023	87,98982	0,9436
	RandomForest	0,9589	0,8868	0,9192	0,4888	0,893	0,6198	88,70218	0,9552
Šolar2 - MSD11	NaiveBayes	0,983	0,769	0,863	0,269	0,861	0,41	77,754	0,903
	RBFNetwork	0,922	0,992	0,956	0,654	0,147	0,24	91,6551	0,87
	ADTree	0,946	0,996	0,971	0,916	0,426	0,581	94,5016	0,913
	AdaBoostM1	0,952	0,97	0,961	0,621	0,499	0,553	92,7753	0,867
	RandomForest	0,957	0,995	0,975	0,913	0,542	0,68	95,4281	0,943
Šolar2 - MSD11	NaiveBayes	0,9981	0,6368	0,7778	0,0261	0,9042	0,054	63,98065	0,9029
-uravnoteženo	RBFNetwork	0,9978	0,7096	0,8254	0,0342	0,8546	0,0652	71,11784	0,8781
	ADTree	0,998	0,8213	0,901	0,05	0,8654	0,1787	82,1834	0,9136
	AdaBoostM1	0,9968	0,8704	0,92743	0,0674	0,7683	0,1218	86,91878	0,9009
	RandomForest	0,998	0,8835	0,9375	0,0745	0,8683	0,1375	88,34154	0,9467
Šolar2 - MSD11	NaiveBayes	0,9774	0,6367	0,7734	0,1968	0,9042	0,3234	66,06097	0,9029
- uravnoteženo z obdržanim razmerjem	RBFNetwork	0,9791	0,7082	0,8171	0,2287	0,7986	0,3697	72,08435	0,8724
	ADTree	0,9841	0,8209	0,895	0,3225	0,8654	0,4698	82,49184	0,9136
	AdaBoostM1	0,9728	0,8798	0,9232	0,4052	0,7462	0,5136	86,78237	0,9004
	DecisionTable	0,9845	0,8904	0,9351	0,436	0,8578	0,578	88,7534	0,9492
	RandomForest	0,9867	0,8838	0,9326	0,4275	0,8802	0,5755	88,35771	0,9507
ZMANJŠANE MNOŽICE									
1/10 Šolar2 - MSD99	DecisionTable	0,958	0,995	0,976	0,918	0,653	0,698	95,5893	0,939
	SMO z linearnim jedrom	0,955	0,996	0,975	0,928	0,529	0,674	95,3657	0,762
	SMO s kvadratnim jedrom	0,927	1	0,962	0,996	0,21	0,347	92,8482	0,605
1/10 Šolar2 - MSD11	DecisionTable	0,959	0,995	0,977	0,92	0,577	0,709	95,7197	0,94
	SMO z linearnim jedrom	0,954	0,997	0,975	0,942	0,52	0,67	95,3671	0,758
	SMO s kvadratnim jedrom	0,943	0,992	0,967	0,834	0,392	0,534	93,799	0,692

Tabela 8.2: Rezultati testiranj podatkovnih množic, ki so bile zgrajene z izboljšanim korpusom Šolar2.

Poglavje 9

Zaključek

Želeli smo se naučiti avtomatskega postavljanja vejic za slovenski jezik. Trenutno za slovenščino obstajajo le programi, ki avtomatsko postavljajo vejice v besedilo s pomočjo sprogramiranih pravil. Ker je slovenščina izjemno morfološko bogat jezik in je potrebno za postavljanje vejic upoštevati nemalo pravil, ki jih je težko ali nemogoče programsko realizirati, smo želeli postavljanje vejic izboljšati s strojnim učenjem. V pomoč strojnemu učenju smo dodali atribute, generirane s pomočjo pravil, ki jih za postavljanje vejic uporablja eden od obstoječih programov.

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 njegovo časovno zahtevnost ter porabo virov. Nato smo dodali nove atribute, generirane na podlagi pravil, ki jih za postavljanje vejic s pomočjo pravil uporablja orodje LanguageTool. Prav tako smo dodali atribute, ki smo jih generirali 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 samo prvo črko oblikoskladenjske oznake in s tem obdržali 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. Tako smo neinformativno zalogo vrednosti z več kot 930 različnimi oblikoskladenjskimi oznakami nadomestili z novo, bolj smiselno. Poskusili smo izboljšati rezultate z lastno implementacijo učenja, ki bi zagotovilo prednosti prečnega preverjanja in podvzorčenja.

Ugotovili smo, da je najbolj pomemben kakovosten korpus. Najbolj uspešne metode strojnega učenja so algoritmi naključna drevesa, alternirajoče odločitveno drevo ter odločitvena tabela. Rezultati pri najboljših algoritmih so zelo podobni pri obeh podatkovnih množicah, kjer smo različno generirali atribute o MSD oznakah. Za primer, kadar je vejica, se je najbolje odrezal algoritem naključni gozdovi na množici, kjer smo za opis MSD oznak uporabili 11 atributov. Za primer, kadar ni vejice, s je najbolje odrezal isti algoritem, vendar na množici, kjer smo za opis atributov uporabili 99 atributov. Vendar pa se rezultat učenja z 11 atributi za opis MSD le minimalno razlikuje od najboljšega rezultata (razlika je 0,001). To zelo verjetno pomeni, da ob uporabi primernega korpusa zadostuje, če za opis oblikoskladenjskih oznak uporabimo le besedno vrsto (razen pri veznikih, kjer uporabimo dve črki za zapis atributa).

Zanimivo bi bilo videti, kako se strojno učenje odreže, če podatkovnim množicam dodamo atribute, generirane glede na vsa pravila za postavljanje vejic v slovenščini. To bi bilo mogoče implementirati, če bi bila pravila za postavljanje vejic v slovenščini v osnovi podana jasno in nedvoumno, bolje definirana glede na povezave besed znotraj stavkov ter glede na oblikoskladenjske oznake besed znotraj povedi. Zanimalo bi nas tudi, kako bi se obnesla implementacija tretjega načina definiranja MSD oznak, ki smo ga opisali v razdelku 5.4.3 in zahteva več računskih zmogljivosti.

Za obstoj jezika, ki ga govori ali razume samo 2,5 milijona ljudi, je izjemno pomembno razvijanje ter konstantno nadgrajevanje jezikovnih tehnologij, ki bodo omogočale, da bo slovenščina tudi v prihodnosti enakovreden jezik ostalim evropskim in drugim jezikom. Za kvalitetno procesiranje naravnega jezika so izjemno pomembne kompleksnejše jezikovne tehnologije, kot

so lematizator, označevalnik in skladenjski razčlenjevalnik. Temelj vseh pa je kvaliteten in po možnosti homogen korpus, ki mora biti sestavljen iz dobrih in (večkrat) lektoriranih besedil s strani strokovnjakov za jezik.

Literatura

[1] *Oblikoslovni označevalnik Obeliks, spletna različica.* Amebis d.o.o., `http://www.slovenscina.eu/tehnologije/oznacevalnik`, 2010. [Dostop: 06/06/2015].

[2] *Skladenjski razčenjevalnik.* Amebis d.o.o., `http://www.slovenscina.eu/tehnologije/razclenjevalnik`, 2010.

[3] *Projekt JOS: jezikoslovno označevanje slovenskega jezika.* `http://nl.ijs.si/jos/`, Institut Jožef Stefan, 2013. [Dostop: 06/06/2015].

[4] *Projekt Sporazumevanje v slovenskem jeziku.* Amebis d.o.o., `http://www.slovenscina.eu`, 2013. [Dostop: 06/06/2015].

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

[6] Jože Toporišič et. al. *, SLOVENSKI pravopis 1, Pravila.* Slovenska akademija znanosti in umetnosti in Inštitut za slovenski jezik Frana Ramovša ZRC SAZU, DZS, d.d., Ljubljana, 1994.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. *Programsko orodje za podatkovno rudarjenje Weka.* University of Waikato, `http://www.cs.waikato.ac.nz/ml/weka`.

[8] Peter Holozan. *Kako dobro programi popravljajo vejice v slovenščini.* Zbornik osme konference Jezikovne tehnologije, Institut Jožef Stefan, Ljubljana.

[9] Peter Holozan. *Uporaba strojnega učenja za postavljanje vejic v slovenščini.* Revija Uporabna informatika, št. 4 - letnik XXI, 2013.

[10] Ian H.Witten and Eibe Frank. *Data mining : practical machine learning tools and techniques – 2nd ed.* Elsevier Inc., 2005.

[11] Igor Kononenko in Marko Robnik Šikonja. *Inteligentni sistemi.* Fakulteta za računalništvo in informatiko, Založba FE in FRI, Ljubljana, 2010.

[12] Helena Dobrovoljc in Nataša Jakop. *Sodobni pravopisni priročnik med normo in predpisom.* Inštitut za slovenski jezik Frana Ramovša ZRC SAZU, Ljubljana, 2011.

[13] Uroš Kosič. *Mera podobnosti na podlagi naključnih gozdov, diplomsko delo.* Fakulteta za računalništvo in informatiko, Ljubljana, 2012.

[14] Simon Krek. *Slovenski jezik v digitalni dobi.* Zbirka Bela Knjiga, Institut Jožef Stefan, Ljubljana, 2012.

[15] Skupnost LanguageTool. *Portal Slogovni priročnik pod okriljem projekta Sporazumevanje v slovenskem jeziku.* `http://slogovni.slovenscina.eu`, 2015. [Dostop: 06/06/2015].

[16] Skupnost LanguageTool. *Spletna stran skupnosti LanguageTool.* `http://community.languagetool.org`, 2015. [Dostop: 06/06/2015].

[17] Jožef Skaza. *EPIS - pravopisni priročnik.* Dobrna: Eknjiga, Ljubljana, 2000.

[18] Leo Breiman Statistics and Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[19] Inštitut za slovenski jezik Frana Ramovša ZRC SAZU. *Slovar slovenskega knjižnega jezika, spletna različica.* http://bos.zrc-sazu.si/sskj.html, 2000. [Dostop: 06/06/2015].