Uporaba strojnega učenja za postavljanje vejic v slovenščini

Peter Holozan, Amebis, d. o. o., Kamnik, Bakovnik 3, 1241 Kamnik
peter.holozan@amebis.si

Izvleček

Za slovenščino obstajata dva programa, ki postavljata vejice v besedilo s pomočjo pravil, ni pa še bilo preizkušeno strojno učenje, ki je že bilo uspešno uporabljeno za postavljanje vejic v drugih jezikih. Za preizkušanje je bil uporabljen seznam primerov z napakami pri vejicah iz korpusa Šolar (209.156 besed). V prvem delu je bilo strojno učenje uporabljeno za problem iskanja vseh vejic, doseženi rezultat je primerljiv z drugimi jeziki (natančnost 0,861 in priklic 0,641) in s programoma s pravili, najboljši rezultat je bil dosežen z uporabo skladenjskega analizatorja, lematizator, oblikoslovni označevalnik in skladenjski analizator pa so bili naučeni z učno množico brez vejic, uporabljen je bil klasifikator ADTree. Preizkušena je bila še uspešnost popravljanja realnih napak v besedilu, pri čemer je bil rezultat slabši (natančnost 0,676 in priklic 0,545 za manjkajoče vejice).
Ključne besede: postavljanje vejic, popravljanje napačnih vejic, slovenščina, strojno učenje, ADTree.

Abstract

Using Machine Learning for Comma Placing in Slovene

For the Slovene language there currently exist two software solutions able to place commas into text using rules, however Machine Learning that has already been successfully used for comma placing in other languages has never been tried with Slovene. For testing, a list of examples with comma mistakes from the corpus Šolar (209156 words), was used. In the first part of the experiment machine learning was used for searching all commas, the obtained result is comparable with other languages (precision 0.861 and recall 0.641) and the rule-based programs. The best result was achieved using the syntax analyser. The lemmatiser, the PoS tagger and the syntax analyser were trained on a corpus without commas, the ADTree classifier was used. Real comma mistakes were also tested but the results were worse (precision 0.676 and recall 0.545 for missing commas).
Key words: comma placing, comma error correction, Slovene, machine learning, ADTree.

1 UVOD

Program, ki bi pravilno postavljal vejice v besedilo, ni uporaben le za pisce, ki tipkajo besedila in pri tem spregledajo kakšno vejico (postavljanje vejic povzroča hude težave celo bodočim učiteljem na razredni stopnji (Šek Mertük, 2011)), temveč tudi za druge namene. Pravilno postavljene vejice tako npr. izboljšajo oblikoslovno označevanje besedil (Hillard idr., 2006), pomembne pa so tudi pri sistemih za razpoznavo govora, ki le iz govora ne morejo pravilno postaviti vejic (Huang & Zweig, 2002).

Za slovenščino že obstajata dva programa (Besana[1] in LanguageTool[2]), ki postavljata manjkajoče vejice; oba temeljita na ročno napisanih pravilih (Holozan, 2012). Nihče pa za slovenščino še ni preizkusil, kako uspešne so pri tem statistične metode, ki uporabljajo strojno učenje iz primerov za izpeljavo pravil za vejice. Strojno učenje zahteva veliko število primerov, iz katerih lahko izpelje pravila; taki primeri napačne oz. pravilne rabe vejic so zdaj na voljo v korpusu Šolar, v katerem so zbrana besedila, ki so jih napisali učenci in dijaki, skupaj z učiteljskimi popravki.

2 PREDHODNE RAZISKAVE

Strojno učenje je bilo že večkrat uporabljeno za učenje postavljanja vejic v drugih jezikih, večinoma pa so raziskovali problem, ko je treba v besedilo postaviti vse vejice (oz. nekateri celo vsa ločila), kar je pomembno predvsem pri sistemih za razpoznavo govora (Huang & Zweig, 2002).

Beeferman idr. (1998) so preizkušali postavljanje vejic v angleščini s pomočjo skritega markovskega modela in z uporabo Viterbijevega algoritma.

Hardt (2001) je preizkušal postavljanje vejic v danščini, in sicer z uporabo Brillovega označevalnika, vendar se je omejil le na ugotavljanje odvečnih vejic, pri čemer so bile odvečne vejice dodane naključno.

[1] http://besana.amebis.si
[2] http://www.languagetool.org/

Zhang idr. (2002) so preizkušali strojno učenje za vejice v angleščini in nemščini, in sicer z odločitvenimi drevesi z uporabo skladenjskih podatkov.

Shieber in Tao (2003) sta preizkušala postavljanje vejic za angleščino; pomembna je njuna ugotovitev, da je smiselno naučiti statistični označevalnik na učnem korpusu brez vejic.

Alegria idr. (2006) so preizkušali strojno učenje v baskovščini. Uporabili so program WEKA[3] in preizkušali različne metode strojnega učenja.

Israel idr. (2012) so se ob problemu postavljanja vseh vejic v angleščini lotili tudi problema popravljanja napačnih (manjkajočih in odvečnih) vejic v besedilu.

Programa za postavljanje vejic v slovenščini je preizkusil Holozan (2012), in to za problem, ko je treba popraviti napačne vejice v besedilu. Uporabljen je bil vzorec, narejen iz korpusa Šolar, ki vsebuje napake, ki so jih naredili učenci osnovnih in srednjih šol.

3 ZASNOVA POSKUSA

Namen poskusa je preizkusiti metode strojnega učenja v slovenščini, in sicer najprej za problem postavljanja vseh vejic (na kar je bila osredinjena do zdaj večina tujih raziskav in kar je uporabno pri razpoznavi govora), potem pa še za problem popravljanja napačnih vejic (kar je uporabno v slovničnih pregledovalnikih, ki tako pomagajo piscem besedil postavljati vejice).

Osnova ideja poskusa postavljanja vseh vejic je povzeta po Alegria idr. (2006) in je taka, da uporabimo korpus s pravilno postavljenimi vejicami, ga oblikoskladenjsko označimo, lematiziramo in skladenjsko razčlenimo (pri čemer je treba upoštevati, da pri praktični uporabi nimamo vejic vnaprej, zato je treba preizkusiti označevanje tudi brez vejic, na kar sta opozorila že Shieber in Tao (2003), medtem ko Alegria idr. (2006) tega niso posebej preizkušali). Vsako besedo z določenim okoliškim oknom pretvorimo v seznam atributov in dodamo atribut, ali ji sledi vejica (ta atribut je potem razred pri klasifikacijskem problemu). Tako zapisane besede uvozimo v program za strojno učenje, v katerem izvedemo eksperimente.

Enako kot pri Alegria idr. (2006) je bil uporabljen program WEKA, ki ima vgrajeno veliko klasifikatorjev. Preizkušeno je bilo večje število klasifikatorjev, potem pa izbranih nekaj najboljših (pri čemer smo upoštevali, da so čim bolj različni), ki so bili potem uporabljeni v nadaljnjih preizkusih, v katerih so bili preizkušeni različni atributi, velikost okna, vpliv označevanja in parametri klasifikatorja.

Za preizkušanje je bilo uporabljeno desetkratno prečno preverjanje, pri čemer primere razdelimo na deset delov, devet delov uporabimo za učenje, preostali del pa za preizkušanje, kar ponovimo desetkrat z različnim delom za preizkušanje in izračunamo povprečni priklic in natančnost.

Za primerjavo sta bila na isti nalogi preizkušena še Besana in LanguageTool.

Drugi poskus je prenos ugotovitev iz prvega poskusa v popravljanje napačnih vejic in primerjava s programoma Besana in LanguageTool. Preizkušanje v tem poskusu je namreč bolj zapleteno, zato je najboljšo kombinacijo za strojno učenje laže poiskati pri problemu iskanja vseh vejic in jo potem uporabiti še pri popravljanju napačnih vejic.

3.1 Korpus

V raziskavi je bila uporabljena posodobljena verzija korpusa (popravljenih je bilo nekaj napačnih vejic), ki je bil uporabljen v Holozan (2012). To je podkorpus, narejen iz korpusa Šolar,[4] ki je zbirka besedil, ki so jih napisali učenci v šoli, in ki vključuje tudi popravke napak. Ta podkorpus vsebuje le povedi z napačnimi vejicami (bodisi manjkajočimi bodisi odvečnimi), pri čemer so mesta manjkajočih vejic označena z znakom ¤, odvečne vejice pa so nadomeščene z znakom ÷; velikost tega podkorpusa je 209.156 besed (vključno z ločili, razen vejic), v podkorpusu je 11.892 pravilno postavljenih vejic, 11.399 manjkajočih vejic in 2709 odvečnih vejic.

Za problem postavljanja vseh vejic (in tudi za učenje pri popravljanju vejic) je bil korpus predelan tako, da so bile vse vejice popravljene (znaki ¤ zamenjani z vejicami, znaki ÷ pa pobrisani), s čimer je bil narejen korpus s pravilno postavljenimi vejicami.

Predvsem za ta problem postavljanja vseh vejic (pa tudi za realno natančnost pri popravljanju napačnih vejic, čeprav je tu težava, da je ta odvisna od deleža napak v korpusu in se je tako težko odločiti, katera besedila vsebujejo povprečno število napačnih vejic) bi bilo sicer bolje uporabiti korpus, ki bi vseboval tudi povedi s pravilno postavljenimi vejicami, vendar takega korpusa ob izvajanju poskusa ni bilo

[3] http://www.cs.waikato.ac.nz/ml/weka/

[4] http://www.slovenscina.eu/korpusi/solar

na voljo. Tudi popravki v korpusu Šolar namreč niso povsem natančni, zato so bili primeri v podkorpusu ročno preverjeni in ustrezno popravljeni.

Druga možnost za postavljanje vseh vejic bi bila uporaba dela katerega od obstoječih korpusov (npr. Gigafide),[5] vendar se tu postavi vprašanje, kako natančno so lektorirana besedila, vključena v korpuse. Se je pa za to rešitev odločila večina tujih raziskovalcev (tudi Alegria idr. (2006), ki so med drugim uporabili časopisna besedila).

3.1.1 Označevanje

Tako Hardt (2001) kot tudi Alegria idr. (2006) so eksperimentirali z označenimi korpusi, saj lahko pravilne oblikoskladenjske oznake in poznavanje strukture povedi pomagajo pri postavljanju vejic.

Zato je bilo tudi za slovenščino uporabljeno označevanje, in sicer oblikoslovni označevalnik in lematizator Obeliks[6] ter skladenjski razčlenjevalnik,[7] ki sta bila razvita v okviru projekta Sporazumevanje v slovenskem jeziku.[8]

Pri poskusih za baskovščino in danščino ni posebej specificirano, ali so označevali korpus s pravilno ali z napačno postavljenimi vejicami, zdi se, da so uporabili različico s pravilno postavljenimi vejicami. Ker pa pravilnost vejic lahko vpliva na natančnost označevalnika (Hillard idr., 2006) in ker pri praktični uporabi (npr. popravljanju napačnih vejic v besedilu) ni mogoče vnaprej imeti pravilno postavljenih vejic, sta bili preizkušeni obe različici označevanja.

3.2 Ocenjevanje rezultatov

Za ocenjevanje rezultatov sta bili uporabljeni metriki natančnost (delež pravilno postavljenih vejic) in priklic (delež odkritih manjkajočih vejic) ter metrika F1, ki je harmonična sredina natančnosti in priklica in se izračuna kot 2 * natančnost * priklic / (natančnost + priklic). Problem postavljanja vejic predstavimo z razredom, ki pove, ali neki besedi sledi vejica. V korpusu je 23.291 mest, kjer mora biti vejica, vejica torej mora biti za 11,1 odstotka besed, večinski razred pa je, da besedi ne sledi vejica, kar je v 88,9 odstotka primerov.

Program WEKA je rezultate izračunal tako za primer, ko ni vejice, kot za primere, ko vejica je. Ker je cilj postaviti vejice v besedilo, je zanimiv predvsem rezultat pri primerih, ko vejica je, saj nam to pove, koliko manjkajočih vejic bi odkrila metoda. Natančnost je pomembnejša od priklica, ker npr. pri slovničnem pregledovalniku nočemo preveč lažnih opozoril, seveda pa tudi priklic ne sme biti premajhen (npr. vsaj 50 %), da je metoda uporabna, zato je pomemben tudi rezultat za F1, ki ga prav tako izračunava program WEKA.

Rezultati so izračunani na besede, ker je beseda (z okoliškim oknom) element pri strojnem učenju.

Referenčna vrednost uspešnosti je rezultat, ki ga dosežeta programa, ki postavljata vejice s pomočjo pravil. Programa sicer nista namenjena za reševanje problema, ko je treba postaviti vse vejice, vendar je vseeno zanimivo videti, kako dobro poiščeta vse vejice.

3.3 Priprava podatkov

Program WEKA potrebuje podatke v formatu ARFF, v katerem glavi z opisom atributov sledi podatkovni del, v katerem vsaka vrstica predstavlja en primer. Rezultat označevanja besedil je v formatu XML-TEI,[9] zato je bil napisan za pretvorbo program v Perlu. Ta za vsako besedo določi atribute, potem pa pri izvozu v ARFF ob sami besedi izpiše še atribute za prejšnje in naslednje besede glede na nastavitev okna (privzeta vrednost je –5 +5, torej pet besed spredaj in pet besed zadaj, s čimer so začeli tudi Alegria idr. (2006)). Vejice niso besede, ampak le atribut *je-vejica* na besedi neposredno pred vejico. Ta atribut je potem uporabljen kot razred pri strojnem učenju.

Program za izvoz v ARFF izvozi vse atribute (razen podatka o obstoju vejice) kot nize, s čimer pa večina klasifikatorjev ne zna delati, zato jih je treba najprej spremeniti v nominalne atribute, pri čemer je pri definiciji atributa našteta zaloga možnih vrednosti. V ta namen je bil v programu WEKA uporabljen filter StringToNominal.

3.3.1 Atributi

Osnovni atributi za vsako besedo so oblika (sama beseda, taka kot je napisana, npr. mize), lema (osnovna oblika besede, npr. miza) in oblikoskladenjska oznaka (ali MSD – morpho-syntactic descriptor, npr. Sozer) po oblikoskladenjskih specifikacijah JOS,[10] ki pove besedno vrsto, podatke o sklonu, spolu, številu

[5] http://www.gigafida.net
[6] http://www.slovenscina.eu/tehnologije/oznacevalnik
[7] http://www.slovenscina.eu/tehnologije/razclenjevalnik
[8] http://www.slovenscina.eu
[9] http://www.tei-c.org/Guidelines/P5/
[10] http://nl.ijs.si/jos/msd/html-sl/index.html

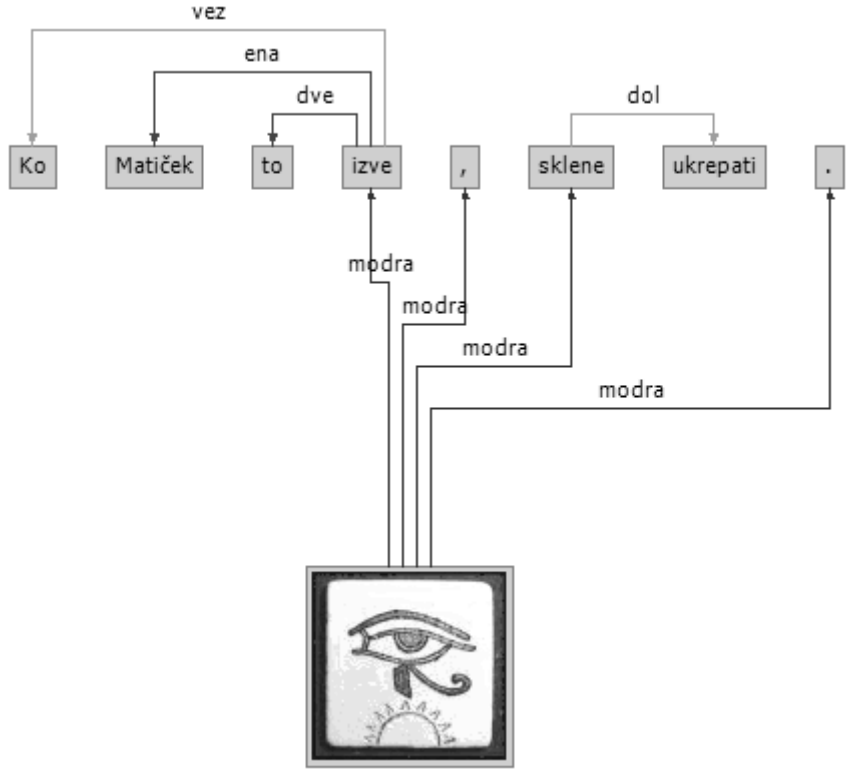
ipd. Ker ločila nimajo oblikoskladenjskih oznak, jim je bila pripisana oznaka Y, da jih lahko obravnavamo enako kot besede. Neobstoječim besedam znotraj okna so bili vsi atributi nastavljeni na *, vsak stavek je enota zase in okno ne sega na sosednje stavke.

Atributi so našteti tako, da so najprej atributi za samo besedo (položaj 0), temu sledijo atributi za predhodne besede (od -1 do -5) in temu atributi za naslednje besede (od +1 do +5).

Po celotnem MSD je bil narejen še poskus z delnim MSD, v katerem so atributi ločeno prvi znak MSD, drugi znak MSD in pri samostalnikih še sklon.

Delni MSD2 je bil poskus, kako čim bolj prenesti informacije iz MSD in se izogniti uporabi celotnega MSD (zaradi predpostavke, da veliko število različnih MSD lahko ovira učenje). Vsak MSD bil razdeljen v dva atributa, prvi je kot prvo črko vseboval besedno vrsto, druga črka pa je bila vrsta pri posamezni besedni vrsti (pri samostalnikih, pridevnikih, glagolih, zaimkih, številnikih in veznikih). Drugi atribut je vseboval sklon pri samostalnikih, pridevnikih, zaimkih, predlogih in številnikih, sicer pa **.

Naslednji poskus je bil uporaba podatkov skladenjskega razčlenjevalnika, pri katerem pa je rezultate težje pretvoriti v atribute kot pri oblikoslovnem označevalniku in lematizatorju, saj so rezultat skladenjskega razčlenjevalnika povezave, ki gradijo drevo.



Slika 1: **Rezultat skladenjskega razčlenjevalnika**

Slika 1 kaže rezultat skladenjske razčlembe za poved »Ko Matiček to izve, sklene ukrepati«. Za postavljanje vejic so pomembne predvsem povezave[11] »vez«, ki kaže na veznike, »modra«, ki kaže na del povedi, in rdeče povezave »ena«, »dve«, »tri« in »štiri«, ki kažejo na osebke, predmete in prislovna določila, pri čemer nas pri modrih in rdečih povezavah zanima začetek bloka, zato mora upoštevati še vse naslednje povezave, da pridemo do začetka tega bloka.

Rezultat razčlenjevalnika (skupaj z rezultatom lematizatorja in oblikoskladenjskega analizatorja) je zapisan v formatu XML, kot prikazujemo na sliki 2 (izpuščene so značke »<S />«, ki označujejo presledke).

[11] Vsi tipi povezav so opisani na http://www.slovenscina.eu/tehnologije/razclenjevalnik.

```
<s xml:id="0.0">
    <w lemma="ko" msd="Vd" xml:id="0.0.1">Ko</w>
    <w lemma="Matiček" msd="Slmei" xml:id="0.0.2">Matiček</w>
    <w lemma="ta" msd="Zk-set" xml:id="0.0.3">to</w>
    <w lemma="izvedeti" msd="Ggdste" xml:id="0.0.4">izve</w>
    <c xml:id="0.0.5">,</c>
    <w lemma="skleniti" msd="Ggdste" xml:id="0.0.6">sklene</w>
    <w lemma="ukrepati" msd="Ggnn" xml:id="0.0.7">ukrepati</w>
    <c xml:id="0.0.8">.</c>
    <links>
        <link afun="vez" dep="0.0.1" from="0.0.4" />
        <link afun="ena" dep="0.0.2" from="0.0.4" />
        <link afun="dve" dep="0.0.3" from="0.0.4" />
        <link afun="modra" dep="0.0.4" from="0.0.0" />
        <link afun="modra" dep="0.0.5" from="0.0.0" />
        <link afun="modra" dep="0.0.6" from="0.0.0" />
        <link afun="dol" dep="0.0.7" from="0.0.6" />
        <link afun="modra" dep="0.0.8" from="0.0.0" />
    </links>
    </s>
```

Slika 2: **Zapis označevanja in skladenjske razčlembe v formatu XML**

Slika 2 je primer, zapisan v formatu XML, ki je rezultat označevanja in skladenjskega razčlenjevalnika. Značke "<s" so povedi, značke "<w" besede, značke "<c" ločila in značke "<link" skladenjske povezave.

Za skladenjske atribute so bili izbrani (vrednost je 1, če je trditev resnična, oz. 0, če ni):

- je vez: beseda, na katero kaže povezava "vez";
- začetek modrega bloka: prva beseda v bloku, na katerega kaže povezava "modra";
- začetek rdečega bloka: prva beseda v bloku, na katerega kaže rdeča povezava.

```
'Ko','ko','Vd','1','0','1','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','Mati
ček','Matiček','Slmei','0','1','0','to','ta','Zk set','0','1','0','izve','izvedeti','Ggdste','0','0','0','sklene','skle
niti','Ggdste','0','0','0','ukrepati','ukrepati','Ggnn','0','0','0',ni vejice 'Matiček','Matiček','Slmei','0','1',
'0','Ko','ko','Vd','1','0','1','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','to','ta','Zk set','
0','1','0','izve','izvedeti','Ggdste','0','0','0','sklene','skleniti','Ggdste','0','0','0','ukrepati','ukrepati','Gg
nn','0','0','0','.','.','Y','1','0','0',ni vejice 'to','ta','Zk set','0','1','0','Matiček','Matiček','Slmei','0','1','0','Ko','k
o','Vd','1','0','1','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','*','izve','izvedeti','Ggdste','0','0','0','sklene','
skleniti','Ggdste','0','0','0','ukrepati','ukrepati','Ggnn','0','0','0','.','.','Y','1','0','0','*','*','*','*','*','*',ni vejice
'izve','izvedeti','Ggdste','0','0','0','to','ta','Zk set','0','1','0','Matiček','Matiček','Slmei','0','1','0','Ko','ko','V
d','1','0','1','*','*','*','*','*','*','*','*','*','*','*','*','sklene','skleniti','Ggdste','0','0','0','ukrepati','ukrepati','Ggnn'
,'0','0','0','.','.','Y','1','0','0','*','*','*','*','*','*','*','*','*','*','*','*',je vejica
```

Slika 3: **Zapis začetka zgornjega primera v formatu ARFF z oknom –5+5**

Slika 3 kaže, kako je začetek zgornjega primera zapisan v formatu ARFF, ki ga zna brati programski paket WEKA.

4 PREIZKUŠANJE

Za problem, ko je treba postaviti vse vejice besedilu, je bilo narejenih več preizkusov, da bi našli najboljšo kombinacijo klasifikatorja, atributov, velikosti okna, načina označevanja in parametrov klasifikatorja.

Zaradi velikega števila možnih kombinacij ni bilo mogoče preizkusiti vseh, ampak se je po posameznih delnih preizkusih ožil izbor (na podlagi natančnosti in delno tudi F1 na mestih, kjer so vejice), katere kombinacije je najbolj smiselno preizkušati naprej.

4.1 Izbiranje klasifikatorja in vpliv velikosti korpusa

Preizkušeno je bilo večje število klasifikatorjev, ki jih podpira program WEKA, vsi so bili uporabljeni s privzetimi parametri.

Tabela 1: **Šolar, celotni MSD, brez skladenjskih atributov**

		Ni vejice			Je vejica		
	Klasifikator	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
100 %	ZeroR	0,889	1	0,941	0	0	0
	HyperPipes	0,892	0,989	0,938	0,340	0,045	0,079
	J48						
	NaiveBayes	0,965	0,947	0,956	0,632	0,726	**0,676**
	Decision Table	0,948	0,986	0,966	**0,830**	0,565	0,672
	BayesNet	0,973	0,918	0,945	0,549	0,797	0,65
	Stacking	0,889	1	0,941	0	0	0
	VFI	0,919	0,928	0,923	0,347	0,345	0,359
	ADTree	0,945	0,977	0,961	0,751	0,546	0,632
	RBFNetwork	0,948	0,975	0,961	0,740	0,570	0,644
	AdaBoostM1	0,928	0,985	0,956	0,768	0,386	0,514
	NaiveBayesUpdateable	0,965	0,947	0,956	0,632	0,726	0,676
	DecisionStump	0,928	0,985	0,956	0,768	0,386	0,514
50 %	ADTree	0,943	0,979	0,961	0,761	0,533	0,627
	DecisionStump	0,927	0,985	0,955	0,766	0,384	0,511
25 %	J48	0,89	1	0,942	0	0	0
	NaiveBayes	0,925	0,992	0,958	0,848	0,351	0,497
	Decision Table	0,948	0,984	0,966	0,817	0,563	**0,666**
	Stacking	0,89	1	0,942	0	0	0
	ADTree	0,944	0,978	0,961	0,746	0,531	0,620
	LWL	0,931	0,986	0,958	0,78	0,409	0,537
	RBFNetwork	0,914	0,995	0,953	**0,854**	0,245	0,381
	AdaBoostM1	0,929	0,986	0,956	0,773	0,389	0,517
	NaiveBayesUpdateable	0,925	0,992	0,958	0,848	0,351	0,497
	DecisionStump	0,929	0,986	0,956	0,773	0,389	0,517

Preizkušeno je bilo še več klasifikatorjev, pri katerih pa izračunavanje bodisi ni uspelo (SMO, LibSVM, HNB, MultilayerPerceptron, Bagging, FT, Prism, J48) bodisi je trajalo predolgo (LWL, KStar, Id3, NBTree, BFTree, LADTree, SimpleCart, REPTree). Je pa seveda mogoče, da bi se dala katera od teh metod usposobiti z ustreznimi parametri klasifikatorja, ustrezno zmanjšanim oknom, manjšim korpusom ali več potrpljenja (počakati nekaj dni na rezultat).

Če želimo iskati manjkajoče vejice, nas zanima predvsem natančnost pri možnosti, ko vejica je, vendar seveda tudi priklic ne sme biti preslab, tako da iščemo tudi dober F1.

Kot uspešni klasifikatorji so se pokazali Decision Table, NaiveBayes, ADTree in RBFNetwork. Slaba

stran klasifikatorja Decision Table pa je, da je preizkušanje neuporabno počasno, zato je bil pri nadaljnjem preizkušanju namesto njega uporabljen AdaBoostM1 (klasifikatorji za nadaljnje preizkušanje so bili namerno izbrani tako, da pripadajo različnim skupinam klasifikatorjem in niso preveč podobni med seboj).

Klasifikatorji, ki niso bili uspešni na celotnem korpusu, so bili preizkušeni še na zmanjšanem korpusu, da bi morda bili uspešni tam (nekateri klasifikatorji pa so bili ponovljeni za primerjavo, koliko vpliva velikost korpusa).

Klasifikator J48, ki je bil uporabljen v Alegria idr. (2006), se je uspešno izvedel le pri 25 odstotkih primerov (vendar je tudi tu uporabil le večinski razred in je dal povsod odgovor, da ni vejice), pri 50 odstotkih in polnem korpusu preizkus ni bil uspešen. Klasifikator SMO pa sploh ni bil uspešen niti pri 25 odstotkih. Ta rezultat je presenetljiv, Alegria idr. (2006) so uporabljali korpus s 130.000 besedami za preizkuse (100.000 besed za učenje in 30.000 za preizkušanje) in malo manjše okno (–5+2), kar pomeni, da 25 odstotkov korpusa v našem poskusu ne bi smelo pomeniti težave. Zato bi bilo smiselno to še enkrat preizkusiti v prihodnosti z ustrezno nastavitvijo parametrov klasifikatorjev.

Manjšanje korpusa je poslabšalo rezultate pri klasifikatorjih NaiveBayes in RBFNetwork, na klasifikatorje Decision Table, ADTree in AdaBoostM1 pa ni bistveno vplivalo.

4.2 Atributi

Vprašanje je, kateri podatki so pomembni, da jih dodamo kot atribute. Osnovna podatka sta sama beseda in lema besede, narejen pa je bil poskus, kako uporabiti oblikoskladenjske oznake (MSD).

Tabela 2: **Šolar**

		Ni vejice			Je vejica		
	Klasifikator	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
Celotni MSD	NaiveBayes	0,965	0,947	0,956	0,632	0,726	0,676
	RBFNetwork	0,948	0,975	0,961	0,740	0,57	0,644
	ADTree	0,945	0,977	0,961	0,751	0,546	0,632
	AdaBoostM1	0,928	0,985	0,956	0,768	0,386	0,514
Delni MSD	NaiveBayes	0,971	0,924	0,947	0,563	0,781	0,654
	RBFNetwork	0,958	0,946	0,952	0,607	0,667	0,636
	ADTree	0,944	0,984	0,964	0,811	0,537	0,646
	AdaBoostM1	0,943	0,968	0,955	0,677	0,53	0,595
Brez oblik	NaiveBayes	0,975	0,904	0,938	0,515	0,812	0,630
	RBFNetwork	0,957	0,943	0,950	0,593	0,662	0,626
	ADTree	0,944	0,984	0,964	0,811	0,537	0,646
	AdaBoostM1	0,943	0,968	0,955	0,677	0,53	0,595
Delni MSD2	NaiveBayes	0,967	0,935	0,951	0,592	0,749	0,661
	RBFNetwork	0,953	0,958	0,955	0,648	0,620	0,634
	ADTree	0,930	0,989	0,959	**0,827**	0,402	0,541
	AdaBoostM1	0,928	0,985	0,956	0,768	0,386	0,514
MSD + delni MSD2	NaiveBayes	0,972	0,925	0,948	0,568	0,784	0,658
	RBFNetwork	0,960	0,949	0,954	0,625	0,683	0,653
	ADTree	0,930	0,989	0,959	0,827	0,402	0,541
	AdaBoostM1	0,928	0,985	0,956	0,768	0,386	0,514
MSD + skladnja	NaiveBayes	0,973	0,920	0,946	0,555	0,793	0,653
	RBFNetwork	0,956	0,949	0,953	0,616	0,652	0,634
	ADTree	0,950	0,983	0,966	0,815	0,588	**0,683**
	AdaBoostM1	0,950	0,964	0,957	0,675	0,594	0,632

Delni MSD (ločeno prvi znak MSD, drugi znak MSD in pri samostalnikih še sklon), je malce izboljšal rezultate pri klasifikatorjih ADTree in AdaBoostM1, poslabšal pa pri NaiveBayes in RBFNetwork.

Zanimiv rezultat je prinesla ukinitev atributov z oblikami (torej so ostale le leme), pri čemer je bil rezultat pri ADTree in AdaBoostM1 popolnoma enak, pri NaiveBayes in RBFNetwork pa se je poslabšal.

Delni MSD2 je bil poskus, kako čim bolj prenesti informacije iz MSD in se izogniti uporabi celotnega MSD (zaradi predpostavke, da veliko število različnih MSD lahko ovira učenje). Vendar je tudi ta poskus samo poslabšal rezultate (je sicer izboljšal natančnost pri ADTree, vendar za ceno velikega poslabšanja priklica) (rezultat je poslabšal celo delni MSD 2 in dodani celotni MSD), tako da je očitno najbolj smiselno uporabiti kar celotni MSD.

Atributi s podatki o skladnji so sicer poslabšali rezultat pri klasifikatorjih NaiveBayes in RBFNetwork, vendar so ga popravili pri ADTree in AdaBoostM1, in to toliko, da je F1 pri ADTree postal najboljši, zato je bila za nadaljnje poskuse izbrana ta kombinacija.

4.3 Velikost okna

Preizkušen je bil vpliv velikosti okna, tj. števila besed pred besedo, za katero ugotavljamo, ali ji sledi vejica, in za njo.

Tabela 3: **Šolar, ADTree, MSD + skladnja**

	Ni vejice			Je vejica		
Okno	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
–5+5	0,950	0,983	0,966	0,815	0,588	0,683
–4+5	0,950	0,983	0,966	0,815	0,588	0,683
–3+5	0,950	0,983	0,966	0,815	0,588	0,683
–2+5	0,950	0,983	0,966	0,815	0,588	0,683
–1+5	0,950	0,983	0,966	0,815	0,588	0,683
–0+5	0,950	0,983	0,966	0,815	0,588	0,683
–5+2	0,950	0,983	0,966	0,815	0,588	0,683
–5+1	0,950	0,984	0,966	0,818	0,582	0,680
–5+0	0,889	1,000	0,941	0,000	0,000	0,000
–0+2	0,950	0,983	0,966	0,815	0,588	0,683

Tabela 3 kaže, da klasifikator ADTree uporablja le trenutno besedo in še dve naprej. Vendar razen na hitrost večanje okna ne vpliva negativno na rezultat, zato je pri nadaljnjih preizkusih uporabljeno kar okno –5+5, tudi zaradi domneve, da pri spreminjanju parametrov klasifikatorja ADTree (torej večanjem drevesa) začne klasifikator upoštevati tudi besede zunaj okna –0+2, ki se je pokazalo kot zadostno tukaj (drevo, ki je rezultat poskusa s parametrom –B 50, res vsebuje tudi položaje +3, –1 in –2 in celo –5, torej bi bilo tam optimalno drevo –5+3, kar potrjuje to domnevo). Ta domneva je tudi razlog, da za nadaljnje preizkušanje nismo uporabili okna –5+1, ki je sicer malenkostno izboljšalo natančnost.

Mogoče vpliva na druge klasifikatorje velikost okna drugače, tako da bi bilo smiselno izvesti poskuse še za druge klasifikatorje, prav tako pa tudi za druge parametre klasifikatorja ADTree.

4.4 Vpliv označevanja

Rezultati postavljanja vejic so zelo uspešni, vendar vsebujejo problematično predpostavko: pri oblikoslovnem označevanju in skladenjski razčlembi je bilo uporabljeno besedilo, ki je vsebovalo pravilno postavljene vejice. To pa seveda ni realna situacija, saj v primeru, da hočemo v neko besedilo postaviti vejice, tega vnaprej seveda ne vemo.

Zato je bil naslednji poskus ugotoviti, kaj se zgodi, če oblikoslovni označevalnik in skladenjski razčlenjevalnik nimata vejic v vhodnem besedilu. Iz korpusa so bile izbrisane vse vejice in korpus je bil ponovno označen in pretvorjen v format ARFF. Ker pa je bil seveda povsod podatek, da ni vejice, je bilo treba iz datoteke ARFF za korpus z vejicami prenesti stolpec s podatki za vejico v datoteko ARFF korpusa brez vejic. Pri tem postopku je potrebna previdnost: nujno je treba preveriti, da se ujema število besed in se besede

pokrivajo. Nekateri tipi napak v izvornem korpusu namreč naredijo težave pri brisanju vejic, tak primer je npr. manjkajoč presledek za vejico, pri čemer brisanje vejice potem zlepi besedi in povzroči, da je v korpusu brez vejic ena beseda manj. Težava je tudi, da tokenizator (rezalnik na besede) včasih spreminja vezavo pike na predhodno besedo različno (npr. pri arabskem zapisu vrstilnih številnikov), če je blizu vejica. Te primere je bilo treba v označenem XML potem popraviti ročno, da so se besede ujemale.

Tabela 4: **Šolar, MSD + skladnja, –5+5**

		Ni vejice			Je vejica		
	Klasifikator	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
Označeno z vejicami	NaiveBayes	0,973	0,920	0,946	0,555	0,793	0,653
	RBFNetwork	0,956	0,949	0,953	0,616	0,652	0,634
	ADTree	0,950	0,983	0,966	0,815	0,588	0,683
	AdaBoostM1	0,950	0,964	0,957	0,675	0,594	0,632
Označeno brez vejic	NaiveBayes	0,971	0,916	0,943	0,538	0,783	**0,638**
	RBFNetwork	0,955	0,943	0,949	0,588	0,647	0,616
	ADTree	0,943	0,982	0,962	**0,787**	0,526	0,630
	AdaBoostM1	0,940	0,957	0,948	0,595	0,510	0,550
Označevalnik, naučen brez vejic	NaiveBayes	0,971	0,917	0,943	0,542	0,785	0,641
	RBFNetwork	0,954	0,947	0,951	0,601	0,639	0,619
	ADTree	0,947	0,982	0,964	**0,794**	0,563	**0,659**
	AdaBoostM1	0,947	0,976	0,961	0,745	0,566	0,643
	DecisionTable	0,954	0,989	0,971	**0,873**	0,617	**0,723**

Tabela 4 pove, da so se rezultati ugotavljanja vejic v primeru, ko besedilo pri označevanju ni imelo vejic, poslabšali (čeprav ne zelo izrazito, največja razlika je bila pri klasifikatorju AdaBoostM1), kar se sklada tudi s splošnimi ugotovitvami Hillarda idr. (2006), da pravilno postavljene vejice izboljšajo oblikoslovno označevanje besedil.

Preizkušeno pa je bilo še, ali lahko označevanje (in s tem posledično določanje vejic) izboljšamo s tem, da lematizator, oblikoslovni označevalnik in skladenjski razčlenjevalnik naučimo iz učnega korpusa brez vejic (to sta uporabila že Shieber in Tao (2003)). V ta namen so bile v učnem korpusu SSJ500k izbrisane vse vejice (in povezave na vejice pri skladenjski razčlenitvi) in na novo naučeni modeli za lematizator, oblikoslovni označevalnik in skladenjski razčlenjevalnik (ta postopek predvsem za oblikoslovni označevalnik porabi veliko procesorskega časa (dobrih 20 ur), vendar ga je treba narediti le enkrat). Rezultati so se izboljšali, niso pa dosegli primera, ko je bilo besedilo označeno z vejicami, kar kaže na to, da so vejice pomembne za razdvoumljanje. Vseeno pa se je pokazalo, da je v primeru, ko je treba v besedilu dodati vse vejice, smiselno naučiti označevalnike z učnim korpusom brez vejic.

Tukaj je bil dodatno preizkušen še klasifikator DecisionTable, ki je bil pri izbiranju klasifikatorjev zelo uspešen, vendar ni bil izbran za nadaljnje preizkušanje zaradi dolgotrajnosti preizkušanja.

4.5 Parametri klasifikatorja

Klasifikator DecisionTable je sicer dosegel najboljši rezultat, vendar je posamezni poskus trajal tri dni. Zato je bilo pri drugouvrščenem klasifikatorju ADTree (alternirajoče odločitveno drevo), ki je bil občutno hitrejši, preizkušeno, kako vplivajo nanj parametri.

Tabela 6: **Šolar, ADTree, MSD + skladnja, −5+5, označeno brez vejic**

	Ni vejice			Je vejica		
Parametri	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
-B 10 -E -3	0,943	0,982	0,962	0,787	0,526	0,630
-B 8 -E -3	0,943	0,981	0,962	0,779	0,524	0,627
-B 6 -E -3	0,943	0,981	0,962	0,779	0,524	0,626
-B 4 -E -3	0,939	0,983	0,960	0,779	0,490	0,602
-B 2 -E -3	0,940	0,978	0,958	0,735	0,499	0,549
-B 1 -E -3	0,940	0,948	0,944	0,553	0,515	0,533
-B 12 -E -3	0,944	0,982	0,962	0,785	0,534	0,635
-B 15 -E -3	0,946	0,982	0,964	0,796	0,555	0,654
-B 20 -E -3	0,949	0,984	0,966	0,819	0,578	0,678
-B 30 -E -3	0,949	0,989	0,969	0,868	0,580	0,695
-B 50 -E -3	**0,954**	**0,987**	**0,971**	**0,861**	**0,622**	**0,723**
-B 10 -E -2	0,938	0,986	0,961	0,808	0,480	0,603
-B 30 -E -2	0,945	0,989	0,967	0,865	0,541	0,666
-B 50 -E -2	**0,949**	**0,991**	**0,969**	**0,883**	**0,572**	**0,694**
-B 50 -E -1	0,949	0,991	0,969	0,883	0,572	0,694

Tabela 6 prikazuje spreminjanje rezultatov spreminjanja parametrov. Parameter -B pove število ponovitev dodajanj vozlišč pri gradnji drevesa in tako povečuje drevo, ki je rezultat učenja, hkrati pa podaljšuje čas, ki je potreben za izračun.

Parametri –3, –2 in –1 povedo, na kakšen način išče klasifikator nova potencialna vozlišča. Pri parametru 3 preveri vse možnosti, pri –2 in –1 pa omeji preiskovanje, kar pospeši iskanje, rezultat pa ni nujno optimalen (najboljše možno odločitveno drevo za dano število vozlišč).

Tabela 7: **Šolar, ADTree, MSD + skladnja −5+5, označeno brez vejic, označevalnik, naučen brez vejic**

	Ni vejice			Je vejica		
Parametri	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
-B 10 -E -3	0,947	0,982	0,964	0,794	0,563	0,659
-B 30 -E -3	0,953	0,988	0,970	0,865	0,612	0,717
-B 50 -E -3	**0,956**	**0,987**	**0,971**	**0,861**	**0,641**	**0,735**

Tabela 7 prikazuje rezultate za bolj realen primer, ko je označeno besedilo brez vejic, označevalnik pa je tudi naučen brez vejic. Tudi tukaj večanje drevesa izboljšuje rezultat, seveda pa zato preizkušanje traja dlje. Zadnji rezultat (s 101 listom v odločitvenem drevesu) je najboljši doseženi rezultat, ki je presegel tudi rezultat s privzetimi parametri pri klasifikatorju DecisionTable. V prihodnosti bi bilo smiselno preizkusiti različne parametre tudi pri drugih klasifikatorjih, da bi našli optimalno kombinacijo.

Dodatna prednost klasifikatorja ADTree je, da izpiše odločitveno drevo, ki bi se ga dalo relativno preprosto uporabiti v drugih programih.

```
: -1.039
| (1)je_vez1 = 1: 1.145
| (1)je_vez1 != 1: -0.335
| | (2)msd3 = *: -1.327
| | (2)msd3 != *: 0.092
| (3)lem1 = in: -1.407
| (3)lem1 != in: 0.058
| | (4)je_vez0 = 0: 0.075
| | | (6)lem0 = biti: -1.09
| | | (6)lem0 != biti: 0.087
| | | | (8)zac_modrega0 = 1: -0.526
| | | | (8)zac_modrega0 = 0: 0.092
| | | (9)msd0 = Dm: -2.691
| | | (9)msd0 != Dm: 0.021
| | | | (10)lem1 = kot: -1.264
| | | | (10)lem1 != kot: 0.026
| | (4)je_vez0 = 1: -1.14
| | (5)msd1 = Vd: 0.797
| | (5)msd1 != Vd: -0.102
| | (7)zac_modrega1 = 1: 0.419
| | (7)zac_modrega1 != 1: -0.134
Legend: -ve = ni-vejice, +ve = je-vejica
```

Slika 4: **Odločitveno drevo za ADTree -B 10 -E -3**

Slika 4 prikazuje primer odločitvenega drevesa pri -B 10 (z 21 listi). Na verjetnost, da gre za vejico, najbolj vpliva podatek iz skladenjskega razčlenjevalnika, da na naslednjo besedo kaže povezava »vez«. Zanimiv je vpliv msd3 z vrednostjo * (kar pomeni, da te besede ni), kar z drugimi besedami pomeni, da vejica tik pred koncem stavka ni posebno verjetna. V devetem volišču je zanimiv mds0 Dm, torej predlog, ki zahteva vezavo z mestnikom, ki zmanjša verjetnost, da je neposredno za njim vejica.

5 PRIMERJAVA Z DRUGIMI REZULTATI

Najboljši pridobljeni rezultat je bilo na koncu treba primerjati s prejšnjimi rezultati, najprej z rezultati metod s pravili za slovenščino, potem pa s statističnimi metodami za druge jezike.

5.1 Primerjava z metodami, ki uporabljajo pravila

Oba programa za postavljanje vejic s pravili (Besana in LanguageTool), ki sta bila preizkušena v Holozan (2012), sta bila preizkušena še za primer, ko v besedilu manjkajo vse vejice, s čimer sta bila programa, ki sta sicer namenjena popravljanju napak pri vejicah, prisiljena postaviti vse vejice v besedilo.

Postavilo se je vprašanje, kako obravnavati rezultate Besane. Ta namreč poleg opozoril, kjer točno postavi vejico, opozarja na manjkajočo vejico tudi v primerih, ko sicer ugotovi, da vejica nekje manjka, ne zna je pa točno postaviti. Ti primeri zahtevajo uporabnika, ki zna potem sam postaviti vejico na ustrezno mesto in niso primerni za samodejno postavljanje vejic, npr. pri razpoznavi govora. Zato ima Besana v tabeli dva rezultata, pri prvem so upoštevane le vejice, ki jih Besana točno postavi, pri drugem pa še tiste, za katere le ugotovi, da bi morala vejica nekje biti.

Tabela 8: **Šolar, vse vejice, ADTree (-B 50 -E -3) (označeno brez vejic, označevalnik, naučen brez vejic)**

	Ni vejice			Je vejica		
Klasifikator	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
ADTree	0,956	0,987	0,971	0,861	**0,641**	**0,735**
LanguageTool	0,934	0,991	0,961	0,876	0,509	0,644
Besana	0,953	0,991	0,971	**0,888**	0,572	0,696
Besana + nekje	0,950	0,988	0,969	0,871	0,624	0,727

Tabela 8 kaže, da je statistično postavljanje vejic doseglo najboljši priklic in F1, vendar je natančnost še vedno najvišja pri Besani, čeprav razlika ni velika.

5.2 Primerjava z rezultati za druge jezike

Rezultati samodejnega postavljanja vejic so zelo odvisni od jezika, kar so npr. pokazali Zhang idr. (2002), ki so preizkusili isti metodi na angleščini in nemščini.

Tabela 9: **Šolar, vse vejice, ADTree (-B 50 -E -3) (označeno brez vejic, označevalnik, naučen brez vejic)**

Jezik	Preizkus	Je vejica		
		Natančnost	Priklic	F1
Angleščina	Beeferman idr. (1998), algoritem A	0,756	0,656	0,702
Angleščina	Beeferman idr. (1998), algoritem B	0,784	0,624	0,694
Angleščina	Zhang idr. (2002), Amalgam	0,744	0,676	0,709
Angleščina	Zhang idr. (2002), jezikovno modeliranje	0,782	0,624	0,694
Angleščina	Shieber in Tao (2003)	0,797	0,626	0,748
Angleščina	Israel idr. (2012)	0,858	0,663	0,748
Nemščina	Zhang idr. (2002), Amalgam	0,854	0,875	0,865
Nemščina	Zhang idr. (2002), jezikovno modeliranje	0,896	0,746	0,815
Baskovščina	Alegria idr. (2006)	0,696	0,486	0,572
Slovenščina	Ta članek	0,861	0,641	0,735

Tabela 9 kaže, da je natančnost pri slovenščini podobna kot pri nemščini, priklic pa je slabši. Tudi najboljši rezultat za angleščino (Israel idr., 2012) ima podobno natančnost in priklic slovenskemu rezultatu.

6 ISKANJE REALNIH NAPAK

Dosedanji rezultati povedo, kako dobro postavijo programi vejice v besedilo, v katerem ni na začetku nobenih vejic, kar je npr. uporabno pri razpoznavi govora, ki ne zazna vejic. Vprašanje pa je, kako dobro se programi obnesejo pri popravljanju pravih napak, saj te niso naključno razporejene, ampak določeni tipi vejic delajo piscem več težav kot drugi. Za tak preizkus je treba dobiti korpus napak pri vejicah, kar je bilo mogoče s korpusom Šolar. Vendar pa je primerov napačnih vejic veliko manj kot vseh primerov vejic, pa še štiri možna stanja so (ob *je vejica* in *ni vejice* še *ni manjkajoče vejice* in *je odvečna vejica*) in je zato vprašanje, ali bi bilo 11399 primerov manjkajoče vejice in 2709 primerov odvečne vejice dovolj za uspešno učenje, še večji korpus primerov napak pri vejicah pa bo težko dobiti.

Zato je bil izbran drugačen postopek: program WEKA nastavimo tako, da je prvih (izključimo privzeto naključno izbiranje) 80 odstotkov primerov učni korpus, zadnjih 20 odstotkov pa uporabimo kot testni korpus, pri čemer se rezultat preizkušanja izpiše za vsak primer posebej. Ker so v korpusu Šolar primeri sicer razporejeni po razredih in letnikih oz. vrstah šol, ne bi bilo v redu, če bi vsi preizkusni primeri prišli iz istega letnika oz. šole (Holozan (2012) je pokazal, da so rezultati popravljanja vejic različni glede na letnik oz. šolo), je bil najprej izveden postopek, ki je delno premešal primere tako, da je bila najprej izločena vsaka peta poved, te izločene povedi pa so bile potem dodane na koncu.

Rezultat preizkušanja (stolpec, ki pove, katero stanje vejice je izbral klasifikator) je bil potem poravnan s podatki o vejicah iz korpusa (pri čemer je bilo treba paziti, da se je poravnalo z zadnjimi primeri in ne s prvimi), oboje je bilo sestavljeno v eno tabelo, potem pa prešteto, kolikokrat se je pojavila katera kombinacija.

```
1653-je-vejica      je-vejic
 694-je-vejica      ni-vejic
1453-manjka-vejica  je-vejic
 885-manjka-vejica  ni-vejic
 575-ni-vejice      je-vejic
36037-ni-vejice     ni-vejic
 197-prevec-vejica  je-vejic
 337-prevec-vejica  ni-vejic
```

Slika 5: **Rezultat primerjave rezultatov preizkušanja s podatki iz korpusa**

Slika 5 prikazuje tak (surov) rezultat za primer, ko je bil korpus označen z vsemi vejicami pravilno postavljenimi, spredaj je število primerov, drugi stolpec je stanje v korpusu in tretji stolpec je rezultat preizkušanja klasifikatorja, torej je npr. v 1453 primerih, ko je vejica manjkala, klasifikator menil, da bi tam morala biti vejica, v 885 primerih pa, da tam ni vejice, po drugi strani pa je v 575 primerih postavil vejico, kjer je ne bi smelo biti, natančnost (kakšen delež dodanih vejic je pravilen) je tako 1453 / (1453 + 575) oz. 71,7 odstotka.

Tak postopek je bil ponovljen za različne načine označevanja, ni pa bilo izvedeno desetkratno prečno preverjanje, ker bi bil ta postopek precej zapleten (in bi ga bilo treba prej bolj avtomatizirati, zdaj so bili nekateri koraki izvedeni ročno za vsak primer posebej). Samo 10 odstotkov primerov pri preizkušanju pa bi bilo morda tudi premalo, da bi lahko potem dovolj zanesljivo dobili rezultat pri primerjavi z napakami v korpusu, zato je bila izbrana razdelitev 80 : 20. Preizkušanje je bilo izvedeno le s klasifikatorjem ADTree s parametri (-B 14 -E -3), da ne bi trajalo predolgo.

Tabela 10: **Rezultat iskanja realnih napak, ADTree (-B 14 -E -3)**

	Popravljanje manjkajočih vejic			**Popravljanje odvečnih vejic**		
Način	**Natančnost**	**Priklic**	**F1**	**Natančnost**	**Priklic**	**F1**
Označeno z vsemi vejicami	0,717	0,622	0,666	0,327	0,631	0,431
Označeno brez vejic	0,690	0,482	0,567	0,283	0,642	0,393
Označeno brez vejic, označevalnik brez vejic	**0,676**	**0,545**	**0,603**	**0,298**	**0,633**	**0,406**
Označeno z vejicami v besedilu	0,675	0,491	0,568	0,293	0,564	0,385
Označeno z vejicami v besedilu, označevalnik brez vejic	0,672	0,541	0,600	0,292	0,592	0,391
LanguageTool	0,812	0,442	0,572	/	/	/
Besana	0,862	0,505	0,636	**0,902**	0,094	0,170
Besana + nekje	0,876	0,702	0,779	0,902	0,094	0,170

Tabela 10 prikazuje rezultat iskanja realnih napak in primerjavo z LanguageTool in Besano. Zanimivo je, da je najboljši rezultat dosežen, če pri označevanju na vhodu izbrišemo vse vejice in potem uporabimo označevanje, naučeno brez vejic (če seveda izvzamemo označevanje, pri katerem so vse vejice postavljene pravilno, česar seveda normalno nimamo). Če že postavljene vejice pri označevanju pustimo v besedilu, je rezultat torej slabši, in sicer ne glede na to, ali je označevalnik naučen z vejicami ali brez njih.

Zanimiv je tudi rezultat pri odkrivanju odvečnih vejic, pri čemer statistična metoda sicer doseže veliko boljši priklic (0,633 proti 0,094), vendar hkrati tudi neuporabno nizko natančnost (0,298 proti 0,902) (tukaj bi bilo smiselno preizkusiti še idejo iz Israel idr. (2012), da ne upoštevamo le dejstva, da se je klasifikator odločil, da neke vejice ni, temveč tudi njegovo oceno te odločitve, tako da vejico označi kot odvečno le, če ta ocena preseže določeno mejo). Tudi pri manjkajočih vejicah je težava predvsem natančnost, priklic je boljši od LanguageTool in Besane (razen če pri Besani upoštevamo še opozorila, da nekje manjka vejica).

Opozoriti je treba še, da je gostota napak v teh primerih velika, saj so bile preverjene le povedi, v katerih je bila bodisi kakšna odvečna bodisi manjkajoča vejica. Zato bi bilo treba pripraviti boljši korpus napak, ki bi vključeval tudi pravilne stavke, da bi dobili pravo natančnost. Je pa natančnost zelo odvisna od kakovosti vhodnega besedila, če natančnost preizkušamo na besedilu, ki nima (ali skoraj nima) napak, bo natančnost slabša, kot če je napak veliko.

Za angleščino so Israel idr. (2012) dosegli natančnost 0,849 pri priklicu 0,200 (F1 0,324), vendar je to rezultat za vse napačne vejice, ni pa posameznih rezultatov za manjkajoče oz. odvečne vejice.

7 SKLEP

Poskusi so pokazali, da je postavljanje vejic z uporabo strojnega učenja zelo uporabno v primeru, ko želimo poiskati vse vejice v besedilu. Za najboljši rezultat je treba uporabiti označevanje z označevalniki, ki so bili naučeni z učnimi korpusi z odstranjenimi vejicami, uporabiti je treba skladenjsko razčlenjevanje, kot najbolj uporaben se je pokazal klasifikator ADTree (alternirajoče odločitveno drevo), njegova prednost je tudi preprosto odločitveno drevo, ki bi se dalo hitro sprogramirati tudi v kakšnem programu. Rezultati se izboljšujejo z večanjem drevesa, vendar hkrati narašča potrebni čas za izračun, najuspešnejši poskus je bil izveden z nastavitvami -B 50 -E 3 z oknom –5+5. Rezultat za slovenščino je primerljiv z rezultati za druge jezike, dosežena je bila natančnost 0,861, priklic 0,641 in F1 0,735.

Glede na to, da program WEKA podpira veliko število klasifikatorjev, čisto vsi niso bili preizkušeni, pa tudi pri tistih, ki so bili, je odprtih še veliko možnih poskusov s parametri klasifikatorjev. Problem je tudi čas, ki je potreben za izračunavanje; pri klasifikatorju ADTree se je pokazalo, da večanje drevesa izboljšuje rezultat, vendar zgornja meja ni bila dosežena, ker postane preračunavanje pri tako velikih drevesih prepočasno (najboljši rezultat se je računal skoraj tri dni). Vsekakor je še veliko možnih kombinacij klasifikatorjev, parametrov, različnih atributov, oken, pri katerih bi bilo verjetno mogoče doseči še boljši rezultat.

Odločitveno drevo, ki je rezultat, bi se morda dalo uporabiti za izboljšavo postopkov postavljanja vejic s pravili, označevanje besedila je sicer relativno zahtevna operacija, kar bi lahko povzročilo težave pri praktični uporabi (npr. kot slovnični pregledovalnik v urejevalniku besedil). V ta namen bi bilo zato morda smiselno poskusiti zgraditi odločitveno drevo s pomočjo atributov, ki jih je laže dobiti, morda celo samo iz samih besed.

Uspeh pri iskanju realnih napak je slabši kot pri iskanju vseh vejic. Rezultati s strojnim učenjem imajo sicer dober priklic (0,545), vendar je natančnost (0,676) slabša od Besane in LanguageTool. Še posebno pa je to očitno pri popravljanju odvečnih vejic, česar LanguageTool sploh ne opravlja, Besana pa ima tudi priklic le 0,094, vendar doseže natančnost 0,902, medtem ko je statistično popravljanje doseglo priklic kar 0,633, vendar je natančnost le 0,298. Zanimivo je, da je bil najboljši rezultat dosežen v primeru, ko so bile v besedilu pred označevanjem izbrisane vse vejice (in je bil tudi označevalnik naučen brez vejic); tudi pravilne vejice so označevanje motile, kar je presenetljiv rezultat. Se pa lahko ta rezultat spremeni, če se bo povečal delež pravilnih vejic v preizkusnem korpusu, zdaj so namreč v njem le povedi z napačnimi vejicami, zaradi tega je tudi natančnost nerealno visoka.

Naloga za prihodnost je razširiti dosedanji preizkusni korpus, pridobljen iz korpusa Šolar, še s pravilnimi povedmi iz korpusa Šolar, ki nastopajo ob povedih z napakami, in potem ponoviti ta poskus. Smiselno bi bilo dodati še primere iz drugih virov, ki so dostopni pod licenco Creative Commons (npr. Wikipedije), in označiti napačne vejice in tako zgraditi in objaviti referenčni korpus za učenje/popravljanje vejic, ki bi bil dostopen pod licenco Creative Commons, s čimer bi ga lahko za eksperimente uporabljali tudi drugi, tako da bi bili rezultati bolj primerljivi.

8 VIRI IN LITERATURA

[1] Alegria, I., Arrieta, B., de Ilarraza Sánchez, A. D., Izagirre, E. & Maritxalar, M. (2006). *Using Machine Learning Techniques to Build a Comma Checker for Basque*. V N. Calzolari, C. Cardie & P. Isabelle (ur.), ACL : The Association for Computer Linguistics.

[2] Beeferman D., Berger A. & Lafferty J. (1998). *Cyberpunc: A lightweight punctuation annotation system for speech*. IEEE Conference on Acoustics, Speech and Signal Processing. Seattle, WA, USA.

[3] Hardt, D. (2001). *Comma checking in Danish*. Paper presented at Corpus Linguistics 2001 conference: Lancaster University (UK), 266–271.

[4] Hillard, D., Huang, Z., Ji, H., Grishman, R., Hakkani-Tur, D., Harper, M., Ostendorf, M., Wang, W. (2006). *Impact of Automatic Comma Prediction on Pos/Name Tagging of Speech*. V zborniku IEEE/ACL 2006 Workshop on Spoken Language Technology.

[5] Holozan, P. (2012). *Kako dobro programi popravljajo vejice v slovenščini*. V zborniku Jezikovne tehnologije: Zbornik C 15. mednarodne multikonference Informacijska družba IS 2012, 8. do 12. oktober 2012, Erjavec, T., Žganec Gros, J.; Ljubljana: Institut Jožef Stefan, okt. 2004, str. 101–106.

[6] Huang, J. & Zweig, G. (2002). *Maximum entropy model for punctuation annotation from speech*. V J. H. L. Hansen & B. L. Pellom (ur.), INTERSPEECH : ISCA.

[7] Israel R., Tetreault J. & Chodorow M. (2012). *Correcting Comma Errors in Learner Essays, and Restoring Commas in Newswire Text*. 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; Montreal, Canada, June 3–8, 2012, str. 284–294.

[8] Shieber, S. M. & Tao, X. (2003). *Comma restoration using constituency information*. V Proceedings of the 2003 Human Language Technology Conference and Conference of the North American Chapter of the Association for Computational Linguistics.

[9] Šek Mertük, P. (2011). *Vejica premalo ali preveč pri študentih razrednega pouka*. Revija za elementarno izobraževanje. Letnik 4, št. 1–2. 123–146.

[10] Zhang, Z., Gamon, M., Corston-Oliver, S., Ringger, E. (2002). *Intra-sentence punctuation insertion in natural language generation*. Tehnično poročilo MSR-TR-2002-58. Microsoft Research.

▪

Peter Holozan je razvijalec v podjetju Amebis, d. o. o., Kamnik in raziskovalec v Amebisovem razvojnem centru. Magistriral je na Fakulteti za računalništvo in informatiko Univerze v Ljubljani in je doktorski študent na Filozofski fakulteti Univerze v Ljubljani (slovenistika). Ukvarja se predvsem z jezikovnimi tehnologijami za slovenščino, med drugim s črkovalniki, slovničnim pregledovalnikom, strojnim prevajanjem, oblikoskladenjskim označevanjem, korpusi (Fida, FidaPLUS) in slovarji (ASP32).