Sveučilište u Rijeci
Filozofski fakultet

Krunoslav Šebetić

Računalna provjera pravopisa hrvatskoga jezika Hunspellom na primjeru imenica a-sklonidbe

DIPLOMSKI RAD

Karlovac, siječanj 2018.

Sveučilište u Rijeci
Filozofski fakultet
Odsjek za kroatistiku

Krunoslav Šebetić

MATIČNI BROJ:
0009058862

Računalna provjera pravopisa hrvatskoga jezika Hunspellom na primjeru imenica a-sklonidbe

DIPLOMSKI RAD

Studij
Hrvatski jezik i književnost

Mentor
dr. sc. Kristian Novak

Karlovac, siječanj 2018.

Tablica sadržaja

1. Uvod i opis zadatka

Ako bi ljudsku vrstu trebalo opisati jednom riječju, teško bi koja bila prikladnijom od riječi *razvoj*. Pojava je jezika onaj trenutak u razvoju ljudske vrste koji je omogućio povezivanje u isprva ne nužno veće ali svakako složenije zajednice. Drugi je važan trenutak u ljudskoj povijesti uporaba pisma koje je omogućavalo bilježenje govorenoga jezika, čuvanje informacija za buduće generacije.

Da je pismo – zapisivanje riječi – važan trenutak u razvoju čovječanstva, svjedoči i podjela na prapovijest i povijest. Prvo razdoblje traje od pojave prvih ljudi (prije šesto tisuća godina) pa do tri tisuće petsto godina prije Krista. Granica nije arbitrarna, granicu predstavlja početak upotrebe pisma u povijesnom razvoju čovječanstva.[1] Uzme li se u obzir dugo trajanje prvoga razdoblja naspram kratkoga drugoga, još je značajnije što do nagloga razvoja i stvaranja velikih civilizacija dolazi u drugome razdoblju – razdoblju obilježeno pismom.

U složenijim je ljudskim zajednicama bilo važno znati koliko je hrane ostalo na zalihama, koliko hrane treba prikupiti za nepovoljno razdoblje, koliko se može zaraditi prodajom određenoga proizvoda okolnim plemenima i narodima, ali važno je bilo i znati koliko je pripadnika zajednice sposobno obnašati vojnu službu – bilo je važno *računati*.

Pomagala je za računanju u ovome ili onome obliku bilo oduvijek, a takve su naprave posebno obilježile osamnaesto i devetnaesto stoljeće pojavom mehaničkih strojeva za računanje koji su mogli puno više od zbrajanja i oduzimanja. Prva se elektro-mehanička računala pojavljuju krajem prve polovice dvadesetoga stoljeća, a koje desetljeće kasnije javljaju se i prva elektronička računala koja više nisu bila samo kalkulatori jer od toga vremena računala prestaju biti ograničena konkretnom namjenom i sposobna su rješavati zadaće opisane računalnim programima.[2, 3]

Pojavom se prvih elektroničkih računala javljaju i prvi obrađivači teksta: specijalizirana računala koja su služila zapisivanju i ispisivanju teksta. Takva su računala s vremenom zamijenila mehaničke i elektro-mehaničke obrađivače teksta i isto takve strojeve za pisanje (pisaće strojeve). Kako je pisati teško i mogućnost je pogreške velika što zbog nedostatka znanja što zbog zatipaka, nije trebalo dugo pa da se pomagalu za pisanje doda još jedno pomagalo: računalna provjera pravopisa i gramatike. Zadaća je ovoga rada opisati mogućnosti računalne provjere pravopisa hrvatskoga jezika Hunspellom.

1 http://hol.lzmk.hr/clanak.aspx?id=32130; pristupljeno 8. siječnja 2017.
2 http://enciklopedija.hr/Natuknica.aspx?ID=48431; pristupljeno 9. prosinca 2017.
3 http://enciklopedija.hr/Natuknica.aspx?ID=44608; pristupljeno 9. prosinca 2017.

U prvome se poglavlju rada opisuje što su to programi otvorenoga koda, odnosno koje su prednosti (i nedostaci) takvih programa jer su upravo otvorenost i dostupnost ono što Hunspell čini posebno prigodnim za izradu računalne provjere pravopisa.

U drugom će se poglavlju opisati pojedine značajke Hunspella, odnosno opisat će se što Hunspell može i kako napraviti rječnik za provjeru pravopisa (općenit pregled značajki).

U trećem će se poglavlju na primjerima prikazati ono što je opisano u drugome poglavlju, odnosno opisat će se koje Hunspellove značajke koristiti za provjeravanje pojedinih pravopisno problematičnih situacija u pisanju, ali opisat će se i koje se pravopisno dvojbene situacije ne mogu provjeravati računalnom provjerom pravopisa.

Zaključak je zadnje poglavlje ovoga rada i u njemu će se dati sinteza prethodnih poglavlja kao i upute kako poboljšati i unaprijediti rječnik za hrvatski jezik kojemu se može pristupiti na internetskoj adresi `https://github.com/krunose/hr-hunspell`, a ovaj je diplomski rad zapravo prikaz i skup iskustava koje sam skupljao od 2014. godine održavajući i unaprjeđujući rječnik.

Prvotnu je inačicu objavio Denis Lacković na `http://cvs.linux.hr/spell/` 2003. godine, kako je stajalo u licenciji inačice rječnika od koje se krenulo dalje. Ipak, izgleda da je prva inačica ugledala svjetlo dana još 2002., a posljednja je promjena učinjena 2004. g. Ta je inačica rječnika bila korištena kao zadani način provjere pravopisa u obrađivaču teksta OpenOffice.org. Koristeći taj obrađivač teksta svakodnevno, primijetio sam da neke (pa i frekventne) riječi nedostaju, a nije bila u dovoljnoj mjeri iskorištena ni mogućnost izrade gramatičkih i tvorbenih morfema. Bio je to više manje popis riječi u kojemu su gramatički nastavci bili grupirani prema formalnom obliku: nastavak *-a* koristio se iz iste klase jednako za nominativ jednine ženskoga roda i za genitiv u muškome rodu, što je onemogućavalo korištenje rječnika za provedbu morfološke analize, ali je i održavanje rječnika bilo otežano – teško je bilo dodavati riječi jer se nova riječ nije mogla jednostavno povezati s gramatičkim ili tvorbenim nastavcima jer nisu postojale odgovarajuće klase (deklinacijski i konjugacijski obrasci). Rječnik je sadržavao i dosta zatipke koji su najvjerojatnije rezultat neprovjeravanja unesenih riječi jer je popis, najvjerojatnije, načinjen računalnom skriptom koja je prikupljala riječi iz digitalnih izvora pa su u rječnik uneseni i zatipci iz tih izvora. Prvu sam riječ dodao 2013. g., ali za nekoga tko nije informatičar i tko se inače time ne bavi, proces je učenja bio dugotrajan. Riječi sam, u okviru tadašnje strukture rječnika skupljao do 2014., kada sam zatražio uključivanje izmjena u obrađivač teksta LibreOffice. Promjene su bile prihvaćene, a tako sam i nastavio dodavati i ispravljati riječi, ali ne i mijenjati strukturu rječnika. Takvo je

stanje 2016. zatekao Mirko Kos te predložio prvu veću strukturnu izmjenu. Budući da sam već održavao rječnik, M. Kos je predložio da se njegove izmjene uključe u ono što sam ja do tada radio umjesto da izmjene objavi pod svojim imenom kao drugi rječnik. Rado sam prihvatio prijedlog jer se radilo o važnim i značajnim izmjenam, a rječnik je potom uključen u LibreOffice kao zadani način provjera pravopisa. Isti se rječnik koristi i za računalnu provjeru pravopisa u internetskom pregledniku Mozilli Firefoxu.

Međutim, u tom sam vrijeme već započeo isti posao kao i M. Kos, ali zbog toga što je predložena izmjena bila velika, važna i – dovršena, uključena je rječnik, a započet sam posao zapustio i nikada ga nisam dovršio. Jedan je od razloga i taj što je bolje ispravljati i dorađivati ono što postoji negoli raditi nešto ispočetka kada je jasno da bi i takav rječnik imao nedostataka i pogrešaka, a sada je prednost u tome što su nedostaci – poznati. Korisniku će više koristiti unaprijeđen postojeći rječnik nego zadržavanje postojećega stanja radi izrade novoga rječnika jer to je posao koji traje. U ovome ću radu pokušati prikazati proces izrade rječnika za računalnu provjeru pravopisa hrvatskoga jezika Hunspellom na temelju dosadašnjih izmjena i iskustava.

U radu će se kratko dotaći i toga koji se tip jezičnih pogrešaka u pisanju može riješiti uparivanjem računalne provjere pravopisa s računalnom provjerom gramatike. Računalna je provjera gramatike kompleksnija tema kojoj se može pristupiti iz različitih kutova: s gledišta lingvistike, ali i s gledišta računalne znanosti čime se to ne može obraditi u ovome radu.

U povijesti možda ništa nije tolikim intenzitetom utjecalo na čovječanstvo kao razvoj računala i drugih digitalnih tehnologija. Računala se danas nose u džepovima i koriste u svim sferama ljudskoga života. Internet i superračunala omogućuju obradu vrlo velike količine podataka u relativno kratkome vremenu. Računala su nezamjenjiv alat u inženjerstvu, digitalnoj obradbi slika i digitalnoj animaciji, izradi klimatskih modela i predviđanja, a nezamjenjiva su i u spremanju velike količine podataka. Široku su primjenu našla i u lingvistici kao pomoć pri prevođenju, pomoć pri učenju jezika, pretvaranju teksta u govor, pretvaranja slikovnih zapisa u tekst, pretraživanju i formiranju velikih jezičnih baza i pretraživih korpusa koji omogućuju lingvistima da jezik analiziraju na do sada nemoguć način. Računalna je provjera pravopisa samo mali dio onoga kako računalo može pomoći čovjeku u vezi s jezikom; a znanost i tehnologija idu samo naprijed.

2. Što je to slobodni računalni program

Računalni program je slijed nedvosmislenih instrukcija[4] koje provodi procesor računala u svrhu obavljanja nekoga zadatka.[5] Danas se računalnim programom uglavnom smatra slijed naredbi pisanih nekim od viših programskih jezika u svrhu izvršavanja na elektroničkome (digitalnome) računalu.[6]

Prvi su računalni programi pisani strojnim jezikom koji je teško razumljiv ljudima, ali je ujedno i jedini način izravnoga davanja instrukcija računalu.[7] Računalni se kôd sastoji od niza jedinica i nula koje interpretira procesor računala dohvaćajući naredbe u svoje registre. Rezultat je te interpretacije odluka koju operaciju provesti i gdje smjestiti rezultat operacije.[8] Jedinice i nule predstavljaju jedina dva stanja koja računala razlikuju: jedinica je stanje kada ima struje, a nula je stanje kada nema struje. Procesor interpretira slijedove jedinica i nula vršeći radnju ovisno o rezultatu unosa.[9]

Strojni je jezik teško pisati i čitati što znači da je i pisanje programa tim načinom dugotrajno i skupo. S vremenom su se pojavili jezici koji su dozvoljavali zamjenjivanje ponavljajućih izraza strojnoga kôda jednostavnijim izrazima razumljivijim čovjeku[10] čime se došlo do programskih jezika viših od strojnoga – do simboličkih programskih jezika koji su olakšali i ubrzali proces pisanja programa: tzv. asembler.[11]

Danas se računalni programi ne pišu strojnim jezikom. Razvijeni su viši programski jezici koji se sastoje od leksema (ključne riječi), gramatike i sintakse koja imitira značajke prirodnih jezika s ciljem jednostavnijega i bržega pisanja računalnih programa.

Tako se napisani programi prevode pomoću drugih računalnih programa – kompajlera (engl. *compiler*) – u strojni kôd.[12] Ponekad se prevođenje odvija i iz višega programskoga jezika u asembler, a zatim iz asemblera u strojni kôd. Pretvaranjem izvornoga kôda računalnoga programa u strojni kôd dobiva se izvršna datoteka koju čovjek više ne može čitati (razumjeti). Osim kompajliranih jezika, postoje i interpretirani jezici čije su naredbe pretvaraju u strojni kôd za vrijeme čitanja.

4 https://www.britannica.com/technology/computer-program; pristupljeno 3. studenoga 2017.
5 http://enciklopedija.hr/natuknica.aspx?ID=68626; pristupljeno 3. studenoga 2017.
6 https://www.britannica.com/technology/computer-programming-language; pristupljeno 3. studenoga 2017.
7 https://www.britannica.com/technology/machine-language; pristupljeno 3. studenoga 2017.
8 http://enciklopedija.hr/natuknica.aspx?ID=68626; pristupljeno 3. studenoga 2017.
9 https://www.britannica.com/technology/digital-computer; pristupljeno 3. studenoga 2017.
10 https://www.britannica.com/technology/computer-programming-language; pristupljeno 3. studenoga 2017.
11 http://www.enciklopedija.hr/natuknica.aspx?ID=4149; pristupljeno 3. studenoga 2014.
12 http://www.enciklopedija.hr/natuknica.aspx?ID=68100; pristupljeno 3. studenoga 2017.

Iz prethodnoga proizlazi da računalni program može biti i slijed jednostavnih instrukcija za izvršenje jednostavnoga zadatka i vrlo složeni programski paketi koji obavljaju vrlo složene zadaće. Svi se računalni programi imenom zovu programska podrška (engl. *software*) te ona omogućava smisleno i pouzdano djelovanje računala, komunikaciju računala i korisnika, ali i komunikaciju između više računala i izvođenje korisničkih programa (Galešev i sur.: 44 – 45)

2.1. Naziv i vrlo kratka povijest slobodnoga programa

U razdoblju između 1969. i 1970. tvrtka AT&T Bell Laboratories razvila je vlastiti operacijski sustav imena Unix. Kasnije je operacijski sustav učinjen portabilnijim čime više nije bio vezan isključivo za jednu arhitekturu i sklopovlje (engl. *hardware*). Dostupnost je ovoga operacijskoga sustava potaknula akademsku zajednicu Sveučilišta Berkley da razvije vlastitu inačicu nazvanu BSD (*Berkeley Software Distribution*). Godine 1984. Richard Stellman odlučio je napraviti *slobodnu inačicu* operacijskoga sustava Unix. Od više inačica ovoga operacijskoga sustava, većina ih je bila vlasnička što je sprječavalo R. Stellmana da ih iskoristi za razvoj vlastitoga slobodnoga operacijskoga sustava. Tri su inačice bile otvorenoga koda: FreeBSD, NetBSD te OpenBSD.[13] Program je otvorenoga kôda onaj program koji nije dostupan samo kao izvršni (binarni) kôd, već je dostupan i izvorni kôd što omogućuje uređivanje i mijenjanje takva programa, ako to dopušta licencija pod kojom je program izdan – od strane korisnika ili drugoga razvijatelja.

R. Stellman je do 1990-ih razvio većinu alata potrebnih za rad operacijskoga sustava, ali nedostajala mu je jezgra koja bi posredovala između programa operacijskoga sustava i računalnoga sklopovlja. Godine 1991. Linus Torvalds počeo je razvijati takvu jezgru (engl. *kernel*) koja se mogla koristiti s alatima R. Stellmana, ali i s dijelovima inačica otvorenih inačica sustava BSD što je dovelo do slobodnoga operacijskoga sustava GNU/Linuxa.[14]

Slobodni računalni program je naziv razvojnoga modela računalnih programa, ali i naziv pokreta kojim je Richard Stellman želio omogućiti korisnicima pristup računalnim programima koji poštuju njihove slobode.

Slobodni je program onaj program koji dozvoljava korisniku da izvršava, kopira, distribuira, mijenja i unapređuje program koji koristi. U engleskom jeziku ne postoji razlika

13 https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/history.html; pristupljeno 3. studenoga 2017.

14 https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/history.html; 3. studenoga 2017.

između *slobodan* i *besplatan* jer engleski za obje situacije koristi riječ *free* (engl. *free software*) čime se maskira razlika između slobode u smislu korištenja i besplatnosti. Zbog toga se često o slobodnome računalnome programu razmišlja kao o programu koji nije potrebno platiti, ali *besplatnost* je takva računalna programa samo slučajnost, posljedica slobode korištenja operacijskoga sustava jer je dostupnost sustava preduvjet za slobodno korištenja, a to onda znači da se sami kôd ne prodaje jer bi onaj tko prodaje bio u poziciji moći što je u suprotnosti s idejom slobode. Zadaća projekta nije stvarati besplatne programe, nego slobodne. Internetnom kruži krilatica *slobodan u smislu slobode govora, ne u smislu besplatnoga pića*.[15]

Da bi program bio slobodnim programom mora zadovoljiti četiri temeljne slobode: nulta je sloboda (ili pretpostavka) da korisnik može pokretati program kako želi i u koju god svrhu želi. Prva sloboda traži da je korisniku dozvoljeno proučavati program (principe rada) i korisniku je dozvoljeno mijenjati ga kako bi program činio upravo ono što korisnik treba i želi. Pristup je izvornome kodu preduvjet za ispunjenje ove slobode. Druga sloboda podrazumijeva da korisnik ima pravo slobodno redistribuirati kopije programa. Treća sloboda podrazumijeva pravo korisnika da redistribuira i izmijenjene inačice programa, ne samo originalne inačice. Cilj je omogućiti zajednici korisnika i razvijatelja (programera) da se mogu koristiti poboljšanim inačicama programa. Dostupnost je izvornoga kôda preduvjet za ispunjenje ove slobode.[16]

Da bi program bio slobodan, slobodan mora biti i svaki drugi program koji prvi program pokreće. Ako jedan program zahtijeva pokretanje drugoga radi izvršavanja neke zadaće, i taj drugi program mora biti slobodan. Ako se planira redistribuirati jedan program koji poziva drugi program radi izvršenja nekoga zadatka, a program se može tako napisati ili postaviti da za izvršenje zadatka nije potreban i drugi program, onda je bitno da samo prvi program bude slobodan.[17]

Licencija pod kojom se izdaju slobodni računalni programi zove se *General Public Licence* ili skraćeno *GPL*. Međutim, činjenica da je nešto objavljeno pod GPL-om ne znači automatski da je samo program objavljen pod tom licencijom i slobodni program. Postoje i brojne druge licencije koje pružaju korisniku temeljne slobode na način kako je to zamislio R. Stellman. Da bi se računalni program okvalificirao kao slobodni, potrebno je proučiti licenciju i vidjeti omogućuje li ona korisniku četiri temeljne slobode. Zapravo je slobodni računalni

15 https://www.gnu.org/philosophy/free-sw.html; pristupljeno 3. studenoga 2017.
16 https://www.gnu.org/philosophy/free-sw.html; pristpuljeno 3. studenoga 2017.
17 https://www.gnu.org/philosophy/free-sw.html; pristupljeno 3. studenoga 2017.

program svaki program objavljen pod licencijom koja ne sprječava korisnika da koristi program onako kako je to definirano četirima slobodama. Licencija GPL samo je jedna od licencija kompatibilnih s idejom slobodnih programa.[18]

2.2. Prednosti i nedostaci slobodnih programa – ukratko

Svaki program ima izvorni kôd – dio programa koji piše razvijatelj i koji je razumljiv čovjeku. Izvorni se kôd piše jednostavnijim ili naprednijim uređivačima teksta (engl. *text editor*). Dostupnost je izvornoga koda ono što razlikuje programe otvorenoga kôda od vlasničkih programa, tj. programa zatvorenoga kôda.

Vlasnički su programi oni programi koji su distribuirani kao izvršne (binarne) datoteke i čije licencije (uvjeti korištenja) dozvoljavaju korištenje programa uz, najčešće, plaćanje naknade. Korisnik kupnjom programa za zaštitu računala od zlоćudnih programa dobiva izvršnu datoteku programa s dokumentacijom na optičkom mediju ili u obliku izvršne datoteke preuzete s interneta nakon što je provedeno plaćanje proizvoda. Korisnik nije u mogućnosti mijenjati izvorno kôd takva programa pa makar bio i iskusan razvijatelj računalnih programa.[19]

Programi otvorenoga koda računalni su programi koji su distribuirani i s izvornim kodom programa: korisnik, posjeduje li potrebna znanja i vještine, može mijenjati program. Međutim, pristup izvornome kodu ne znači nužno da je korisniku prema licenciji (uvjetima korištenja) dozvoljeno i mijenjati program ili distribuirati izmijenjene inačice programa. Treba razlikovati mogućnost mijenjanja programa od mogućnosti da se te promjene provedu legalno. Postoje programi pisani programskim jezikom koji ne dozvoljava (ne omogućuje) izradu izvršnih datoteka već se program izvršava izravno iz izvornoga koda tako što interpreter – računalni program koji interpretira izvorni kôd naredbu po naredbu – izvršava program. Takvi programi nužno moraju biti distribuirani zajedno s izvornim kodom, ali licencije takvih programa mogu još uvijek zahtijevati naknadu za korištenje programa ili pak mogu zabranjivati proučavanje, mijenjanje ili distribuiranje izmijenjenih inačica programa.

Bez obzira na navedeno, termini se *otvoreni kod* te *slobodni program* često koriste kao sinonimi i često obuhvaćaju iste programe. Gotovo je svaki slobodni program ujedno i program otvorenoga koda jer je pristup izvornome kodu uvjet za pružanje slobode korisniku

18 https://www.gnu.org/licenses/gpl-faq.html; pristupljeno 5. studenoga 2017.

19 Usp. na https://www.gnu.org/philosophy/categories.html#ProprietarySoftware; pristupljeno 5. studenoga 2017.

da s programom radi što i kako želi. Isto je tako velik broj programa otvorena koda ujedno i slobodni program.[20]

U prethodnome je poglavlju opisano što zapravo jest slobodni program, odnosno koje uvjete program mora zadovoljavati da bi bio slobodnim programom. Prednost je slobodnih programa to što korisniku daje slobodu da koristi, mijenja i distribuira program – izvorne i izmijenjene inačice – kome želi i kako želi. Slobodni program ne ograničava korisnika načinom ni svrhom uporabe i upravo to i jest jedna od glavnih prednosti slobodnih programa.[21] Mogućnost mijenjanja izvornoga koda programa kada to dopušta i licencija omogućuje prilagođavanje konkretnim zahtjevima korisnika, čini program prilagodljivim i dostupnim širem krugu korisnika. Prednost je takvih programa ta što se oko njih često formira zajednica stručnih i talentiranih razvijatelja koji dokumentiraju program, pružaju korisničku podršku i ispravljaju nedostatke programa.

Ipak, nisu svi aspekti ovoga razvojnoga modela pozitivni. U razvoju je vlasničkih programa uključen kapital koji određuje kvalitetu i razvojni intenzitet. Ako je potrebno da program ima određenu značajku, tvrtka koja razvija program može platiti razvijatelja koji će implementirati traženu značajku. Otkrije li se nedostatak u programu, tvrtka može platiti razvijatelju da ukloni nedostatak. Situacija je sa slobodnim programima uglavnom – suprotna.

Slobodni programi često nastaju iz potrebe, želje za dokazivanjem ili težnje da se riješi neki problem a gdje je težnja začinjena entuzijazmom. R. Stellman je želio pružiti svijetu programe koji ne ograničavanju korisnikov način upotrebe računala. L. Torvalds je bio student koji se želio okušati u izradi jezgre koja ima potencijal postati operacijskim sustavom. Spajanjem jezgre i alata i jest nastao slobodni operacijski sustav: GNU/Linux. Međutim, spomenuti dvojac nije to napravio sam: objavili su svoje težnje na internetu i oko tih je ideja nastala zajednica entuzijasta koji su na kraju i ostvarili zacrtani cilj.

Kako slobodni programi ne zarađuju prodajom, često razvijatelji doniraju svoje slobodno vrijeme, a ponekad koji odvojen sat nije dovoljan. Često se događa da u slobodnim programima određeni nedostaci ostaju godinama jer jednostavno jer jednostavno nitko nije u mogućnosti donirati velik broj radnih sati koji nisu plaćeni. U tvrtkama koje razvijaju vlasničke programe rade čitavi timovi dobro plaćenih razvijatelja. Slobodni programi često riješe temeljni problem (stvaranje slobodnoga operacijskoga sustava), ali taj proizvod nastave koristiti tek entuzijasti ili sami razvijatelji programa. Operacijski je sustav GNU/Linux

20 v. na https://www.gnu.org/philosophy/categories.html#OpenSource; pristupljeno 5. studenoga 2017.

21 v. https://www.gnu.org/philosophy/categories.html#FreeSoftware; pristupljeno 5. studenoga 2017.

postigao veliki uspjeh na poslužiteljima. Glavni je razlog taj što se radi o operacijskome sustavu koji je moguće mijenjati, a velike tvrtke mogu platiti razvijatelje koji potrebama tvrtke mogu prilagoditi i operacijski sustav i razviti razviti aplikacije za korištenje na tome sustavu bez ikakvih naknada ili drugih zakonskih ograničenja. Tako je operacijski sustav GNU/Linux korišten na gotovo svim superračunalima.[22]

Prednost je slobode u odnosu na vlasničke aplikacije i programe mogućnost odabira, odnosno tvrtka nije ograničena mogućnostima i volji tvrtke čiju aplikaciju koristi u svome poslovanju. Za taj se fenomen često koristi engl. izraz *Vendor Lock-In.*[23] Često takve tvrtke ne prodaju program, jer se slobodni računalni programi ne prodaju, već takve tvrtke pružaju druge proizvode ili usluge uz pomoć slobodnih programa.

Možda upravo zbog prethodnoga, slobodni računalni programi nikada nisu postali popularni na stolnim računalima, među krajnjim korisnicima. Koliko je slobodni program jak u industrijskome sektoru, toliko je slab u korisničkome jer ne postoji kapital koji bi omogućio razvijatelju da ispravi određeni nedostatak u programu i dalje ga slobodno i besplatno podijeli sa svijetom. Često se događa da se ispravljaju samo manji nedostaci, a oni u dizajnu programa često bivaju zanemarenima jer ne postoje preduvjeti (besplatno vrijeme) za ispravljanje takvih nedostataka. Rijetko je koji program otvorenoga koda izrastao u profesionalni alat u domeni stolnih, korisničkih računala. Gotovo svaki vlasnički korisnički program ima ekvivalent u obliku slobodnoga računalnoga programa. Slobodni računalni programi nisu besplatna alternativa vlasničkim programima, oni su samo konkurentski proizvodi iste namjene nastali drugačijim razvojnim modelom. Ipak, rijetko koji slobodni program prestigne vlasnički u smislu broja korisnika.

Kao što se slobodni razvojni model pokazao korisnim u industriji, isto bi tako trebalo, i moglo, biti u školstvu i znanosti. Slobodni računalni programi ostavljaju učenicima, studentima, profesorima i znanstvenicima odriješene ruke da ga koriste kako žele i za što žele. Program koji može biti pokrenut na jednome operacijskome sustavu, potencijalno može biti pokrenut i na drugome operacijskome sustavu. Potom i na trećemu. Poništavaju se ograničenja i upravo to i jest razlog zbog kojega školstvo i znanost trebaju prigrliti slobodne računalne programe: ako je potreban program za izvršenje nekoga zadatka, onaj tko takav program napiše, može ga objaviti na internetu i omogućiti drugima pristup izvornome kodu

22 http://www.zdnet.com/article/almost-all-the-worlds-fastest-supercomputers-run-linux/; 5. studenoga 2017.
23 http://www.makeuseof.com/tag/software-vendor-lock-avoid/; 9. prosinca 2017.

toga programa. Netko s drugoga kontinenta može iskoristiti taj program za izvršenje drugoga takvoga zadatka, ali isto tako može ispraviti nedostatke programa čineći ga tako još boljim.

Razvojni je model slobodnih računalnih programa jednak onome po kojemu se razvijaju i znanstvene ideje i spoznaje. Rezultati jednoga otkrića bivaju objavljenima, a ti se rezultati preispituju i nadograđuju, dovode do novih otkrića čineći svijet boljim.

3. Hunspell – slobodni program za računalnu provjeru pravopisa

Hunspell je računalni program koji omogućuje računalnu provjeru pravopisa. Zadani je način provjere pravopisa u LibreOfficeu (slobodnome uredskome paketu), Mozilli Firefox (slobodni internetski preglednik), Mozilli Thunderbirdu (slobodnome klijentu elektroničke pošte), ali je i zadani način provjere pravopisa i u nekim vlasničkim programima kao što su Google Chrome (internetski preglednik), Appleovom operacijskom sustavu OS X, inDesign (profesionalni alat za pripremu dokumenata za tisak) i drugi.[24]

Već se iz popisa programa koji koriste Hunspell vide prednosti koje pružaju slobodni programi – dostupnost omogućava široku primjenu. Tradicionalno se provjera pravopisa koristi samo unutar obrađivača ili uređivača teksta, a ovdje se iz (nepotpunoga) popisa programa koji koriste Hunspell vidi kako je računalna provjera pravopisa ovim programom dostupna i izvan obrađivača teksta. Naime, računalna provjera pravopisa (i gramatike) dostupna unutar Microsoftova Worda, dostupna je samo unutar Microsoftova Worda. Vlasnička narav takva uredskoga paketa i vlasnička narav programa (ili dijela programa) koji provodi provjeru pravopisa (i koji sadržava pravila za hrvatski jezik), dostupna su samo unutar Microsoftova Worda, ili pak samo unutar Microsoftovih programa na Microsoftovome operacijskome sustavu poželi li ta tvrtka učiniti tu značajku dostupnom u drugim svojim proizvodima. S druge pak strane operacijski sustavu GNU/Linux i Appleov OS X, imaju mogućnost računalne provjere pravopisa na bazi cijeloga operacijskoga sustava, ne samo unutar obrađivača teksta. Provjera pravopisa nije ograničena na samo jedan paket ili alat, provjera je pravopisa dostupna na (gotovo) svim aplikacijama unutar tih operacijskih sustava. To je moguće zbog toga što Hunspellova licencija dopušta slobodno korištenje, a pravila za pojedine jezika najčešće su isto objavljena pod slobodnom licencijom. Potrebno je napraviti pravila za računalnu provjeru pravopisa hrvatskoga jezika samo jednom, i ona se mogu koristiti uz Hunspell gdje god je dostupan i taj slobodni program; tamo gdje Hunspell nije dostupan, postoji mogućnost da ga se prilagodi, posjeduje li se dovoljno znanja i vještina, konkretnoj aplikaciji ili operacijskome sustavu.

Izvorni se kôd Hunspella može naći na adresi `https://github.com/hunspell/`, ali datoteke s te adrese nisu spremne za izravno korištenje već ih treba *kompajlirati* u izvršne datoteke. Izvršne su datoteke dostupne u repozitorijima (gotovo) svake distribucije

24 http://hunspell.github.io/; pristupljeno 5. studenoga 2017.

GNU/Linuxa. Hunspell često nije potrebno instalirati na ovaj način, kao zaseban program, jer često računalna provjera pravopisa dolazi uključena s aplikacijama.

3.1. Upotreba Hunspella na operacijskome sustavu GNU/Linux

U ovome se radu ne može ulaziti u opis instalacije programa na operacijskome sustavu GNU/Linux jer postoji nekoliko načina i takvi bi (zapravo) tutorijali oduzimali previše mjesta i vremena pa će se pretpostaviti da se posjeduje računalo s operacijskim sustavom GNU/Linux na kojemu je već instaliran Hunspell, a velika je vjerojatnost da je Hunspell već predinstaliran na takvo računalo. Isto tako, ovaj rad ne obuhvaća detaljan opis tehničkih pojedinosti Hunspella, opisuje se samo najnužnije radi razumijevanja onoga što slijedi u sljedećim poglavljima rada. Zadaća je ovoga rada opisati kako napisati pravila za računalnu provjeru pravopisa hrvatskoga jezika Hunspellom, ne ulaziti u moguće načine provedbe računalne provjere pravopisa.

Hunspell će automatski provjeravati pravopis iz aplikacija za koje se to očekuje, recimo obrađivač teksta, ali Hunspell je moguće pokrenuti i kroz naredbeni redak. Recimo da na računalu postoji datoteka `provjera_pravopisa.txt` i potrebno je provjeriti pravopis u sadržaju te datoteke, u naredbeni će se redak unijeti

```
hunspell -d hr_HR provjera_pravopisa.txt
```

Primjer 1: Pokretanje Hunspella iz naredbenoga retka

što znači sljedeće:[25]

- `hunspell` – poziva program, pokreće ga
- `-d` je argument (stoji za engl. `dictionary`) koji omogućava učitavanje pravila
- `hr_HR` je oznaka pravila koje treba učitati (omogućeno argumentom `-d`)
- `provjera_pravopisa.txt` je ime datoteke koju se provjerava

Uz pretpostavku da je u datoteci pisalo *Raćunalna provjera pravopisa.* Prethodna će naredba vratiti rezultat koji izgleda ovako:

25 Za više v. https://linux.die.net/man/1/hunspell; pristupljeno 5. studenoga 2017.

```
Raćunalna                File: provjera_pravopisa.txt

Raćunalna provjera pravopisa.

0: Računalna
```

Primjer 2: Izgled provjere pravopisa Hunspellom kao samostalnom aplikacijom iz naredbenoga retka, recimo izvan obrađivača teksta

Riječ koju se trenutno provjerava, označena je, a ispod se te riječi nalaze prijedlozi za ispravak. U ovom slučaju rječnik za provjeru pravopisa nalazi samo jedan mogući ispravak. Pritiskom na tipku `0` (nula), Hunspell će pogrešno napisanu riječ u datoteci zamijeniti ispravno napisanom, predloženom riječi. Postoji li više mogućih zamjena, u naredbenom će retku mogući ispravci biti poredani brojevima od nula nadalje i na korisniku je da odabere kojim prijedlogom želi zamijeniti riječ koja je (potencijalno) pogrešno napisana.[26]

Naravno, provjera se pravopisa unutar obrađivača teksta vrši onako kako bi korisnik i očekivao: nepravilno napisane riječi bivaju podcrtane crvenom valovitom linijom, a klikom na takvu riječ pojavljuje se padajući izbornik iz kojega korisnik može odabrati kojim prijedlogom želi zamijeniti pogrešno napisanu riječ, želi li ignorirati pogrešno napisanu riječ te, ako je riječ zapravo dobro napisana, želi li ju možda dodati u prilagođeni rječnik.

Prilagođeni je rječnik teksta datoteka u koju se mogu spremiti one riječi (jedna u retku) koje se ne nalaze u rječniku za provjeru pravopisa, ali ih korisnik često koristi. Najčešće su to osobna imena, uskostručni termini i sl. Ponekad bi takve riječi bilo poželjno uključiti i u matični rječnik, ali o svakoj se riječi treba posebno razmisliti. Najbolje je obavijestiti razvijatelja pravila za pojedini jezik i ostaviti na njemu da odluči hoće li riječi iz prilagođenoga korisničkoga rječnika uključiti u matični rječnik.

3.2. Hunspellovi rječnici za provjeru pravopisa

Hunspell kao projekt ne nudi i rječnike za provjeru pravopisa, što se može činiti čudnim. Hunspell je samo mehanizam koji razumije određena pravila i koja zatim koristi za

26 Detaljnije se o upotrebi Hunspella može pročitati na https://linux.die.net/man/1/hunspell; pristupljeno 5. studenoga 2017.

računalnu provjeru pravopisa. Pravila su sadržana u tekstnim datotekama koje se dodaju drugim programima (obrađivači teksta) kao dodaci.

Zanimljivo je i to što ne postoji središnje mjesto na kojemu se nalaze svi rječnici koje je moguće koristiti uz Hunspell. Rječnici su rasipani po internetu i nije rijetkost da jedan jezik ima i nekoliko inačica rječnika, pa čak različite aplikacije i koriste različite rječnike. Rječnik za engleski jezik, recimo, ne traži se na istome mjestu gdje i izvorni kôd Hunspella, pravila su (rječnici) odvojeni projekt koje je potrebno posebno tražiti na internetu. Husnpell samo pruža infrastrukturu.

Ipak, postoje veći projekti kojima je računalna provjera pravopisa bitna značajka. Takav je i slobodni obrađivač teksta LibreOffice. Iako rječnici (pravila za provjeru pravopisa) nisu službeno vezani ni uz LibreOffice, svakom je razvijatelju rječnika za provjeru pravopisa u interesu da se rječnik uključi u izvorni kôd LibreOffica iz jednostavnoga razloga: LibreOffice je jedina ozbiljna alternativa vlasničkom obrađivaču teksta, Microsoftovu Wordu. S druge pak strane, LibreOffice za provjeru pravopisa koristi samo Hunspellove rječnike. Ta aplikacija može koristiti i druge rječnika, ali onda se radi ili o provjeri gramatike ili o provjeri pravopisa u sklopu naprednijih aplikacija za provjeru gramatike koje pružaju i mogućnost provjere pravopisa. Iz svega proizlazi da se u izvornome kodu mogu naći rječnici za (gotovo) sve jezika za koji posjeduju pravila za računalnu provjeru pravopisa Hunspellom. LibreOffice koristi i druga jezična pomagala poput automatskoga ispravljanja, automatskoga rastavljanja riječi na slogove, rječnika sinonima, provjere gramatike, kategoriziranje jezika, ali ta pomagala izlaze iz okvira teme ovoga rada.

Zbir je svih, ili bar velike većine, rječnika moguće pronaći na adresi `https://github.com/LibreOffice/dictionaries`, a adresa kontakta s razvijateljima dostupna je na adresi

`https://wiki.documentfoundation.org/Development/Dictionaries`.

Međutim, ne treba zaboraviti kako su to samo rječnici povezani s projektom LibreOffice i vjerojatno nisu ni svi rječnici ni sve inačice rječnika, samo one koje koristi LibreOffice. Detaljnije će se o hrvatskome rječniku za računalnu provjeru pravopisa u kojemu od sljedećih poglavlja, prvo treba opisati pravila općenito, odnosno opisati koje značajke Hunspell podržava.

3.3. Pravila za izradu rječnika

Za računalnu je provjeru pravopisa potrebno dvoje: instaliran Hunspell i rječnik koji sadrži popis pravila. Rječnik je potrebno dodati u datoteku na računalu u kojoj Hunspell po zadanim postavkama traži rječnike. Na distribuciji Debian operacijskoga sustava GNU/Linux, rječnik je potrebno dodati u datoteku `/usr/share/hunspell` za korištenje iz naredbenoga retka kako je prikazano ranije. Za korištenje istoga rječnika uz, recimo, LibreOffice, rječnik je potrebno kopirati i na mjesto na kojemu će LibreOffice tražiti rječnik. U slučaju ručnoga instaliranja LibreOfficea inačice 5.3 na spomenutom operacijskom sustavu, lokacija će biti `/opt/libreoffice5.3/share/extensions/dict-hr/`. Sâm se rječnik sastoji od dvije datoteke koje se, konvencionalno, imenuju prema jeziku i mjestu (državi) u kojoj se jezik koristi zbog potencijalnih razlika u pravilima i riječima. Tako rječnik za engleski ima nekoliko inačica: `en_US`, `en_GB`, `en_AU`, `en_CA`, `en_ZA` što su oznake rječnika engleskoga jezika za Sjedinjene Američke Države, Ujedinjeno Kraljevstvo, Australiju, Južnu Afriku. Hrvatski je rječnik imenovan `hr_HR` označavajući tako da se radi o hrvatskome jeziku i označavajući mjesto govorenja: Hrvatska.

Rječnik se zapravo sastoji od dvije tekstne datoteke. Jedna ima datotečni nastavak *dic*, a druga *aff*. Prvi je skraćenica od engl. *dictionary*, a drugi od engleske riječi *affix*, odnosno hrvatski *afiks*.

Datoteke sadrže upravo ono što im i imena govore (*dic*, *aff*): datoteka `hr_HR.dic` sadrži popis riječi, jedna riječ u retku, a datoteka `hr_HR.aff` sadrži tvorbene i gramatičke nastavke, uz još ponešto, koji omogućuju mijenjanje riječi u najčešće, kanonskome obliku u datoteci `hr_HR.dic`.

Pojedine će se značajke Husnpella opisivati kroz pravila za hrvatski jezik, odnosno neće se davati kompletan pregled svih Hunspellovih značajki nego samo onoliko koliko je potrebno da se opiše postupak izrade pravila za hrvatski jezik.

3.3.1. Datoteka s popisom riječi

Podržana pravila (značajke) sadržani su u dokumentu `Hunspell4.pdf` koji je dostupan na
`https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/`.

Sadržaj je i struktura datoteke s popisom riječi jednostavna u odnosu na datoteku s popisom gramatičkih i tvorbenih nastavaka. Uopćeno, datoteka se s popisom riječi može prikazati na sljedeći način[27]

```
4
abeceda/TZ
ako
čovjek/VOUA
život/VO
```

Primjer 3: Osnovna struktura rječničke datoteke (hr_HR.dic)

Broj četiri označava ukupan broj riječi koji se nalazi u datoteci. Svaka riječ zauzima prostor i računalo je mora učitati u memoriju kako bi je moglo koristiti u provjeravanju pravopisa. Budući da se na priloženome popisu riječi nalaze četiri riječi, broj u prvome retku pomaže računalu rezervirati dovoljno memorije.

Nakon oznake ukupnoga broja riječi, slijedi popis riječi. Svaka riječ mora biti u svome retku. Od četiri riječi, tri imaju dodatke: `TZ`, `VOUA`, `VO`. Kosa predstavlja razdjelnik između riječi i nastavka, a nastavak je zapravo oznaka skupine gramatičkih i tvorbenih morfema iz datoteke s gramatičkim i tvorbenim nastavcima. Oznake se, u ovom konkretnom primjeru, sastoje od dva slova, znači oznaka `TZ` predstavlja oznaku, adresu, ključ, po kojemu će Hunspell znati, pronašavši istu takvu oznaku u datoteci s afiksima, koje nastavke, i kako, dodati na riječ `abeceda` da bi se dobili svi padeži ove riječi: *abecede*, *abecedu*...

Kakav će biti tip oznake, u ovom slučaju dvoslovne, ovisi o definiciji u datoteci s gramatičkim i tvorbenim nastavcima. Ove oznake mogu biti i jednoslovne, ali i brojčane. Postoji i treća mogućnost, a to je da dvije oznake – `VOUA` – budu zamijenjene jednom brojčanom oznakom što poboljšava izvođenje programa. Bez obzira što je ukupan broj nastavaka i dalje jednak (i klasa `VO` i klasa `UA`), računalo sve te nastavke drži pod jednom, brojčanom oznakom[28].

27 https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/; pristupljeno 9. studenoga 2017.

28 v. str. 2 datoteke Hunspell4 na https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/; pristupljeno 9. studenoga 2017.

U navedenome je primjeru riječ `ako` prilog što znači da je nepromjenjiva riječ pa stoga nema ni znaku klase jer se tu nema što mijenjati po nastavcima. Međutim, ono što se još može dodati u ovu datoteku jest upravo oznaka vrste riječi.[29]

3.3.2. Datoteka s gramatičkim i tvorbenim nastavcima

Datoteka je s gramatičkim i tvorbenim nastavcima složenija od datoteke s popisom riječi. Strukturu je datoteke moguće podijeliti na tri veća dijela: u prvome se sadrži informacije o enkodiranju teksta datoteke, ali i informacije o zamjenama jednoga slova drugim. Enkodiranje je važno jer je to, da se ne ulazi ovdje u detalje, način na koji računalo povezuje znakove ispisane na zaslonu s mjestima na tipkovnici, odnosno prema enkodiranju računalo raspoznaje i povezuje signal koji šalje tipkovnica s točno određenim znakom. Isto mjesto na tipkovnici može biti povezano s različitim brojem znakova. Engleska tipkovnica ima drugačiji raspored tipaka od hrvatske pa je zaključiti da povezivanje slova s brojčanim oznakama ovisi o jeziku i setu znakova koji taj jezik koristi i o još koječemu. Kako bi rječnik bio valjan, oznaka enkodiranja u datoteci mora odgovarati enkodiranju kojom je sama datoteka i pisana, što se postavlja na razini operacijskoga sustava i uređivača teksta (engl. *text editor*). Primjer povezivanja slova s brojčanim oznakama (enkodiranje) moguće je vidjeti na `http://www.utf8-chartable.de/` (pristupljeno 9. studenoga 2017.). Odabralo se ići s UTF-8 jer se radi o enkodiranju koje podržava velik broj znakova, a praktično je standard u razmjeni dokumenata na internetu i izradi internetskih stranica što povećava kompatibilnost s drugim operacijskim sustavima, jezicima i uređivačima teksta.

Drugi dio datoteke sadržava popis ostalih pravila važnih za rječnik, a jedno od njih ono koje definira kako će se provjeravati polusloženice: kao jedna riječ ili kao dvije, odnosno hoće li se *spomen-ploča* provjeravati kao *spomen-ploča* ili će rječnik posebno provjeravati *spomen*, a posebno *ploča*. Mogućnosti su u ovom dijelu velike i ovdje ih nije moguće opisati sve, tim više što bi to značilo prepisivanje iz službene dokumentacije a za čime nema potrebe. U ovom će se radu objašnjavati pojedine značajke ovoga dijela rječnika s obzirom na hrvatski jezik, odnosno s obzirom na izradu rječnika za računalnu provjeru pravopisa hrvatskoga jezika. Opis pojedinih značajki može se pronaći u dokumentu `Hunspell4` koji je dostupan na

https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/

29 v. str. 9. i 10. datoteke Hunspell4 na https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/; pristupljeno 9. studenoga 2017.

ali postoji i još jedan izvor: primjeri, koji su vrlo kratko opisani u dokumentu `Hunspell4`, u izvornome su kodu Hunspella prikazani na primjerima, odnosno prikazano je koje rezultate daje određeno pravilo u rječniku, ali i koje rezultate rječnik neće prihvatiti. Primjeri su dostupni na `https://github.com/hunspell/hunspell/tree/master/tests/v1cmdline`.

Na navedenoj su adresi primjeri razvrstani tako da prikazuju što za koje pravilo mora biti upisano u rječnik, odnosno što za određenu značajku mora biti upisano u datoteku s gramatičkim i tvorbenim nastavcima, a što u datoteku s popisom riječi. Daje se i točan, poželjan rezultat, odnosno daje se popis rezultata koje će računalna provjera pravopisa prihvatiti, ali isto tako postoji i popis rezultata koje računalna provjera pravopisa neće prihvatiti i predložit će zamjenu. Navedeno se može dobro prikazati na pravilu `compoundflag`. Više će se o pojedinim značajkama Hunspella govoriti u sljedećemu poglavlju gdje se opisuje korištenje Hunspella za izradu pravila za hrvatski jezik, a i onda samo ona pravila koja će pomoću i izradi rječnika onako kako ću predlagati u tome poglavlju. To ne znači da je to jedini način i da su pravila koja će tamo biti opisana jedino što bi trebalo iskoristiti. Zbog navedenoga, ovdje nije potrebno objašnjavati što konkretno pravilo radi nego je bitno opisati strukturu danoga internetskoga izvora:

- `compoundaffix.dic` – sadržaj rječničke datoteke za konkretno pravilo
- `compounaffix.aff` – sadržaj datoteke s gramatičkim i tvorbenim pravilima za konkretno pravilo
- `compundaffix.good` – datoteka s rezultatima koje će računalna provjera pravopisa prihvaćati točnima
- `compoundaffix.wrong` – datoteka s rezultatima koje računalna provjera pravopisa neće prihvaćati točnima

Rastavljanje ove datoteke na tri dijela logička je i više znači čovjeku nego računalu. Sama je datoteka popis pravila koje računalu učita u memoriju i to je sve, ali podjela na tri dijela olakšava opisivanje datoteke naglašavajući ulogu svakoga od dijelova. Ovaj drugi dio ne definira gramatičke ili tvorbene morfeme za riječi nego definira pravila koja koja služe kao pomoć pri navezivanju nastavaka na osnove riječi.

Treći dio datoteke s gramatičkim i tvorbenim nastavcima i sadržava pravila za dodavanje tvorbenih i gramatičkih nastavaka na riječi iz rječnika. Pravila mogu mijenjati dva sufiksa ili prefiksa te još i prefiks ili sufiks. Sve u svemu, maksimalno je dozvoljeno koristiti tri afiksa. Ova se značajka može iskoristiti za dodavanje gramatičkih nastavaka na tvorbene koji se onda mogu dodavati na osnove riječi, ako se odluči tvorbu riječi rješavati na ovaj

način. U primjeru se radi o deklinacijskome obrascu i-tipa imenica ženskoga roda (imenice tipa *riječ*). Pravilo za dodavanje sufiksa izgleda ovako:

```
SFX AA Y 15
SFX AA 0 0 .
SFX AA 0 i .
SFX AA 0 i .
SFX AA 0 0 .
SFX AA 0 i .
SFX AA 0 i .
SFX AA 0 ju .
SFX AA 0 i .
SFX AA 0 i .
SFX AA 0 i .
SFX AA 0 ima .
SFX AA 0 i .
SFX AA 0 i .
SFX AA 0 ima .
SFX AA 0 ima .
```

Primjer 4: Prikaz datoteke s gramatičkim i tvorbenim morfemima

Oznaka `SFX` označava da je riječ o sufiksu. Želi li se mijenjati prefiks, potrebno je pisati `PFX`. Oznaka `AA` imenuje klasu.

Slovo `Y` znači da se ova klasa može koristiti istovremeno kada i druge klase, odnosno ovi se sufiksi mogu koristiti i, potencijalno, s prefiksom. Naravno, teško da se može naći prefiks koji bi išao uz *riječ*, ali to nije važno za opisivanje strukture pravila.

Broj petnaest stoji za ukupan broj pravila unutar klase. Što se tiče oznaka u redovima od drugoga nadalje: prvi znak (nula) predstavlja znak (slovo) koje treba oduzeti od riječi upisane u rječničku datoteku te dodati znak drugi znak – nula znači ništa – a zadnji znak označava uvjet. Ako je napisana točka, to znači da uvjeta nema, da se navedeno pravilo izvodi bez obzira na koji znak završava riječ.

Tako u datoteci `hr_HR.dic` treba pretpostaviti unos

```
1
riječ/AA
```

Primjer 5: Prikaz dodavanja klasa na riječi u rječničku datoteku (hr_HR.dic)

a u datoteci `hr_HR.aff` barem (minimalni primjer)

```
SET UTF-8
TRY ABCDEFGHIJKLMNOPQRSTUVWXYćČčČđĐžŽabcdefghijklmnopqrstuvwxy
FLAG long
SFX AA Y 4
SFX AA 0 0 .
SFX AA 0 i .
SFX AA 0 ju .
SFX AA 0 ima .
```

Primjer 6: Prikaz izrade klase u datoteci s gramatičkim i tvorbenim nastavcima

Dobro je to što se deklinacijski obrazac za *riječ* svodi na promjenu svega nekoliko nastavaka: nulti, nastavak *i* te *ima* u množini. Iz toga će Hunspell prihvaćati riječi *riječ*, *riječi*, *riječju*, *riječima*, ostale inačice neće biti prihvaćene. Primjer gotove datoteke s gramatičkim i tvorbenim nastavcima moguće je vidjeti na adresi

`https://github.com/krunose/hr-hunspell`

Naravno, datoteka na spomenutoj adresi sadrži nešto više informacija, a potrebno je doraditi i pravila za dodavanje afikasa, ali o tome više u sljedećem većem poglavlju.

3.4. Morfološka analiza Hunspellom

Hunspell posjeduje još jednu zanimljivu značajku – morfološku analizu teksta. Riječima je moguće dodijeliti oznaku vrste riječi, ali i oznaku drugih morfoloških kategorija kao što su rod, broj, padež za odgovarajuće imenske riječi te vrijeme, lice i ostalo za glagole. Čak je moguće i definirati tvorbene nastavke od kojih je pojedina riječ dobivena zajedno s osnovnom riječi. Dokument `Hunspell4`[30] (str. 10) navodi sljedeće mogućnosti

30 https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/; pristupljeno 14. studenoga 2017.

- `po:` (engl. *part of speech*) vrsta riječi
- `al:` oznaka za alomorf, dokumentacija daje primjer za englesku riječ *sing* gdje se oznakom `al` označavaju *sang* i *sung*
- `ds:` derivacijski sufiks. Ako je u rječniku riječ *pomoćnica*, oznaka `ds: nica` može zabilježiti kako se radi o dodavanju nastavka *nica* na osnovu označenu `st:` *pomoć*

Ovo nisu sve oznake, opis se svih može pronaći u dokumentaciji, ali nema potrebe da se ovdje prepisuje iz drugih izvora, bitnije je opisati kako mislim da bi ovu značajku trebalo koristiti prilikom izrade rječnika za računalnu provjeru pravopisa hrvatskoga jezika.

Ako se oznaka `po:` uključi u rječničku datoteku, prilikom svake analize teksta, rječnik će uvijek ispisivati tu oznaku. Značajka se morfološke analize može koristiti samo iz naredbenoga retka. Druge aplikacije, poput obrađivača teksta (LibreOffice), same ne nude tu značajku. Odnosno aplikacija koja koristi Hunspell mora podržavati one značajke koje želi pružiti korisniku. Za sada LibreOffice samo provodi provjeru pravopisa, ne nudi i morfološku analizu tijekom obrade teksta, ali to i ima smisla. Dakle, da bi se provela morfološka analiza teksta potrebno je imati rječnik koji nosi spomenute oznake[31] i potrebno je pokrenuti Hunspell iz naredbenoga retka na sljedeći način

```
hunspell -d hr_HR -m a.txt
```

Primjer 7: Pokretanje Hunspella sa značajkom morfološke analize teksta

gdje je razlika u odnosu na uobičajeno pozivanje Hunspella iz naredbenoga retka u argumentu `-m` koji pokreće analizu.

Naravno, računalni program ne može u homonimičnim situacijama znati o kojemu se, recimo, padežu točno radi pa nabraja sve potencijalne padeže prema formalnom obliku riječi. Za primjer bi *riječ* to značilo da će za oblik *riječi*, Hunspell navesti genitiv, dativ, vokativ, lokativ i instrumental jednine te nominativ, genitiv, akuzativ i vokativ množine. Upravo zbog toga nema potrebe homonimične padežne oblike pisati posebno jer se od petnaest oblika iz Primjera 4 sa 22. str. jednostavnim sažimanjem mogu dobiti svega četiri oblika kao u Primjeru 6 sa 23. stranice. Razlika je samo u tome što primjeru treba dodati još oznake vrste riječi i

31 Sadašnji rječnik na https://github.com/krunose/hr-hunspell (pristupljeno 14. studenoga 2017.) ne podržava morfološku analizu teksta jer riječi nisu označene.

gramatičkih padeža sažetih iz upravo spomenutih razloga. Tako rječnička datoteka mora izgledati kao na sljedećem primjeru

```
1
riječ/AA    po: imenica
```

Primjer 8: Rječnička datoteka s dodanom oznakom vrste riječi

a datoteku s gramatičkim i tvorbenim nastavcima treba urediti na sljedeći način:

```
SET UTF-8
TRY ABCDEFGHIJKLMNOPQRSTUVWXYćĆčČđĐžŽabcdefghijklmnopqrstuvwxy
FLAG long
SFX AA Y 4
SFX AA 0 0 .      Njdžr
SFX AA 0 i .      GDVLjdNGAVmnžr
SFX AA 0 ju .     Ijdžr
SFX AA 0 ima .    DLIjdžr
```

Primjer 9: Prijedlog uređivanja datoteke gramatičkih i tvorbenih nastavaka

Primjer pokazuje minimalni unos u datoteci s afiksima. U prvom je retku pravilo o enkodiranju teksta u datoteci, drugi redak sadrži znakove koje će Hunspell koristiti prilikom stvaranja kombinacija s obzirom na riječi koje se nalaze u rječniku: za pogrešku *kuča*, Hunspell će koristiti znakove iza `TRY` kako bi pronašao najvjerojatnije rješenje; tako će, kada dođe na red slovo ć računalni program ustanoviti da se zamjenom *č* sa *ć* dobiva riječ *kuća* koja se nalazu u rječniku što onda riječ *kuća* čini najvjerojatnijom zamjenom za *kuča*.

Međutim, ono što je najvažnije ovdje jest prikaz sažimanja unosa u klasi `AA`, nema potrebe imati petnaest unosa u toj klasi (primjer 4) kada se nastavci ponavljaju, a računalo ne razumije kontekst pa ne može znati je li riječ *riječi* genitiv, dativ, vokativ ili lokativ jednine, odnosno radi li se o nominativu, genitivu, akuzativu ili vokativu množine. Iz toga je razloga bolje pisati ovako sažeto jer manje unosa u rječniku znači brže izvršavanje računalne provjere pravopisa, potrebno je manje resursa za izvršavanje zadatka. Uostalom, način na koji će se napisati pravilo ne utječe na način izvršavanja računalne provjere pravopisa, razlika će samo

biti prilikom provedbe morfološke analize gdje će umjesto `riječi +Gjdžr`; `riječi +Djdžr`; `riječi +Ljdžr` ... biti prikazano `riječi GDVLjdNGAVmnžr` što je ustvari isto, a ovo je drugo rješenje optimalnije i efikasnije, skraćuje se vrijeme izvedbe provjere.

4. Primjer pravila na jednini imenica a-sklonidbe

Do sada je u radu bilo govora o tome što je to računalni program, što je računalni program otvorenoga koda, ukratko je prikazano kako pisati pravila za računalnu provjeru pravopisa Hunspellom, ali još nije bilo govora o tome što računalna provjera pravopisa jest.

Izraz je *računalna provjera pravopisa* prijevod engleskoga izraza *spell checker*. Zadaća je računalne provjere pravopisa provjeravati je li koja riječ ispravno napisana, što zapravo i nema veze s provjerom pravopisa. Pravopis je – time i provjera pravopisa – jedno, a provjeravanje je onoga što radi *spell checker* opet nešto drugo. Pravopis predstavlja skup pravila o pravilnome pisanju, priručnik s uputama o pisanim praksama kojega standardnoga jezika. (Badurina 2012: 65) Neka su od poglavlja u pravopisu pisanje velikoga i maloga slova, pisanje zareza, pisanje pravopisnih i interpunkcijskih znakova, pisanje *č*, *ć*, *đ*, *dž*, pisanje *ije* i *je* itd.[32] Računalna provjera pravopisa može provjeravati samo dio onoga što je zabilježeno u pravopisnim priručnicima. Zbog toga je naziv *računalna provjera pravopisa* odgovarajuć, ali koristi se i u ovom radu zbog nedostatka bolje zamjene za engleski izraz *spell checker*.

Računalna provjera pravopisa, onako kako to radi Hunspell, ne može provjeravati zarez, što je dijelom zbog dizajna Hunspella, a dijelom zbog aplikacija koje koriste Hunspell za provjeru pravopisa. Ovdje se ne može ulaziti u detalje kako Hunspell radi kao program, ali ono što je važno za onoga tko piše pravila za Hunspell i za korisnike jest činjenica da Hunspell ne zna što je to rečenica, provjera se vrši na razini riječi čak ignorirajući i pravopisne i interpunkcijske znakove. Hunspell iz naredbenoga retka razbija tekst na riječi. Riječ je svaki niz slova i brojeva koji su odijeljeni bjelinom ili bjelinom i krajem teksta (zadnja riječ u dokumentu ili retku). Takva je definicija riječi prilično jednostavna jer inače lingvisti puno teže definiraju riječ, a u *Uvodu u jezičnu morfologiju* I. Marković razlikuje pravopisnu, leksičku i gramatičku riječ (str. 6 – 12) Pravopisna je riječ u tekstu omeđena bjelinama, leksem je rječnička riječ: neostvarena apstraktna jedinica umnoga leksika. Gramatička je riječ dio paradigme, ostvaraj leksema kakav se pojavljuje u sintaksi, odnosno nalazi se na razmeđi morfologije i sintakse. Naravno, iz ponuđenih je definicija, odnosno mogućnosti gledanja na i definiranja riječi, vidljivo da u različitim jezicima svijeta nije isto težište svake od ponuđenih definicija, odnosno udio svake od kuta gledanja nije jednako zastupljen u općem shvaćanju što riječ jest i što ju čini.

32 http://pravopis.hr/pravila/; pristupljeno 14. studenoga 2017.

Računalo općenito nema nikakvo shvaćanje značenja. U složenijim sustavima značenje je moguće fingirati postoji li pristup označenom velikom korpusu teksta, ali Hunspell ne nudi takvu mogućnost pa je onda definicija riječi za Hunspell vrlo jednostavna: svaki skup slova između dvije bjeline ili bjeline i kraja dokumenta i izvan konteksta. Računalna provjera pravopisa provjerava samo postoji li određeni oblik riječi, odnosno provjerava li dozvoljava li rječnik i dozvoljavaju li pravila u datoteci afikasa određeni oblik, ali u potpunosti ignorirajući odnose između riječi. Do neke se mjere odnos može provjeravati računalnom provjerom gramatike, ali to je tema za neki drugi rad.

Hunspell će rečenicu *Vani pada kiša.* rastaviti na [1] Vani, [2] pada i [3] kiša te će provjeriti svaku riječ posebno nalazi li se u rječniku. Ne nalazi li se riječ na popisu, Hunspell će predložiti najsličniju, formalno najbližu riječ onoj koja je napisana. Bliskost riječi nije određena značenjem ili paradigmatskim oblicima već formalnom bliskosti riječi.

Zbog takva pristupa moguće su i neke pogreške u tekstu koje računalna provjera pravopisa ne može provjeravati. Računalna provjera pravopisa ne može razlikovati i ispravljati situacije kada korisnik ne razlikuje *č* i *ć*, *ije* i *je*, a njihova zamjena mijenja značenje. Pogriješi li korisnik i napiše li *kreči* umjesto *kreći,* provjera pravopisa neće prepoznati grešku jer u hrvatskome jeziku postoji i riječ *kreči* (imperativ od glagola *krečiti*) i riječ *kreći* (imperativ od glagola *kretati*). Tu su i druge situacije koje računalna provjera pravopisa ne može provjeravati. U rečenici *Trebali bismo*. računalna provjera pravopisa ne može ispravljati pogreške poput *Trebali bi*. zbog dvije stvari: valjani su oblici i *bi* i *bismo*, a računalna provjera pravopisa ne može procijeniti je li korisnik želio/trebao napisati *bi* ili *bismo* jer su moguće rečenice i *Trebali bismo* (mi bismo trebali) i *Trebali bi* (oni bi trebali).

Mana računalna provjera pravopisa ima još: ponekad se može dogoditi da zamjena slova rezultira slijedom slova koji odgovara stvarnoj riječi. Ako se pretpostavi da korisnik želi napisati *tako*, a umjesto toga zamijeni *ak* u *ka* dobit će se riječ *tkao* (gl. prid. radni od glagola *tkati*) – postojeća riječ, ali i pogreška na koju računalo neće upozoriti korisnika jer Hunspell nema mehanizam koji bi omogućio prosuđivanje je li korisnik doista i želio napisati *tkao* ili se radi o zamjeni slova i zapravo je trebalo pisati *tako*. Isto se može događati i korisnicima koji ne razlikuju *p* i *b* i općenito miješaju glasove koji dijele opreku po zvučnosti. Tako, zapravo pogreška, *snop* možda treba biti ispravljena u *snob* ili obrnuto, ali računalna provjera pravopisa Hunspellom to neće moći.

Prethodno je navedene probleme moguće rješavati računalnom provjerom gramatike, ali i opet samo do neke mjere. Provjera gramatike može u obzir uzimati nekoliko riječi

odjednom, ponekad i cijele rečenice ili čak odlomke, ali i opet postoje situacije u kojima rečenice mogu biti formalno točne, ali da još uvijek budu pogrešne u odnosu na ono što je korisnik želio napisati. Postoje dva alata koji omogućavaju provjeru pravopisa na obrađivaču teksta LibreOffice. Jedan je od dva LanguageTool[33], a drugi je LightProof[34]. Prvi je pisan programskim jezikom Java i dostupan je kao samostalna aplikacija ili kao dodatak za LibreOffice (i druge alate). LightProof je pisan programskim jezikom Python i time je zanimljiviji razvijateljima i podržavateljima otvorenoga koda jer je i sâm Python otvorena kôda: lakše ga je integrirati u druge sustave, ne postoji ovisnost o drugom proizvodu, programski je jezik dobro dokumentiran, a pomoć nude i drugi korisnici na forumima i sl.

Čak ni provjera gramatike ne može riješiti sve probleme, odnosno ispraviti sve pogreške koje korisnik može napraviti. Na engleskoj Wikipediji stoji šaljiva pjesmica koja dobro ilustrira problem, prenosim samo prvu kiticu[35]:

```
Eye have a spelling chequer,
It came with my Pea Sea.
It plane lee marks four my revue
Miss Steaks I can knot sea.
```

Primjer 10: Nedostatak provjeravanja riječi u izolaciji

Pretpostavka je da engleski ima više takvih zvukovnih homonima čisto zbog engleskoga pravopisa i načina na koji engleski bilježi foneme: jednom na ovaj način, drugi puta na drugi. Hrvatski vjerojatno ima manje ovakvih slučajeva jer je pravopis pretežito fonološki pa je i manje zvukovnih homonima, odnosno kada do homonima i dolazi, uglavnom se oni i jednako pišu pa je i mogućnost ovakvih grešaka manja. Međutim, smanjena mogućnost pojavljivanja pogrešaka ovoga tipa ne poništava činjenicu da mana toga tipa s ovakvim alatima ipak postoji.

Prigovor je ovakvoj provjeri pravopisa da takva provjera ne barata jezičnim modelom jezika koji se provjerava. Jezični bi model trebao moći u *Eye have a spelling chequer* ispraviti *Eye* u *I* te *chequer* u *checker* jer takvi modeli imaju podataka o tome koje se riječi najčešće nalaze ispred i iza *have*, odnosno *spelling*. Na sličan način Google predlaže drugu riječ nakon

33 https://languagetool.org/; pristupljeno 14. studenoga 2017.
34 https://cgit.freedesktop.org/libreoffice/lightproof/; pristupljeno 14. studenoga 2017.
35 https://en.wikipedia.org/wiki/Spell_checker; pristupljeno 14. studenoga 2017.

što je korisnik napisao samo prvu jer iz baze podataka izvlači koje se riječi najčešće nalaze iz prve. Naravno, ta tvrtka to radi radi poboljšavanja rezultata pretraživanja i veću će važnost dati aktualnim događanjima iz popularne kulture negoli što se provjeravati cjelokupan korpus hrvatskoga jezika.[36] Iako bi takvi programi mogli pomoći u ispravljanju velikoga dijela situacija u hrvatskome standardnome jeziku gdje provjera pravopisa bez takva modela ne može, recimo *U prvoj je zamijeni izbornik* u *U prvoj je zamjeni izbornik*. Budući da su i *zamijeni* (imperativ od *zamijeniti*) i *zamjeni* (DL od *zamjena*) ispravni oblici riječi, ali je razlika u pisanju *ije* i *je* u čemu korisnici hrvatskoga standardnoga jezika često griješe, provjera pravopisa s razvijenim modelom jezika omogućila bi ispravljanje *u prvoj je zamijeni izbornik* jer bi model sugerirao da između *je* i *izbornik* može ići samo *zamjeni*. Međutim, i tu može doći do pogrešaka jer postoje primjeri u kojima se može odrediti treba li pisati *ije* ili *je* moguće tek iz širega konteksta.

Bez obzira na to što Hunspell nema model, još uvijek se radi o dobrom načinu provjeravanja grešaka u pisanju. Naime, ako je *chequer* ispravno napisana riječ, onda tu riječ i ne treba označavati kao pravopisnu pogrešku. Teško da će korisnik u stvarnoj situaciji napisati *spelling chequer*, odnosno ako korisnik i pogriješi pa ne napiše ni *checker* ni *chequer*, Hunspell će ponuditi oba rješenja tako da korisniku ostaje mogućnost izbora. Još se uvijek radi o alatu koji pomaže u ispravljanju pogrešaka, a činjenica da se radi u relativno malom alatu, koji je ujedno i slobodan program, znači da je računalnu provjeru pravopisa za hrvatski jezik moguće uključiti u različite druge aplikacije bez teškoća i složenosti koje bi tražile aplikacije sa složenim jezičnim modelima. Uvijek je problem i treniranja programa s modelom: koji korpus, količina korpusa, samo standardni ili i razgovorni jezik, koji tip teksta, treba li ili ne treba uključivati i beletristiku... Nedostatak je Hunspella moguće donekle ublažiti uključivanjem i računalne provjere gramatike alatima slobodnoga koda. Kako se radi o alatima koji nisu povezani s Hunspellom, to bi trebalo istražiti i obraditi posebno.

4.1. Neki nedostaci računalne provjere pravopisa Hunspellom

U rječnik za računalnu provjeru pravopisa ovoga tipa – bez jezičnoga modela – nije poželjno uključivati što više riječi, nije dobro ići na količinu. Riječi se unose u datoteku `hr_HR.dic`, ali povećanjem se broja riječi povećava i mogućnost da pogrešno pisanje bude izgledom jednako i točno napisanoj riječi: *zamijeni* (glagol) ~ *zamjeni* (imenica). Primjera ima

36 https://support.google.com/websearch/answer/106230; pristupljeno 9. prosinca 2017.

još: *spavačica ~ spavaćica*, *ču* (aorist od *čuti*) ~ *ću* (pom. gl. *biti*), ali do pogrešaka može doći i izostavljanjem ili dodavanjem nekoga slova. U svakom je slučaju rezultat riječ koju pravopis prepoznaje točnom, ali je u konkretnom tekstu pogreška. Takve se situacije ponekad mogu riješiti računalnom provjerom gramatike ili upotrebom složenijega programa za računalnu provjeru pravopisa, ali kako je već spomenuto: to može otežati integraciju računalne provjere kao programa u druge programe. Pogreške zamjenom ili izostavljanjem slova su češće kada su slova koja se nalaze jedno pored drugoga na tipkovnici, a njihova zamjena tvori novu riječ. Ponekad do pogrešaka može doći čak i miješanjem velikoga i maloga slova. Evo nekoliko primjera koje sam zabilježio upotrebom rječnika koji održavam na `https://github.com/krunose/hr-hunspell`: *puta ~ pita*, *čijem ~ čujem*, *plače ~ plaće*, *kreče ~ kreće*, *sjedi ~ sijedi*, *tako ~ tkao, ako ~ kao, Marka* (Marko) ~ *marka* (poštanska), *Višnja* (ime) ~ *višnja* (voće), *ugašen ~ ugažen*, *opći ~ opeći*, *povrijeđene* ~ povrijeđenje, *preobraćenje ~ preobraženje*, *pućenje* (usana) ~ *pušenje, biotip ~ biotop*, *obje ~ oboje*, *bik ~ bok, grčenje ~ grđenje*, *navoženje ~ navođenje*, *biotip ~ biotop*, *dušnik ~ dužnik*, *duranje ~ durenje*, *tjelašce ~ tjelešce*, *ulegnuće ~ uleknuće*, *uparenje* (upariti) ~ *uperenje* (uperiti), *miješanje* (miješati) ~ *miješenje* (*mijesiti*), *navoženje* (navoziti) ~ *navođenje* (navoditi)...

Propuštanje grešaka nije, naravno, moguće i u slučajevima koji doista i jesu pravopisni problemi poput pisanja *č* i *ć*, *ije* i *je* itd.: *plače* (glagol) ~ *plaće* (imenica), *kreče* (krečiti) ~ *kreće* (kretati), *plače* (plakati) ~ *plaći* (plaća), *sjedi ~ sijedi*, *sljedeći ~ slijedeći*, *sijaći ~ sijači*, *primjeni* (imenica) ~ *promijeni* (glagol), *zalisak ~ zalizak*, *vraćati* (vratiti) ~ *vračati* (vračati), *čete* (četa) ~ *ćete* (htjeti), *skučenje* (skučiti se) ~ *skućenje* (skućiti)...

Pogreške mogu nastati i zbog umetanja ili izostavljanja razmaka: *pedeseti pet* → *pedeset i pet*. Računalo će prihvatiti i prvi primjer jer su obje riječi pravilne, a ono ne može odlučivati o kontekstu. Ova je situacija ispravljiva računalnom provjerom gramatike gdje bi pravilo trebalo biti formulirano tako da ako je prva riječ broj kojemu je dodan *i*, a druga je riječ opet broj, treba umetnuti razmak između prvoga broja i *i*. Isto tako, nije moguće ispravljati upotrebu *bi*, *bismo* i *biste* jer su sve riječi pravilne i točne pa će doći do propuštanja primjera *tražili bi* i za *tražili bismo*, *tražili biste*. Ovo je samo donekle ispravljivo računalnom provjerom gramatike.

Međutim, problem mogu biti i drugi parovi i primjeri: *Koreanac ~ Korejac*, *Koreanka ~ Korejka*, *Kur'an ~ Kuran*, *rušenje ~ ruženje, skućenje ~ skučenje*, *ulupljenje ~ ulubljenje*, *zagašenje ~ zagaženje*, *gvirc ~ kvirc ~ gverc*. Onaj tko radi rječnik treba odlučiti koje riječi uključiti u rječnik, a koje ne. Nije problem ako se oblici različitih riječi poklapaju: *rak*

(životinja) ~ *rak* (bolest), odnosno ako imaju potpuno jednaku paradigmu jer računalo ionako ne može razlikovati, recimo, odnos živo ~ neživo pa će za *rak* reći i nominativ i akuzativ, a za *raka* će reći i genitiv i akuzativ. Nema načina da računalo iz konteksta izvuče o kojem se slučaju radi. Takve se riječi koje imaju istu paradigmu mogu pisati i kao jedan unos u rječniku, ne treba ih rastavljati u dva. Ono što smatram da bi svakako bilo dobro rastaviti na zasebne unose riječi koje imaju različite paradigme: *sat*: *sati ~ satovi*, *pas*: *pasa ~ psa*...

Važno je voditi računa da je rječnik za provjeru pravopisa rječnik koji treba pomoći korisnicima koristiti hrvatski standardni jezik pa je onda logično da rječnik sadržava riječi iz hrvatskoga standardnoga jezika, ali potrebno je i širiti se izvan toga na razgovorni rječnik i slično, ali samo onoliko koliko dopušta uzus i samo za riječi koje prihvaća jezična zajednica. Zbog toga je širenja ponekad teško odlučiti unijeti u rječnik *žmukljer* ili *žmukler* ili pak i jedno i drugo. Radi se o žargonizmima i slengu koji nije dio standardnoga jezika pa onda postoje i varijacije. Ponekad je jedna varijanta češća od druge, ali to je tako samo u manjem broju slučajeva. Unositi se riječi u rječnik treba oprezno.

Postoji još jedan tip pogreške koju nije moguće riješiti računalnom provjerom pravopisa, a ponekad čak ni računalnom provjerom gramatike. Razlika je piše li *ne podcrtavanje* i piše li *nepodcrtavanje*, isto tako *ne važan* i *nevažan, je li* (upitna formacija) i *jeli* (jesti)... U tome slučaju treba li što pisati sastavljeno ili nesastavljeno ovisi o tome što je onaj tko piše zapravo mislio, a to računalna provjera pravopisa i računalna provjera gramatike ne mogu znati.

4.2. Način uvrštavanja i odabir riječi za rječnik

Prve se dvije situacije iz prethodnoga potpoglavlja teško mogu zaobići. Doduše, računalna provjera pravopisa služi kao pomoć (engl. *aid*) prilikom pisanja. Ne može se korisnika u potpunosti osloboditi odgovornosti pa da računalo piše umjesto njega. Dogodit će se pogreška s vremena na vrijeme, računalna će ju provjera pravopisa propustiti, ali jedan se dio može ispraviti računalnom provjerom gramatike.

Zadnja skupina primjera u prethodnom potpoglavlju ima drugi problem: nešto ih je lakše zaobići, ali važno je postaviti čvrsta pravila o uvrštavanju riječi u rječnik. Ako se prihvati da je bolje imati manje riječi negoli više pa da slučajne pogreške budi češće propuštene, onda je potrebno donijeti kriterije s obzirom na stilsku, vremensku i prostornu raslojenost leksika kao i s obzirom na status riječi u standardnome jeziku. Tako žargonizme ne

treba uvrštavati, pretjerano knjiški leksik također. Takve se riječi koriste rijetko, mogu biti slične s nekom drugom riječi u rječniku i rezultirati nepodcrtavanjem riječi. Što se prostorne raslojenosti tiče, ne treba unositi ni lokalizme, regionalizme ili dijalektizme jer nisu dijelom standardnoga jezika, velika je mogućnost postojanja više inačica iste riječi što bi onda značilo da bi u rječnik trebalo unositi i sve inačice riječi što zapravo onemogućava kvalitetnu provjeru pravopisa. Isto se odnosi na vremensku raslojenost pa one riječi koje se danas više ne nalaze u aktivnoj upotrebni nema potrebe unositi u rječnik, nepotrebno se otežava održavanje. O funkcionalnoj će raslojenosti nešto više biti riječi u nastavku, no u tome kontekstu ovdje treba napomenuti da velik problem predstavljaju i termini pojedinih struka jer treba odlučiti uključivati li termine ili ne, a ako da – koje struke, što ako se odluči ne uključivati termine, a dio se tih i takvih riječi može čuti i u znanstveno-popularnome stilu na radiju, televiziji. Što sa tekstovima koji se pišu za razne portale (vijesti, trgovina, računala), Wikipedia? Postaviti čvrste kriterije nije moguće ako se prvo ne definira što riječ i jest.

4.2.1. Riječ u kontekstu računalne provjere pravopisa

Govornici nekoga jezika u pravilu lako prepoznaju riječi toga jezika, ali nemaju svi jezici riječ u istom smislu, ne čini riječ u svim jezicima isto. (Marković 2012: 4) Riječ je najmanji slobodni oblik. Vezani su oni oblici koji se pojavljuju s drugima, recimo riječi na *-ost*: *mladost*, a slobodni su oni koji mogu činiti rečenicu. Nemaju svi jezici vezane oblike, recimo kineski, ali ono što je riječ jest najmanji neslobodni oblik: sintagma je sastavljena od više vezanih oblika, ali riječ je najmanji neslobodni oblik jer se ne može rastaviti na manje slobodne oblike. (Marković 2012: 5)

Bez obzira na prethodno određenje, još uvijek može biti teško definirati koji oblik, i na koji način uvrstiti u rječnik jer postoji *pravopisna* riječ, *rječnička* riječ, *gramatička* riječ te *fonološka*, odnosno *fonetska* riječ. Pravopisna riječ je jedinica koja je u pisanome tekstu s obje strane omeđena bjelinom. (Marković 2012: 7) Međutim, takvo definiranje može biti problematično jer riječi u pisanome tekstu nisu uvijek odijeljene bjelinom. Na kraju se rečenice, na zadnjoj riječi u rečenici piše pravopisni znak. Unutar rečenice pišu se različiti pravopisni znakovi poput zareza ili navodnih znakova, koji nisu riječ, ali zbog njih riječ nije omeđena uvijek bjelinom. Neka računalna datoteka može imati samo jednu riječ i time neće biti omeđena bjelinama uopće. Što sa sufiksoidima i prefiksoidima koji mogu biti pravopisni problem, a ne moraju biti odijeljeni bjelinama ni s jedne strane ako se radi o sufiksoidu koji je s jedne strane omeđen crticom, a s druge pravopisnim znakom: *To je sufiksoid -fob.* U

rječnicima[37] se nalaze i takvi oblici iako ne zadovoljavaju kriterij najmanjega vezanoga oblika jer i sami nisu samostalni, odnosno od tih se elemenata tvore drugi najmanji vezani oblici. K tome, u navedenom primjeru, prefiks nije čak ni omeđen bjelinama.

Rječnička riječ je potencijalna, neostvarena jedinica općega umnoga rječnika, odnosno ukupnost oblika i značenja neke riječi. Problem je u tome što se tako ne mogu objasniti i supletivne osnove (Marković 2012: 7) Odnos *čovjek ~ ljudi* čini zapravo jednu riječ, ali zbog ukupnosti svojih oblika i osnove, često se navode kao dvije riječi. Pitanje je treba li takve primjere pisati odvojeno u rječnik ili ih treba navoditi kao jednu riječ, ali onda u definiranju afikasa uklanjati praktično cijele osnove. Međutim, uklanjanje osnove u *čovjek* i dodavanje nove osnove pomoću afikasa protivno je gramatici. Praktično se na nultu osnovu dodaje sufiks *ljud-* i onda još gramatički morfem *i*: *čovjek* > ø + ljud + i. Iako možda neopravdano u smislu pisanih gramatika, mislim da je ovo zapravo najbolji način jer postoje računalni programi koji koriste Hunspellove rječnike za omogućavanje elastičnijega pretraživanja[38]. Tko je vršio pretraživanje bez elastičnoga pretraživanja za riječ je *čovjek* dobio samo rezultate gdje se spominje riječ *čovjek*, doslovno *čovjek*, nisu bivali uključeni i drugi oblici te riječi. Postoje alati koji omogućuju nešto fleksibilnije pretraživanje, recimo: *čovjek** ili čovjek*?* pa bi rezultati pretraživanja dali i *čovjek* i *čovjeka* i *čovjeku*... Međutim, što kada se pretražuje neka sintagma, možda čak i ako se upišu četiri ili pet riječi: *računalna provjera pravopisa za hrvatski*. Elastičnije će pretraživanje korisniku ponuditi i rezultate s obzirom na sve oblike svake od riječi u sintagmi: *računalne provjere pravopisa za hrvatski*, *računalnoj provjeri pravopisa za hrvatski* itd. Znam da se Hunspellov rječnik u te svrhe koristi u alatu ElasticSearch[39], ali to je izvan teme ovoga rada. Ono što je važno jest da u Hunspellov rječnik nije dobro posebne unose raditi za *čovjek* i *ljudi* jer onda program neće moći prepoznati da je oblik *ljudi* množina imenica *čovjek* pa bi pokoji rezultat mogao biti izostavljen. Nedostatak je ovoga pristupa što je potrebno napraviti poseban deklinacijski obrazac mimo svega što stoji u gramatikama isključivo za riječ *čovjek*, a onda i svaku takvu riječ ispočetka, bez obzira što, zanemari li se supletivna osnova, takav deklinacijski obrazac imaju i druge riječi. Međutim, na riječi se *čovjek* ne može koristiti obrazac kao na svim drugim takvim riječima jer je potrebno zamijeniti osnove novim deklinacijskim obrascem.

Gramatička je riječ dio paradigme, odnosno onaj oblik koji se pojavljuje unutar sintakse s pridruženim morfosintaktičkim obilježjima. (Marković 2012: 11) Koristi li se ovaj

37 http://hjp.znanje.hr; pristupljeno 30. studenoga 2017.

38 http://www.elastic.co/; pristupljeno 30. studenoga 2017.

39 https://github.com/elastic/elasticsearch; pristupljeno 30. studenoga 2017.

kriterij onda bi svaki oblik iste riječi trebao posebno biti unesen u rječnik. Ovakav bi pristup onemogućio druge alate da prepoznaju oblike iste riječi, odnosno oblici bi iste riječi bili tumačeni kao zasebne riječi.

Fonološka ili fonetska je riječ jedinica koja je djelokrug nekoga fonološkoga procesa u nekome jeziku. U hrvatskom je odlučujući fonološki proces jedan naglasak. (Marković 2012: 19) Ovaj princip u definiranju riječi u hrvatskome standardnome jeziku nije moguće koristiti jer se ne piše prema izgovoru: *ne znam*, *ne trebati*, ali *nemati*.

Iz svega proizlazi da bi za izradu rječnika za provjeru pravopisa Hunspellom najbolje bilo kombinirati pojedine pristupe definiranju riječi. U rječnik se moraju upisivati riječi neovisno o fonološkom kriteriju, odnosno svaka riječ je ono što je odijeljeno bjelinama, ali može biti i manja jedinica od najmanje slobodne jedinice zbog prefiksoida i sufiksoida koji se u tekstu ponekad mogu javiti, a netko ih može pogrešno napisati, recimo *avijo-* radi ispravka u *avio-*. U rječniku svoj unos može imati samo jedna riječ, a svi se ostali oblici te riječi moraju raditi pomoću afikasa. Mora se poštivati i rječnički kriterij jer ono što je u rječniku uvedeno kao nova riječ *čovjek*, *čovječji*, treba tako biti i u rječniku za računalnu provjeru pravopisa. Tvorbu ne treba provoditi pomoću afikasa nego unosom nove riječi. Ako bi se od riječi *Platon* tvorbeno izvodila riječ *platonski*, onda bi rječnik prihvatio i *Platonski* usred rečenica, ali i *platon* iako treba *platonski* i samo *Platon.* Da bi se to moglo postići, potrebno je imati poseban unos za *Platon*, a poseban za *platonski*. Isto je i sa imenima koja su jednaka sa općim imenicama: *Višnja ~ višnja*, ali i one koje se svojim oblicima u paradigmi poklapaju s općim imenicama: *Marka* (Gjd) ~ *marke* (poštanska). Opće imenice imaju i množinu (višanja), ali osobna imena, pisana velikim slovom, najčešće imaju samo jedninu.

Tvorbu afiksima treba izbjegavati jer je teško i prefikse dodavati glagolima. Prefiksa za glagole ima puno, teško je pronaći način da se pouzdano zna koji glagol može imati koje prefikse. Iz rječnika bi se sa `https://github.com/krunose/hr-hunspell` mogli izvući svi glagoli, moglo bi se ispisati sve osnove, za svaku bi se osnovu mogli odrediti svi oblici, odnosno izvući svi prefigirani oblici konkretne osnove i tako za sve osnove. Zatim bi se mogla ukloniti sva ponavljanja. Takve bi prefikse trebalo posebno unositi u datoteku s afiksima, ali bi isto trebalo takve konjugacijske klase raditi s obzirom na tip glagola, s obzirom na promjene po konjugacijskima oblicima, uzimajući u obzir i glasovne promjene. Budući da se ovaj rad bavi imenicama, neću dublje ulaziti u razrađivanje glagola, ali lakše je i modularnije glagol sa svakim prefiksom unijeti kao zasebnu riječ, iako bi to moglo negativno utjecati na elastična pretraživanja. Doduše u manjoj mjeri negoli je to sa imenskim riječima.

Poseban su problem polusloženice, moguće je unositi svaki dio posebno, ali onda bi bili dozvoljeni i oblici koji u pravilu nisu mogući. Takve riječi treba unositi zajedno sa spojnicom: *spomen-ploča*, *više-manje*, *brže-bolje*.

Imena se gradova i naseljenih mjesta moraju upisivati posebno. Čudno je naići u rječniku na *Beli*, ali je potrebno kako bi računalo prihvaćalo *Beli Marof*. U ovom slučaju računalo neće prihvaćati *beli* nego će predlagati *Beli*. Isto tako i za *Marof*. Računalnom provjerom pravopisa Hunspellom nije moguće utjecati na pogrešno pisanje *Gorski Kotar* jer riječ *Kotar* može stajati i na početku riječi i računalo se protiv pisanja velikim slovom ne može buniti. Ono što se može jest provjerom gramatike tražiti primjere u kojima iza riječi *Gorski* pisane velikim slovom slijedi riječ *Kotar* pisana velikim slovom, da ispravlja u *Gorski kotari.* Isto tako, računalna provjera pravopisa Hunspellom neće moći ispraviti pogreške *gorski kotar* jer je riječ *gorski* isto moguće pisati malim slovom kada se radi o pridjevu. Iz tih je razloga potrebno imati dva unosa: *gorski* malim slovom i *Gorski* pisano velikim slovom. Ovaj pristup olakšava održavanje i dopunjavanje rječnika.

Nepromjenjive riječi s navescima poput priloga *sad* i *sada*, mogu se rješavati, iako možda ne u skladu s objašnjenjima navezaka u gramatikama, tako da se kao zaseban unos – kao nova riječ – unese samo *sad*, a da se pomoću dodavanja nastavaka, klasom na taj unos dodaje navezak *-a*. To je važno jer bi onda elastičnije pretraživanje moglo uvrstiti i rezultate pretraživanja i sa *sad* i sa *sada* jer se zapravo i radi o jednoj riječi.

4.2.2. Kriteriji za uvrštavanje

S obzirom na probleme koji su izneseni do sada, postavlja se pitanje uspostavljanja kriterija za uvrštavanje riječi u rječnik. Prvo treba odlučiti kako tretirati riječi koje u strogom smislu nisu dijelom hrvatskoga standardnoga jezika, odnosno hoće li se – i treba li – u rječnik unositi i nestandardne riječi, žargonizme, vulgarizme, termine manje poznatih struka i slično. Što je s riječima koje pripadaju književno-umjetničkome stilu?

Kada se govori o stilovima hrvatskoga jezika, govori se o pet funkcionalnih stilova: znanstvenome stilu, administrativno-poslovnome stilu, novinarsko-publicističkome stilu, književnoumjetničkome (beletrističkome) stilu te razgovornome stilu. (Silić 2006: 43 – 108)

Jezik ne funkcionira u svim situacijama jednako, odnosno u jednoj će situaciji hrvatski standardni jezik biti korišten na jedan način, a u nekoj drugoj situaciji na drugi način. Razlike u korištenju jezika ne znači da poštivanje načina korištenja jezika u jednoj situaciji i istovremeno nepoštivanje pravila koji vrijede za neku drugu znači da se ne koristi hrvatskim

standardnim jezikom, odnosno da se ne poštivaju pravila i zakonitosti hrvatskoga standardnoga jezika. (Silić 2006: 36)

Međutim, gramatička i stilistička pravila koja vrijede s obzirom na stilove ne znači da leksik prati takvo razdjeljivanje, odnosno ne postoje riječi koje pripadaju isključivo znanstveno-popularnome ili pak riječi koje pripadaju isključivo administrativno-popularnome stilu. Zbog toga se ne može reći da u rječnik treba uvrštavati riječi iz ovoga ili onoga stila, već o tome treba odlučiti s obzirom na ulogu rječnika, a nju se može definirati s obzirom na leksičke i stilske slobode kojega od stilova.

Književnoumjetnički je stil najslobodniji, u njemu se može očekivati gotovo svaka riječ koja pripada i koja ne pripada hrvatskome jeziku, velika je mogućnost biranja različitih riječi, a čak je dozvoljeno i nepoštivanje pravopisnih i gramatičkih pravila hrvatskoga jezika. U tom je slučaju teže raditi rječnik za provjeru pravopisa koji bi koristio korisniku za pisanje književnoumjetničkih tekstova, pogotovo radi li se o poeziji. Rječnik može pomoći onome tko piše roman ili pripovijetku, ali će se u takvim tekstovima nužno naći i riječi koje se u svakoj drugoj komunikaciji ne koriste. Dodavanje bi svih riječi opterećivalo rječnik a uz neznatnu korist. Izrazito knjiške riječi ne treba uvrštavati u rječnik. Provjera će pravopisa pomoći onome tko piše i roman ili pripovijetku i bez tih riječi. Tko zna za njih, zna ih i pisati. Naravno, može pogriješiti, ali bilo bi neopravdano unositi takve riječi u rječnik i zbog toga što postoji više oblika za istu knjišku riječ. To bi značilo da se, osim male frekventnosti, u rječnik trebaju unositi i različiti oblici nefrekventnih riječi koje će rječnik za računalnu provjeru pravopisa predlagati kao zamjenu za pravopisnu pogrešku čak i u situacijama kada nitko i ne pokušava napisati takvu knjišku riječ. Primjer neka bude *ljuba* i *ljubovca*. Radi se o riječi za isto, ali u različitim inačicama. Navedene su riječi u rječnicima označene knjiškima i smatram da ih ne treba uključivati u rječnik za računalnu provjeru pravopisa zbog niske frekventnosti u hrvatskome standardnome jeziku i zbog izrazite stilske obojenosti, a takve bi riječi, odluči li se unositi ih u rječnik, opterećivale rječnik i otežavale održavanje, ponekad takve riječi zahtijevaju i posebne deklinacijske ili konjugacijske obrasce što nepotrebno usložnjava izradu i održavanje rječnika.

Isto je i s riječima koje su u rječnicima označene arhaičnima ili zastarjelima, odnosno nefrekventne riječi označene kao etnološke i ostale specifičnosti tipa *čarlama*. Radi se o vrsti narodnoga plesa, a još k tome ima i *č* u sebi pa se može to koristiti kao razlog za uvrštavanje, ali mislim da je bolje takve stvari izostavljati. Onaj tko treba takve riječi, lako ih može dodati u osobni rječnik, odnosno takve riječi korisnik i sam treba dodavati u osobni rječnik s

obzirom na aplikaciju koju koristi jer svaki obrađivač teksta ima osobni rječnik koji nije povezan ni sa osnovnim rječnikom za računalnu provjeru pravopisa ni sa osobnim rječnicima drugih programa i aplikacija. To bi omogućilo bolju računalnu provjeru pravopisa za toga korisnika, a za druge korisnike takve riječi ne bi opterećivale rječnik.

Što se tiče povijesnih i drugih specifičnosti, one riječi koje su poznatije, odnosno mogu se još uvijek naći u svakodnevnome govoru, barem u frazemu ili u šali: *gulag*, treba dodavati na popis riječi u rječnik. Isto tako, smatram da nazive za davno izumrla zanimanja koja se više nigdje ne spominju osim u rječnicima, ne treba uvrštavati. Recimo *ornator* je riječ slična riječi *orator* a starorimski je ekvivalent današnjemu frizeru. Kao što se vidi iz priloženoga, razlika je između riječi u samo jednome slovu, a mala je vjerojatnost da će netko htjeti pisati *ornator*. Onaj tko želi, i zna za tu riječ pa je može dodati u osobni rječnik bez obzira na korpus riječi u osnovnom rječniku. Tko treba takvu riječ, može ju dodati u osobni rječnik, a ona neće opterećivati rječnik za druge korisnike.

Takve, specifične riječi, ulaze već i u domenu znanstveno-popularnoga stila, ali i u domenu termina. Termine koji se koriste isključivo u jednoj znanstvenoj disciplini, ili su dio više znanstvenih disciplina, ali se rijetko javljaju u tekstovima izvan te struke, ne treba uvrštavati u rječnik. Razlog je taj što je potreban jedan opći rječnik koji će biti bazom, a onda se termini pojedine struke unose u drugi, novi rječnik koji se koristi kao dodatak temeljnome rječniku. Isto se može napraviti i sa stilski obilježenim riječima pa oni koji pišu književnost mogu rječnik s popisom takvih riječi koristiti uz temeljni rječnik. Nazive svih kemijskih elemenata ne treba unositi, tek ono što je poznatije (kisik, dušik). Često su kemijski elementi i spojevi kraće riječi pa se pojavljuju kao prijedlozi i tamo gdje ih se ne očekuje. Isto je i s oznakama za kemijske elemente. Često se sastoje od svega dva-tri slova pa računalo nudi i preko dvadesetak prijedloga ako se pogriješi u pisanju riječi od tri slova, što onda otežava prihvaćanje prijedloga jer je točan prijedlog potrebno tražiti u velikom nizu ponuđenoga. Problem su i imena biljaka i životinja. Ne može se reći da je neka biljka ili životinja važnija od neke druge biljke ili životinje, ali u rječnik treba dodavati ono što se po jezičnom osjećaju izvornoga govornika čuje češće, što bi se moglo uvrstiti u opći leksik, a ono što je manje često, ne treba unositi u rječnik. Kao za svaku drugu terminologiju, moguće je napraviti dodatni rječnik i koristiti taj rječnik kao dodatak.

Razgovorni je stil isto šarolik. One riječi koje možda ne pripadaju visokom standardnome jeziku, ali česte su u potrebi, moraju se nalaziti u rječniku. Ovakvi se rječnici mogu koristiti i za pisanje poruka elektroničke pošte, a takva komunikacija ne mora uvijek

biti službena, ali i opet je zadaća rječnika za računalnu provjeru pravopisa da pomogne korisniku ne pogriješiti u pisanju *č* i *ć* ili *ije* i *je*. Potrebno je dodavati i žargonizme, ali i vulgarizme koji su mogući u svakodnevnoj komunikaciji. Naravno, neće se prepisivati čitav rječnik žargonizama, ali kao izvorni govornik hrvatskoga jezika i korisnik rječnika za računalnu provjeru pravopisa, očekivao bih takve riječi na popisu. Ipak, ne treba uvrštavati lokalne izraze i fraze, to korisnik može dodati u osobni rječnik, ali riječi koje nisu u strogom smislu standardnoga jezika treba uvrštavati samo ako su ipak u određenoj mjeri prihvaćene na cijelom prostoru Republike Hrvatske. Ako se prihvati da je standardna riječ *računalo*, a da je razgovorno *kompjutor*, postavlja se pitanje treba li u rječnik unositi *kompjutor*, *kompjuter* ili oboje. Pravopisni rječnici bilježe *kompjutor*, ali se na terenu najčešće čuje *kompjuter*. Isto je i sa *filter* i *filtar* itd. Mislim da bi u rječnike trebalo unositi samo one riječi koje se nalaze u pravopisnim priručnicima, ako se nalaze, znači: *kompjutor* i *filtar*. Zadaća je računalne provjere pravopisa pomoći korisniku *pisati*, a pisati se korisniku može pomoći samo ako se koliko-toliko piše hrvatskim *standardnim* jezikom jer je standard normiran i postoje pravila za pisanje samo u *standardnome* jeziku. Postoje riječi koje su dio razgovornoga stila i ne nalaze se u pravopisnim priručnicima: *bajbuk* ~ *bajbok*, *gvirc* ~ *gverc*, *šaraf* ~ *šeraf*. Ponekad se može odrediti riječ koja je raširenija od alternativnoga oblika pa treba nju uključiti u rječnik. Ako pravopisni rječnik bilježi *šaraf*, onda tako treba i unositi u rječnik za provjeru pravopisa koji će onda za *šeraf* predložiti *šaraf*. Ako nije moguće odrediti koji bi oblik bio češći, a onaj tko radi rječnik ima osjećaj da bi riječ ipak trebalo uključiti u rječnik, uključiti treba oba oblika. Inače nijedan.

U kontekstu treba spomenuti da bi bolje bilo u rječnik ne unositi latinske i slične strane riječi. Mišljenja sam da strane riječi i izraze ne treba unositi u rječnik. Strane su riječi i izrazi, recimo latinske sintagme, često dio određene struke ili grane znanosti, određene djelatnosti pa bi se one mogle uvrštavati u posebne rječnike koji sadržavaju samo termine te struke. Na taj bi se način rasteretio osnovni rječnik, a onima koji imaju potrebe za posebnim riječima i izrazima, omogućila bi se provjera i tih riječi.

U rječnik bi isto tako trebalo uključiti sve države, ali onako kako se pišu u službenoj komunikaciji, onako kako je pravopisno točno, odnosno onako pišu institucije Republike Hrvatske. Često se može vidjeti i *Mjanmar* i *Mijanmar* i *Mjanma* i *Myanma* i *Myanmar*.[40] Treba odrediti što je službeno točno i što je u skladu s hrvatskim pravopisom i uključiti te

40 `http://hjp.znanje.hr/index.php?show=search_by_id&id=e1piXhE%3D&keyword=Mjanma`; pristupljeno 1. prosinca 2017.

oblik(e). Uglavnom, trebalo bi uključiti imena svih država, ali i važnijih gradova tih država, ne samo glavnih gradova. Ipak, treba voditi računa o službenom načinu pisanja imena tih gradova, ali i pravilima hrvatskoga pravopisa vezano uz poglavlje o pisanju stranih vlastitih imena. Isto vrijedi i za važnije planine, rijeke, mora, oceane, kontinente. Rijeke treba unositi samo važnije na razini svijeta, ali sve hrvatske. U rječnik ne treba dodavati sve hrvatske gradove, sela, zagrebačke, splitske i riječke kvartove i slično, treba unijeti važnije hrvatske gradove, ali i one gradove koji danas možda nisu od nekoga velikoga značaja, ali su bili u prošlosti. Od povijesnih osoba i političkih tvorevina, treba unositi samo one koji se nalaze u osnovnoškolskim i srednjoškolskim udžbenicima te možda još pokoje ime iz sveučilišnih udžbenika, ali ne treba dodavati apsolutno sve. Sve se takve osobne i političke tvorevine mogu unijeti u poseban rječnik koji se može koristiti kao dodatak temeljnome. Isto je i sa riječima karakterističnima za određeno povijesno razdoblje. Za riječi poput *bajoneta*, *bajuneta, bajonet* i *bajunet* treba provjeriti postoji li ustaljen izraz u vojnoj terminologiji pa onda u tom obliku unijeti u rječnik, inače će biti teško odabrati jedan od oblika. Iako jest riječ o *terminologiji*, ipak je se radi o riječi koju je moguće sresti u novinskim i novinarskim tekstovima pa bi bilo potrebno i imati je u rječniku. Ne treba unositi samo termine koji nisu poznati širem krugu ljudi. Neke će to struke pogoditi više neke manje, ali mislim da drugačije nije moguće. Prevelik rječnik nije moguće održavati, a prevelik rječnik neće točnu zamjenu dati prvu, što opet utječe na brzinu pregledavanja i lakoću korištenja rječnika za računalnu provjeru pravopisa. Puno je jednostavnije, brže i lakše prihvatiti prvu riječ s popisa prijedloga negoli je to petu-šestu jer u tome slučaju treba čitati cijeli popis. Isto je s *bilijar* i *biljar* i sličnim riječima. Treba provjeriti što stoji u pravopisu, ali i kako imenuju oni koji se time bave, recimo kakav športski savez i slično. Ako se ustanovi da su oba oblika raširena, treba unijeti oba. Isto kao *sport* i *šport*, budući da se danas sve češće čuje *sport*, treba unijeti, ali isto tako ima i onih koji govore i pišu *šport*, onda treba i to.

U svemu se može postaviti i pitanje unošenja riječi u rječnik s proširenom i neproširenom osnovom. Ponekad se relativno podjednako koriste riječi sa proširenom i neproširenom osnovom, ali u nekim je situacijama češće korištenje proširene osnove u množini negoli je korištenje kratke množine. Primjer je množina riječi *sin* gdje je duga množina *sinovi*, a kratka *sini*. Naravno, rječnik mora prepoznati dugu množinu, ali netko će možda pokušati koristiti i kratku množinu. Iako kratka nije pogrešna, previše je stilski obilježena i nedovoljno frekventna da bi se ti oblici unosili u rječnik. Ponekad će rječnik prihvatiti kratak oblik, ponekad neće, ali prihvatit će samo kada množina od *sin* poklapa

(slučajno) s nekom drugom riječi: *sini* (od gl. *sinuti*). Korisnike to može zbunjivati i neće svatko pravilno povezati da se zapravo radi o drugoj riječi. Ponekad će se netko pitati zašto u rječniku postoji ovaj oblik ove riječi, ali ne postoji onaj oblik iste riječi i prijavljivat će to kao nedostatak rječnika.

Vlastita imena i prezimena ne treba unositi u rječnik. Treba samo imena i prezimena važnih osoba u hrvatskoj povijesti, ali inače ne treba popisivati sva imena iz rječnika u rječnik za računalnu provjeru pravopisa jer prevelik broj imena i prezimena ima velik broj inačica, a onda bi se trebalo dodavati sve te inačice u rječnik što bi opterećivalo i povećavalo rječnik. Imena i prezimena bi moglo biti gotovo koliko i svih drugih riječi. Potrebno je isto tako unijeti imena važnijih umjetnika, političara, sportaša, glumaca i pjevača. Ova formulacija nije precizna, ali i ne može biti. U rječnik je važno unijeti imena koja se u određenom području ili struci često koriste ili se autori često citiraju. Važno je imati ona imena koja se koriste u tekstu i nije moguće od rječnika za provjeru pravopisa skupljati imena enciklopedijski. Neizbježno je da nečije strano ime kolidira s nekom riječi u hrvatskome jeziku što će, ako ništa drugo, dozvoliti pogrešku u pisanju velikoga i maloga slova. Prevelik broj imena može rezultirati i time da korisnik pogrešno napiše ime jedne osobe, ali da isto tako to ime postoji u rječniku, ali se zapravo veže uz drugo prezime. Takva situacija može nekoga prevariti da „ispravi“ u pogrešno zapravo po nagovoru rječnika. Iako rječnik nije kriv za nečije nedostatno znanje, treba pokušavati izbjegavati takve situacije. Zato mislim da nije dobro unositi imena svih nobelovaca iz svih polja jer bi to nepotrebno opteretilo rječnik. Puno sličnih imena znači i puno prostora za zabunu. Onaj tko želi napisati ime nekoga znanstvenika ili književnika, moći će, ali će računalna provjera pravopisa u svakom slučaju podcrtati to ime što će korisniku privući pažnju da još jednom provjeri je li sve napisano kako treba. Prihvati li rječnik ime zato što postoji na popisu riječi, a korisnik zapravo nije htio napisati to ime – neće se podcrtati ništa i veća je vjerojatnost, pogotovo kada se radi o korisnicima koji se previše oslanjanju na ovakve alate za pomoć pri pisanju, da će pogrešno ime ostati neispravljeno. Treba izabrati ona imena koju su poznatija i koja se spominju u pisanju novinarskih i znanstveno-popularnih tekstova. Recimo dva-tri imena koja se ne može zaobići kada se piše o području čijem su razvoju doprinjeli. Isto tako treba voditi i računa o stvaranju pridjeva od tih imena jer je posebna riječ pridjev pisan po izvornome pisanju (Shakespeareov, shakespearski) i posebno riječ pisana prilagođeno (šekspirski). Treba unositi oba oblika kao zasebne unose ako su oba oblika u upotrebi.

Problem je u tome što se imena u rječnik ne mogu unositi kao cjeline, samo kao zasebne riječi pa je onda potrebno kao tri riječi unositi ime [1] *Wilhelm* [2] *von* [3] *Humboldt*, a budući da je riječi u rječniku potrebno razvrstati abecednim redom, često je teško i onome tko održava rječnik uvijek prepoznati i sjetiti se zbog čega je točno riječ *von* u rječniku. Skloniji sam takve riječi ne uključivati u rječnik negoli ih uključivati jer je manja šteta riječ ne imati u rječniku pa da ju sustav podcrta negoli imati je u rječniku pa da zbog nepodcrtavanja pogreška ostane neispravljena. Do neke se mjere ovaj problem može riješiti provjerom gramatike jer je tim alatima moguće definirati pravila za odnose između riječi, ali i opet samo na formalnoj razini, bez ikakva razumijevanja konteksta ili značenja. Međutim, takvi alati omogućuju ispravke poput *Gorski Kotar* u *Gorski kotar*. Računalna provjera pravopisa mora dopuštati i *kotar* i *Kotar* jer se ta riječ može naći i na početku rečenice, a onda je nužno da bude napisana velikim početnim slovom.

4.3. Primjer izrade pravila za imenice a-sklonidbe

Imenice su riječi koje karakteriziraju gramatičke kategorije roda, broja i padeža, a s obzirom na rod dijele se na imenice muškoga, ženskoga i srednjega roda. Razlike se među njima uspostavljaju gramatičkim morfemima. (Silić – Pranjković 2007: 97) Imenicama je svojstvena predmetnost, opredmećeno svojstvo ili opredmećeni proces, odnosno imenicama se imanuju pojave vanjskoga svijeta i ljudskoga doživljaja svijeta: osobe, mjesta, stvari... (Marković 2012: 231) (Barić i sur. 2005: 100)

Imenice su a-sklonidbe imenice koje u genitivu jednine imaju gramatički morfem -a i nisu hipokoristici. (Barić *et al*. 2005: 104) Značajke su a-sklonidbe u jednini nepostojano *a* u nominativu jednine, a ono se *a* se najčešće ne gubi u genitivu množine. Akuzativ je jednine jednak genitivu ako se imenica odnosi na što živo, a nominativu ako se odnosi na što neživo. Isto vrijedi i ako se imenice odnose na skupove živih bića i biljaka, kada imenice za živo stoje za neživo. U vokativu je najčešći nastavak *-e*, ali ako osnova završava na nepčanik, nastavak je *-u*. Taj nastavak imaju i imenice na -ic, -č(a)c, -č(a)k, -ć(a)k, -d(a)k, -dž(a)k, -đ(a)k, -t(a)k, -z(a)g, etnici i toponimi na -ez, imenice na -k, -g, -h, -c, imenica na -ar, -ir, ostale na -r imaju nastavak -e. Značajka je dativa i lokativa da imaju isti nastavak: -u, a u imenica se na -k, -g, -h provodi palatalizacija. Nastavak je za instrumental -om, a imenice na nepčanik imaju -em, kao i imenica na -(a)c. Imenice na -ar u instrumentalu mogu imati i -om i -em, tako i imenica *put*,

a pravilno bi bilo u imenicama s osnovama na -s i -z koristiti nastavak -om. (Težak – Babić 2009: 99 – 102)

N	ø	ø	ø	ø	ø	ø
G	a	a	a	a	a	a
D	u	u	u	u	u	u
A	a	ø	ø	a	a	ø
V	e	e	u	e	e	e
L	u	u	u	u	u	u
I	om	om	em	om	em	om

Primjer 11: Nastavci jednine imenica a-sklonidbe (Težak Babić 2009: 99 – 102)

U Primjeru 11 navedeni su nastavci za jedninu imenica muškoga roda, odnosno imenica a-sklonidbe. Važno je napomenuti da je iz primjera vidljivo kako se dijelovi deklinacijskoga obrasca preklapaju, a dijelovi se razlikuju. U primjeru nisu davane cijele riječi nego samo nastavci radi lakše usporedbe (nastavak do nastavka).

Međutim, osim navedenih, moguće je u nominativu jednine imati i gramatički morfem -e kao u imenici *Hrvoje*. U deklinaciji se -e mijenja dalje po petom stupcu nastavaka. (Silić – Pranjković 2007: 97)

Prvo će biti važno izdvojiti one dijelove koji se mogu grupirati, odnosno napraviti posebne klase za dijelove deklinacije koji se mijenjaju. Naime, već je rečeno kako Hunspell dozvoljava dodavanje dvaju nastavka na kraj riječi, to je potrebno iskoristiti za: dodavanje gramatičkih morfema na osnovu (drugi, zadnji nastavak), ali i za rješavanje glasovnih promjena između dodavanja osnove i nastavka. Recimo da se deklinacijski obrazac daje za imenicu *muškarac* (samo jednina), onda je u nominativu nulti nastavak i ostaje nepostojano *a*, ali je u genitivu nastavak *a*, ali se gubi nepostojano *a* iz nastavka *-(a)c*. Tako bi se navedeno moglo postaviti ovako:

```
SFX AA Y 1
SFX AA 0 0 .+Njdmr

SFX AB Y 1
SFX AB 0 a .+Gjdmr
```

```
SFX AC Y 2
SFX AC 0 0/AA ac
SFX AC ac c/AB ac
```

Primjer 12: Klasa AA: gram. morfem za Njd, AB: gram. morfem za Gjd, AC: korištenje gramatičkih morfema s obzirom na glasovne promjene

Primjer 12 pokazuje sljedeće: klasa AA sadrži nulti morfem, nominativ jednine muškoga roda. U ovom dijelu ne treba pisati da se radi o *imenicama* jer je ta oznaka u rječniku, ne u datoteci s gramatičkim nastavcima. Uglavnom, posebno se mora navoditi nominativ, a posebno genitiv zato što je u klasi AC potrebno napraviti tako da u nominativu ne dolazi do glasovne promjene, pa se dodaje nulti morfem – ustvari samo zbog oznake Njdmr radi morfološke analize, a u genitivu je potrebno ukloniti ac (muškarac), umjesto toga dodati na osnovu samo c (*muškarc) kako bi se na kraju mogao dodati gramatički morfem -a (muškarca).

Zbog toga što nije teško predvidjeti na kojim se sve mjestima kroz deklinacijsku paradigmu ovoga tipa imenica javljaju promjene – u *muškarac* razlika je u nominativu naspram ostalih padeža, a u riječi je *stric* razlika samo u vokativu – potrebno je sve nastavke iz primjera 11 unijeti kao posebne klase, odnosno moguća su manja grupiranja, recimo nastavak -a u genitivu jednine te nastavak -u u dativu i lokativu jednine:

```
# nominativ jednine imenica muškoga roda
SFX AA Y 1
SFX AA 0 0 .      +Njdmr

# genitiv, dativ i lokativ jednine imenica muškoga roda
SFX AB Y 2
SFX AB 0 a .      +Gjdmr
SFX AB 0 u .      +DLjdmr

# akuzativ jednine imenica muškoga roda za živo
SFX AC Y 1
SFX AC 0 a .      +Ajdmr

# akuzativ jednine imenica muškoga roda za neživo
```

```
SFX AD Y 1
SFX AD 0 0 .      +Ajdmr

# vokativ na -e jednine imenica muškoga roda
SFX AE Y 1
SFX AE 0 e .      +Vjdmr

# vokativ na -u jednine imenica muškoga roda
SFX AF Y 1
SFX AF 0 u .      +Vjdmr

# vokativ jednak nominativu imenica muškoga roda (Marko)
SFX AG Y 1
SFX AG 0 0 .      Vjdmr

# instrumental na -om jednine imenica muškoga roda
SFX AH Y 1
SFX AH 0 om .          +Ijdmr

# instrumental na -em jednine imenica muškoga roda
SFX AI Y 1
SFX AI 0 em .          +Ijdmr
```

Primjer 13: Prijedlog gramatičkih morfema jednine imenica a-deklinacije

Nominativ mora stajati posebno zbog nepostojanoga -a koje se gubi u drugim padežima. Akuzativ se dodaje ovisno o tome radi li se o imenici za živo (-a) ili za neživo (-ø), ali tu može biti problema kada se radi o imenicama koje se pojavljuju i kao živo i kao neživo: *Ima raka ~ Ima rak*. Zbog toga *rak* treba imati dva unosa, jednom sa deklinacijskom paradigmom za živo, drugi puta sa deklinacijskom paradigmom za neživo.

Ne radi se ni samo o glasovnim promjenama, i za imenice tipa *Marko* potrebno je ukloniti -o radi dodavanja nastavka -a u genitivu jednine. Tako će za tu imenicu biti potrebna sljedeća klasa:

```
SFX AJ Y 2
SFX AJ 0 0/AAAG .
```

```
SFX AJ o 0/ABACAH o
```

Primjer 14: deklinacijski obrazac za riječ Marko

Tako se na imenicu *Marko* dodaje nulti morfem u nominativu i vokativu jednine. U ostalim je padežima potrebno ukloniti *-o* radi dodavanja drugih gramatičkih morfema: *Mark*/o/a/u/om. Možda se u instrumentalu moglo *-o* i ostaviti pa dodati samo *-m,* ali to bi kompliciralo situaciju za druge riječi, za riječi koje ne završavaju na *-o,* a u instrumentalu imaju nastavak -om. Klasa kao što je AJ treba napraviti tako za svaku riječ u rječniku a-sklonidbe. Kada se misli na *rak* kao životinju onda treba dodati klasu AC radi morfološke analize da Hunspell ispiše da je oblik *raka* oblik koji postoji u genitivu i akuzativu jednine. Naravno, ispisat će i da je *rak* i nominativ i akuzativ jednine jer *rak* može biti i bolest (neživo) pa je nastavak u akuzativu nulti, ali je važna razlika ta što *rak* kada označava bolest najčešće nema množine. U slučaju kada ta riječ označava životinju, potrebno je napraviti klasu, po mogućnosti u sklopu klase AJ i množinu, a u riječi *rak* u rječniku koja je rezervirana za bolest, treba napraviti klasu koja neće sadržavati oblike za množine.

Možda se sada takav posao može činiti uzaludnim jer unutar Hunspella praktično nema razlikovanja, ali ako bi se takav princip doslovno provodio kroz rječnik, klase bi se mogle koristiti kao oznake (engl. *tag*) za identificiranje određenih oblika riječi što bi moglo biti od velike koristi u izradi računalne provjere gramatike. Informacije bi iz takvoga rječnika poslužile za polazna točka, materijal, za druge alate.[41]

U primjeru 11 su navedene moguće kombinacije gramatičkih morfema, ali tomu još treba dodati u obzir kategorije živo ~ neživo i sve ostale glasovne promjene: zbog toga se jedan od navedenih deklinacijskih obrazaca može pojavljivati i više puta. Recimo deklinacija riječi *čovjek* naspram deklinacije riječi *vitez* jer u vokativu u prvoj dolazi *č* od *k,* a u drugoj dolazi u tom padežu dolazi *ž* od *z* pa se mogu pretpostaviti sljedeće klase:

```
SFX AK Y 2
SFX AK 0/AAABACAH .
SFX AK k č/AE .

SFX AL Y 2
SFX AL 0 /AAABACAH .
```

41 http://wiki.languagetool.org/tips-and-tricks; pristupljeno 1. prosinca 2017.

```
SFX AL z ž/AE .
```

Primjer 15: prikaz deklinacijskih obrazaca kednine za riječi čovjek i vitez

Međutim, ni tu nije kraj. Primjer pokazuje samo jedninu, ali treba voditi računa o tome da, recimo za klasu AK, postoje riječi koje imaju i produljenu množinu. Postoje riječi s tim obrascem i za živo i za neživo pa prema tome treba i napraviti dodatne klase. Čini se kao mnogo i jest mnogo: što sa imenicama koje imaju samo jedninu ili samo množinu? I za njih treba raditi posebne klase. Tako i za riječi sa supletivnim osnovama. Posla je puno, a proces je dodavanja klasa spor. Postoji i mogućnost pogreške u dodavanju klasa, ako se radi ručno, ali i postoji mogućnost da se neka klasa preskoči, da se ne napravi, a onda je potrebno uređivati datoteku s klasama, ali i ispravljati sve riječi u rječničkoj datoteci. Da se neka situacija preskoči može se dogoditi ako je broj riječi premalen, a ako je pak prevelik – lako se pogriješi, puno ima ponavljanja.

Možda postoji način i da se ukloni mogućnost pogreške, ali svejedno se radi o velikom poslu koji traje i traje. Zbog nedostatka vremena, nisam bio u prilici pokušati automatizirati izradu deklinacijskih klasa i obrazaca jer bi predugo trajalo, a pokušavati na svega nekoliko riječi nema smisla. Dakle, ako bi se u tekstnu datoteku unosile riječi tako da prvo slijedi jedinstveni brojčani identifikator, odijeli se tabulatorom pa se unese osnova riječi, zatim se i ona odijeli tabulatorom te se, ako ima potrebe, unese dio riječi koji treba ukloniti radi glasovne promjene, a iza njega zatim slijedi promjena te na kraju gramatički morfem:

```
1 | čovjek 0 0 | Njd
1 | čovjek 0 a | GAjd
1 | čovjek 0 u | DLjd
1 | čovjek k če | Vjd
1 | čovjek 0 om | Ijd
1 | čovjek čovjek ljudi | NGVmn
1 | čovjek čovjek ljude | Amn
1 | čovjek čovjek ljudima | DLImn
```

Primjer 16: prijedlog izrade klasā računalom radi smanjenja mogućnosti pogreške

Ovakvo raspisivanje nema veze sa Hunspellom niti je dio rječnika za računalnu provjeru pravopisa, ali bi se na ovaj način mogla eliminirati mogućnost pogreške ili

izostavljanja klase. Nedostatak je ovoga pristupa što bi se sve riječi iz rječnika i opet trebale ovako raspisati ručno što i opet znači da je potrebno utrošiti puno vremena. Ne znam postoji li kakav izvor iz kojega bi se riječi računalom mogle oblikovati na prikazan način.

Računalna bi skripta mogla čitati datoteku redak po redak, razlikujući homonime prema brojčanoj oznaci – tako brojem 2 *rak* kao bolest i brojem 3 *rak* kao životinja – za svaku riječ pripremiti višedimenzionalno polje (engl. m*ultidimensional arrays*) ili nekako drugačije i spremati informacije o svakom retku konkretne riječi. Recimo za riječ *čovjek* ne treba uklanjati slova iz kanonskoga oblika, ne treba dodavati slova na kanonski oblik, da je, za prvi redak, gramatički morfem nulti te da radi morfološke analize treba dodati oznaku `+Njdmr`.

Skripta bi skupljala informacije riječi pod 1 sve dok je sljedeći redak isto obrojčan brojkom 1. Ako je sljedeći redak broj 2, skripta bi spremila informacije pod jedan u neki od oblika pogodnih za lagan pristup (JSON, višedimenzionalno polje) i počela prikupljati informacije pod brojem dva. Nakon prikupljanja, skripta bi usporedila prikupljeno s onime što je već pohranjeno u formatu JSON ili je u višedimenzionalnome polju i napravilo jedno od sljedećega: ako točno takav obrazac već ne postoji, skripta bi dodala i to na popis, ako takav obrazac već postoji, pod brojem koji označava riječ, skripta bi samo dodala i oznaku obrasca koji se nalazi u polju. Oznaka bi se u višedimenzionalnome polju poklapala sa brojčanom oznakom riječi čime bi bilo moguće automatski dodati odgovarajuću klasu na odgovarajuću riječ. Ponovno, problem je samo oblikovati riječi na predložen način. Naravno, informatičar bi možda našao i bolji i efikasniji način za analizu, recimo pomoću baze podataka i slično.

Za potrebe sam ovoga rada napravio skriptu koja može upravo to, koristi tekst oblikovan kao u primjeru 16 te na temelju toga radi rječničku datoteku i datoteku a afiksima. Skripti se može pristupiti na adresi `https://github.com/krunose/hr-hunspell/tree/master/tools/dpl/bft.php`, ali treba voditi računa da skripta ima brojne nedostatke i napravljena je samo kao potvrda (dokaz) koncepcije i ideje. Datoteka `bft.php` ne razlikuje radi li se o prefiksu ili sufiksu, nije moguće istovremeno dodavati prefikse i sufikse što bi mogao biti nedostatak prilikom komparacije pridjeva jer je na komparativ potrebno dodati i sufiks i afiks istovremeno.

Skripta radi na sljedeći način: sve unose označene istim brojem, skripta tretira kao jednu riječ. Brojevi moraju biti dodati radi razlikovanja homonima i radi tretiranja riječi sa supletivnim osnovama kao jedne riječi jer bi, ako bi se samo pisalo *čovjek*, bilo teško prepoznati da je *ljudi* zapravo množina iste riječi. Računala su dobra u automatizaciji jednostavnih zadataka, ali ne mogu se nositi sa značenjem. Drugo je polje zapravo osnova

riječi s koje onda skripta uklanja znakove navedene u sljedećemu polju, a dodaje skup znakova iz četvrtoga polja. Valja primijetiti da za množinu riječi *čovjek* to znači uklanjanje cijele osnove (*čovjek*) i dodavanje cijele riječi (*ljudi*) kao nastavka iako je to gramatički neopravdano. Moguće je dodavati i nova polja koja bi se mogla koristiti za davanje novih funkcija, ali onda to znači da treba uređivati i skriptu. Sada nije moguće automatski izraditi valjani rječnik, ali je moguće izlistati riječi i izlistati (samo) sufikse koje te riječi imaju. To se onda može iskoristiti za izradu rječnika koji može služiti za provjeru pravopisa: dodati broj riječi na početak rječničke datoteke i dodati ono što je potrebno na početak datoteke s afiksima (enkodiranje teksta i ostalo). Ova skripta ne koristi naprednije značajke Hunspella, a većina ih i nije opisana u ovome radu jer bi to širilo rad i izašlo bi se iz okvira predviđenih za izradu diplomskoga rada.

Za funkcioniranje je skripte potrebna datoteka u kojoj će biti raspisani sufiksi riječi (recimo `wordlist`), zatim je potrebna računalna skripta (`bft.php`) koja će koristiti datoteku s riječima za stvaranje datoteka `test-hr_HR.dic` te `test-hr_HR.aff`. Rezultatima se obradbe namjerno domeće *test* da bi ih se razlikovalo od originalnoga rječnika, da ne dođe do zabune i da se valjani rječnik ne zamijeni datotekama koje zapravo samo služe kao potvrda koncepta. Naravno, na ovome je moguće graditi dalje i napraviti iz toga koristan alat koji bi smanjio mogućnost griješenja koja je prisutan prilikom ručne izrade i dodavanje pravila. Prednost je i to što se ne kreće od pravila prema riječima nego obrnuto. U tom se slučaju rade samo pravila koja su doista i potrebna, a skripta ne udvostručuje pravila nego koristi prethodne informacije i dodjeljuje i stvara samo klase koje se još nisu pojavile u datoteci s popisima riječi. Nažalost, u ovom pristupu je potrebno nebrojeno puta ponavljati isto u datoteci iz koje bi skripta napravila pravila i zbog toga bi ovaj način izrade rječnika bio dugotrajan i spor, ali zato i razmjerno jednostavniji jer bi dodavanje novih riječi i ispravljanje pogrešaka bilo jednostavnije: lakše je raspisati nekoliko oblika za novu riječ negoli tražiti koja bi klasa odgovarala toj riječi tako da se zadovolje svi kriteriji. Možda se ne čini tako, ali tražiti je odgovarajuću klasu u datoteci s afiksima vrlo spor i zahtjevan posao. Skripta u sadašnjem obliku može dodavati samo jedan sufiks pa nije moguće iskoristiti prednosti dodavanja dvaju sufiksa. I to je moguće riješiti računalnom skriptom, ali to bi zahtijevalo puno veći angažman i svakako izlazi iz okvira ovoga rada. Takva bi poboljšana skripta mogla nakon brisanja *čovjek* kao poseban sufiks dodati *ljud* i kao poseban sufiks dodati morfem *i* (*ljudi*) čime bi se promjena *čovjek* > *ljud* zapravo tretirala kao glasovna promjena, a drugi bi sufiks (*-i*) imao ulogu gramatičkoga morfema što bi značilo da je i u toj situaciji moguće razviti alat koji i

imenicu *čovjek* svrstao u skupinu imenica a-sklonidbe s promjenom *k* u *č* u vokativu jednine; sada će se sve imenice sa supletivnim osnovama razlikovati od drugih imenica istoga tipa koje nemaju supletivne osnove.

Za potrebe sam ovoga rada iz rječnika sa `https://github.com/krunose/hr-hunspell` izvukao imenice sa željom da u radu predstavim sva pravila za deklinaciju imenica u hrvatskome jeziku za korištenje u računalnoj provjeri pravopisa Hunspellom. Međutim, opseg je imenica bio prevelik, a pristupiti mu se može preko internetske adrese

`https://github.com/krunose/hr-hunspell/tools/imenice.txt`

ali radi se o sveukupno dvadeset tisuća riječi što je previše riječi za potrebe jednoga diplomskoga rada. Iskustvo mi govori da posao uz korištenje dvaju sufiksa bez pomoći računalnih skripti ne bi mogao biti završen ni unutar godine dana – i to samo za imenice. Budući da sadašnji rječnik nije rađen u skladu s onime što se ovdje predlaže, često su glagolske imenice (*pisanje*) rađene od glagola (*pisati*), ali mislim da je bolje ne uvoditi tvorbu u rječnik jer bi, provodi li se sve dosljedno, od jednoga glagola bilo potrebno raditi i glagolsku imenicu i imenicu za pripadnika muškoga spola i pripadnicu ženskoga spola i odnosni pridjev i posvojni pridjev i glagolska imenica... *učiti* → *učenje* → *učen* → *učitelj* → *učiteljica* → *učeći*, pa onda još i prefiksacijom: *naučiti, podučiti, poučiti, izučiti*...

Iz konteksta elastičnoga pretraživanja ima smisla grupirati glagole s obzirom na prefikse, to isto ulazi u tvorbu, ali to ujedno i otežava izradu rječnika bez pomoći računalnih skripti i povećava broj pravila (klasa) u tolikoj mjeri da i nisam siguran da korist prelazi uložen trud. Raditi bi prefiksaciju, ne samo glagola nego i imenica, imalo opravdanja kada bi postojao dobro isfiltriran korpus riječi i kada bi postojali alati (računalne skripte) koji bi iz toga korpusa mogle napraviti funkcionalni rječnik. Kao i u mnogim drugim stvarima, potrebno je napraviti kompromis između broja klasa i elegancije kojom deklinacijska pravila može napraviti čovjek i između efikasnosti i brzine kojom to može računalo, pa makar bilo i određenih ponavljanja. Međutim, brzina kojom računalo obrađuje veliku količinu podataka, moglo bi omogućiti i rješavanje prefiksacije u glagola i imenica, ali kako je to izvan okvira ovoga diplomskoga rada, reći ću samo da je potrebno iz *učitelj* tvorbom raditi riječi mocijske parnjake (*učiteljica*). Isto tako se iz glagola (*učiti*) treba tvoriti radni i trpni glagolski pridjevi (*učio*, *učen*) te glagolski prilozi (*učeći*). Iz konteksta je elastičnoga pretraživanja logičnije u datoteku nastavaka unositi i tvorbene nastavke (od jedne riječi tvoriti više), ali je iz konteksta računalne provjere pravopisa bolje tvorbu isključivati iz datoteke nastavaka. Međutim, ElasticSearch koristi rječnik koji koristi i Hunspell pa je za sada potrebno napraviti

kompromis, ali najbolje bi bilo da se za svaki alat napravi poseban rječnik. Pridjevi bi isto trebali biti zasebni unosi u rječnicima (*učiteljski*), ali naravno, svi rodovi bi trebali biti napravljeni iz toga jednoga unosa (*učiteljska, učiteljsko*) bez obzira na ElasticSearch i bez obzira na Hunspell.

Da se vratim na izvučene riječi za potrebe ovoga rada: ustanovivši da se ne može obraditi dvadesetak tisuća riječi pruženome vremenskome roku, odlučio sam obraditi samo imenice muškoga roda. Međutim, i tih je riječi bilo još uvijek previše pa sam odlučio napraviti sljedeće: izvući riječi koje se u gramatikama daju kao primjeri različitih tipova deklinacijskih obrazaca u imenica a-sklonidbe, odnosno želio sam dobiti što je moguće više varijacija u deklinacijskome obrascu imenica a-sklonidbe. Naravno, nisam mogao obraditi ni sve imenice a-sklonidbe jer računalna skripta zahtijeva da se svaka riječ raspiše u svim padežima i oba broja ručno, a kad se broj oblika pomnoži sa brojem riječi, jasno je zašto takav posao nije bilo moguće odraditi za potrebe diplomskoga rada.

Umjesto toga sam izvukao sto devedeset i pet imenica koje se u gramatikama navode kao primjeri različitih situacija u deklinacijskome obrascu imenica a-sklonidbe muškoga roda. Od toga broja riječi nastale su ukupno sto trideset i dvije klase. Znači radi se samo o imenicama a-sklonidbe i već se tu vidi koliko je klasa potrebno da se taj tip imenica opiše u kontekstu izrade rječnika za provjeru pravopisa Hunspellom. Naravno, riječi su i birane da daju što više klasa pa je za pretpostaviti da se dodavanjem drugih riječi a-sklonidbe broj klasa ne bi puno mijenjao, pa bi se razlika između broja riječi i broja klasa samo povećavala, što i jest poželjno – važno je sa što manje klasa opisati što veći broj imenica. Čini mi se čak da ne postoji prepreka da se jednostavno ručno odredi korpus riječi, a da se iskoriste računalne skripte kako bi se pojednostavilo opisivanje tih riječi. Recimo, moguće je izraditi alat koji bi, za svaku riječ koja je ručno dodana na popis i koju je provjerio i odobrio čovjek, pretraživao Hrvatski jezični portal[42] (HJP), izvlačio deklinacijske obrasce, a rezultate oblikovao tako da budu upotrebljivi za izradu rječnika za provjeru pravopisa Husnpellom. Alati koji mogu razumjeti rezultate na HJP-u (engl. *parse*) već postoje.[43] Takav je alat dostupan i moguće je čitati njegov izvorni kôd.[44] Ipak, automatizacija takva procesa nije bez prepreka jer bi trebalo konzultirati vlasnike Hrvatskoga jezičnoga portala. Ako bi računalna skripta poslala HJP-u pedesetak tisuća zahtjeva pa slobodno mogu reći odjednom, pitanje je bi li poslužitelj to

42 http://hjp.znanje.hr/; pristupljeno 11 prosinca 2017.

43 https://play.google.com/store/apps/details?id=com.dekoraktiv.android.rsr; pristupljeno 11. prosinca 2017.

44 https://github.com/akoncic/rjecnikstranihrijeci-android; pristupljeno 11. prosinca 2017. (Čini mi se da je naziv alata netočan jer se ne radi o rječniku stranih riječi već o alatu koji upit šalje na Hrvatski jezični portal i prilagođava rezultat pretraživanja prikazu na uređaju koji pokreće operacijski sustav Android).

izdržao. Isto je tako i pitanje je li takvo korištenje stranice legalno, odnosno u skladu s pravilima po kojima je stranica ustupljena na korištenje krajnjem korisniku.

Ako bi se već i išlo na automatizaciju te razine i složenosti, možda bi bolje bilo koristiti neki tekstni format, možda XML, ili bazu podataka (SQLite) za izradu višenamjenskoga rječnika. Umjesto jednostavnoga oblika prikazanoga primjerom 16, moguće je napraviti rječnik koji će nositi sve informacije o gramatičkim svojstvima riječi, koji će nositi i značenje riječi, možda i oznaku pripadnosti riječi standardnome jeziku, ali i oznaku treba li konkretnu riječ koristiti u izradi računalne provjere pravopisa Hunspellom ili ne. Riječ koja je označena, bit će korištena u automatiziranju kompiliranju rječnika, ako riječ nije označena, računalni bi je program preskočio. Takav bi format čuvanja i skupljanja riječi bio koristan ne samo za izradu rječnika za provjeru pravopisa, već bi omogućio i pretraživanje riječi prema drugim parametrima, recimo s obzirom na vremensku, funkcionalnu ili stilsku raslojenost: *izvuci sve knjiške riječi koje počinju slovom a*.

5. Zaključak

Računalna provjera pravopisa zapravo ne provjerava pravopis, ne provjerava sve ono što je napisano u pravopisnim priručnicima. Zadaća je računalne provjere pravopisa provjeriti postoji li riječ napisana u dokumentu na popisu dozvoljenih riječi. Nema provjere sastavljenoga i nesastavljenoga pisanja, nema provjeravanja pisanja interpunkcijskih i pravopisnih znakova, provjera je pisanja velikoga i maloga slova također u nekim segmentima krnja. Računalna provjera pravopisa zapravo samo provjerava postoji li riječ napisana u dokumentu na popisu dozvoljenih riječi, ali svaka se riječ provjerava posebno, za sebe, izvan svakoga konteksta što znači da će računalna provjera pravopisa propustiti mnoge pogreške. Ipak, računalna je provjera pravopisa samo alat koji bi trebao pomoći korisniku prilikom pisanja, alat koji pomaže ne može pisati i uređivati tekst umjesto korisnika. Lakše je naučiti pravilno i dobro pisati negoli napraviti alat koji će pomoći ispraviti sve pogreške koje čovjek može napraviti. Način ispravljanja ne ovisi samo o kvaliteti alata nego i o kontekstu u kojemu se tekst piše. Tehnički dokument (tehničke upute, priručnik, upute za upotrebu) i znanstveni ili stručni rad u području računalne tehnologije bitno su drugačiji tekst od književnoga djela: upotrebljavaju se druge riječi, drugačija je sintaksa, drugačije su pogreške, drugačiji je raspon pravopisnih i interpunkcijskih znakova, drugačija je upotreba znakova i ne postoji računalni program/sustav koji bi mogao sa stopostotnom uspješnošću ispravljati sve tekstove. Pogrešno je pretpostavljati da bi program za računalnu provjeru pravopisa trebao ispravljati sve moguće pogreške jer, banalizirajući, može se onda tražiti da program za računalnu provjeru pravopisa od dadaističkoga teksta radi dobro strukturiran i koherentan tekst, a jasno je zašto to nije smisleno. Programi za računalnu provjeru pravopisa i gramatike ne rješavaju korisnika odgovornosti za ispravnost teksta. Program za računalnu provjeru pravopisa može samo olakšati pregledavanje teksta ispravljanjem očitih pogreški još za vrijeme nastajanja teksta. Mislim da bi čak nekoliko sati hrvatskoga jezika u završnim razredima srednje škole (seminarski radovi, završni rad, priprema za pisanje seminarskih radova na fakultetima) trebalo iskoristiti na pokazivanje u čemu je računalna provjera pravopisa nedostatna kako bi se korisnici mogli bolje ispravljati tekst.

Što je alat za računalnu provjeru pravopisa složeniji, teže ga je integrirati u druge programe kao što su obrađivači teksta (engl. *word processor*), zahtijevaju više resursa i izrada je pravila za takve sustave složenija. Hunspell je računalni program otvorena koda što znači da ga može koristiti svatko u bilo koje svrhe i na bilo koji način. Zreo je to program čije

značajke omogućuju izradu pravila za računalnu provjeru pravopisa za gotovo svaki tip jezika. Značajke koje nedostaju može dodati svatko s dovoljno znanja i vještina. Hunspell je *de facto* standard govoreći o alatima otvorena koda. Uostalom, računalna se provjera pravopisa može upariti s računalnom provjerom pravopisa što će, ako se pažljivo sastave pravila, omogućiti dobru provjeru i svakako će smanjiti broj pogrešaka. U tome je kontekstu bolje ne uključiti riječ na popis negoli je uključiti. Hrvatski je jezik bogat gramatičkim oblicima i razumljivo je da će u takvome jeziku doći do kolizije oblika, da je upotrijebljena riječ izgledom jednaka obliku koji je moguć, ali je u konkretnoj situaciji pogrešna.

U izradi rječnika za računalnu provjeru pravopisa nije uvijek moguće vjerno pratiti gramatičke opise nekoga jezika u gramatikama i pravopisima, sami alati daju neka ograničenja, a ograničenja predstavljaju i sami pravopisni problemi koje ljudi, koji znaju pisati, rješavaju relativno jednostavno, koristeći razumijevanje i jezika i izvanjezične stvarnosti, računala su zakinuta za tu sposobnost pa je važno ispravljati ono što se može i koliko se može, a o ostalome treba učiti korisnike kako bi se računalnom provjerom pravopisa mogli služiti što bolje i efikasnije.

Nedostaci Hunspella izneseni u ovome radu ne trebaju biti razlog da se Hunspell za računalnu provjeru pravopisa ne koristi, upravo suprotno: pravila su za računalnu provjeru pravopisa Hunspellom javno dostupna i svatko može ispraviti nedostatak u pravilima ili upozoriti na nedostatak koji se, možda, ne može ispraviti zbog nedostataka samoga programa, ali se problem može dokumentirati pa onda ili riješiti kojim drugim alatom ili pak jednostavno upozoravanjem korisnika.

Pogrešno je na rječnik za računalnu provjeru pravopisa gledati kao na skup svih, zapravo što većega broja riječi. Rječnik koji sadrži pravila za računalnu provjeru pravopisa zapravo je rječnik s određenom namjenom i svrhom. Kao što se u pravopisnome rječniku neće tražiti značenje riječi i kao što se u tome rječniku neće tražiti riječ s kojom nije povezan neki pravopisni problem, tako se u rječniku za računalnu provjeru pravopisa trebaju nalaziti samo one riječi koje su nužne za kvalitetnu provjeru pravopisa. Višak riječi ne pomaže nego odmaže stvarajući kolizije. Uključe li se sve riječi u rječnik za izradu pravila računalne provjere pravopisa, dogodit će se to da velik broj pogrešaka neće biti prepoznat.

Oni koji rade pravila za računalnu provjeru pravopisa trebaju se osloboditi straha od crvene valovite crte jer podcrtavanje koje riječi ne znači automatski da tu riječ u rječnik i treba uključiti. Treba biti selektivan, rječnik za računalnu provjeru pravopisa ne može sadržavati sve. Izbor riječi u računalnoj provjeri pravopisa zapravo utječe na kvalitetu

provjere. Ovaj je rječnik rječnik sa svrhom. Treba uključivati riječi koje pripadaju općem leksiku hrvatskoga standardnoga jezika, koncentrirati se na riječi koje se nalaze u novinarskim i sličnim publicističkim tekstovima, Wikipedia je zapravo dobar orijentir jer obrađuje različite teme, ali uglavnom na osnovnoj razini, bez korištenja uskostručnih termina koje ne bi razumjeli obrazovani govornici hrvatskoga standardnoga jezika ako nisu dijelom te struke. Osnovni rječnik ne treba, po mome mišljenju, prelaziti četrdesetak tisuća riječi, ali bi svaka struka, znanstvena disciplina trebala imati svoj, specijalistički rječnik koji bi se koristio uz osnovni, opći rječnik. Riječi se u tim rječnicima ne bi trebale preklapati. Ako je termin jedne struke ujedno i dio općega leksika i nalazi se u općemu, temeljnome rječniku, specijalistički rječnik treba raditi imajući to na umu.

Sâm rječnik nije jednostavno sastaviti, onaj tko sastavlja često mora donositi odluke koje nisu poštene ni prema onome tko sastavlja ni prema korisnicima rječnika, ali ipak su nužne. Postoji cijeli niz riječi koje ne djeluju i ne izgledaju kao uskostručni termini, ali mi se čini da se redovito koriste u samo nekim prilikama. Takve su riječi često imena: *LibreOffice, Hunspell*, ili pak opće riječi; izvan konteksta računalne provjere pravopisa i značajke podcrtavanja pogrešnih riječi, ni u jednoj drugoj situaciji nisam pisao riječ *nepodcrtavanje*. Ta riječ postoji, koristim je, na kraju krajeva, u ovome tekstu, ali se ne radi o frekventnoj riječi. Takve riječi mislim da ne treba uključivati u rječnik za provjeru pravopisa. Možda se uključivanje duljih riječi i može opravdati jer je mogućnost da se dulja riječ poklopi s kojom drugom puno manja negoli u kraćih riječi gdje duljina ograničava kombinaciju slova.

Alati otvorena koda imaju svoje nedostatke, ali ih imaju i vlasnički alati i programi. Često ove druge rade stručnjaci, ali isto tako često takvi alati nisu dovoljno baždareni jer nije sve zapisano u pravopisnim i gramatičkim priručnicima ujedno i često i prihvaćeno u široj upotrebi. Alati su otvorena koda, pa onda i rječnik za provjeru pravopisa, odraz zajednice koja takav alat koristi pa se ono što zajednica ne prihvaća iz rječnika isključuje, a ono što nedostaje u rječnik se uključuje. Otvorenost je prednost Hunspella nad ostalim alatima, odnosno to je značajka koja bi Hunspell trebala činiti posebno cijenjenim u školstvu i znanosti.

Provjera bi računalne provjere pravopisa, ali vjerujem i hrvatski jezik uopće, imala koristi od jednoga slobodnoga rječnika izrađenoga na profesionalan i stručan način koristeći alate otvorena koda koji bi bio dostupan na internetu i dostupan svima za korištenje. Rječnik koji ne bi trebao emulirati papirni rječnik, već bi iz temelja treba biti izrađen kao elektroničko izdanje koje bi koristilo sve posebnosti i pogodnosti računalne obrade podataka pa bi onda i takav rječnik mogao služiti kao predložak za izradu rječnika za računalnu provjeru pravopisa,

računalnu provjeru gramatike alatima poput LanguageToola. Uostalom, takav bi rječnik mogao jednostavno služiti i za izradu specijalističkih rječnika jer ne bi trebalo razdvajati rječnike na opće (jednojezične) i rječnike stranih riječi kako se to radi u papirnim izdanjima jer bi jednim klikom korisnik mogao isključiti ili uključiti u pretraživanje uskostručne termine. A jedan bi takav klik, ako je rječnik dobro sastavljen, mogao stvoriti i specijalistički rječnik koji bi se mogao koristiti kao dodatak osnovnome rječniku.

Otvorenost i sloboda ovoga alata omogućuje korisnicima da se obrate održavatelju pravila te da u dogovoru mijenjaju i dorađuju. Hrvatski jezik ima gotov rječnik za računalnu provjeru pravopisa, ali prostora za promjenu i izmjenu ima i više negoli bi održavatelj takva rječnika i htio.

Sažetak

Zadaća je ovoga diplomskoga rada prikazati mogućnosti izrade računalne provjere pravopisa slobodnim alatom Hunspell. U radu će se prikazati što zapravo računalna provjera pravopisa jest, koji su prednosti i nedostaci Hunspella, odnosno prikazat će se što se od računalne provjere pravopisa ovim alatom i može. U radu će se pokušava dati kratak pregled metode izrade pravila. Predstavit će se kriteriji za odabir korpusa riječi koji bi se u takav rječnik trebale uključiti, pokušavaju se prikazati najčešći problemi izrade računalne provjere pravopisa Hunspellom, ali se isto daje i prijedlog kako formirati pravila na primjeru imenica a-sklonidbe hrvatskoga jezika. Rad se ne dotiče drugih vrsta riječi i ne prikazuju se sve značajke Hunspella koje bi se mogle iskoristiti za izradu rječnika za provjeru pravopisa te se koncentrira samo na sufiksaciju imenica spomenutoga tipa. Na kraju se daje i prijedlog za automatizaciju izrade rječnika korištenjem računalnih skripti. Sama računalna provjera pravopisa nije dostatna i treba ju upariti i s drugim alatima slobodna kôda kao što su alati za računalnu provjeru gramatike.

Ključne riječi

Hunspell, računalna provjera pravopisa, rječnik, računalo

Popis literature i internetskih izvora

Literatura

1. Barić *et al., Hrvatska gramatika*, Školska knjiga, Zagreb, 2005.
2. Ivan Marković, *Uvod u jezičnu morfologiju*, Disput, Zagreb, 2012. str. 6 – 12.
3. Josip Silić, *Funkcionalni stilovi hrvatskoga jezika*, Disput, Zagreb, 2006.
4. Josip Silić, Ivo Pranjković, *Gramatika hrvatskoga jezika za gimnazije i visoka učilišta*, Školska knjiga, Zagreb, 2007.
5. Ljiljana Butković, „Metodički model obradbe vremenske raslojenosti leksika“, *Život i škola: časopis za teoriju i praksu odgoja i obrazovanja*, Vol. LIV, No. 19, svibanj 2008.
6. Povijest hrvatskoga jezika / Književnost i kultura devedesetih / Mićanović, Krešimir (ur.). - Zagreb : Zagrebačka slavistička škola , 2012. 65-96 (ISBN: 978-953-175-431-6) (dostupno na https://bib.irb.hr/prikazi-rad?&rad=538528; pristupljeno 14. studenoga 2017.)
7. Stjepko Težak, Stjepan Babić, *Gramatika hrvatskoga jezika*, Školska knjiga, Zagreb, 2009.
8. Vinkoslav Galešev i sur., *Informatika i računalstvo: multimedijski udžbenik informatike i računalstva za srednje škole i gimnazije*, SysPrint, Zagreb, 2006.
9. Vladimir Anić, *Veliki rječnik hrvatskoga jezika*, ur. Ljiljana Jojić, Novi Liber, Zagreb, 2006.

Internetski izvori

1. `http://enciklopedija.hr/Natuknica.aspx?ID=44608`
   pristupljeno 9. prosinca 2017.
2. `http://enciklopedija.hr/Natuknica.aspx?ID=48431`
   pristupljeno 9. prosinca 2017.
3. `http://enciklopedija.hr/natuknica.aspx?ID=68626`
   pristupljeno 3. studenoga 2017.
4. `http://hjp.znanje.hr`
   pristupljeno 30. studenoga 2017.
5. `http://hjp.znanje.hr/`
   pristupljeno 11. prosinca 2017.
6. `http://hol.lzmk.hr/clanak.aspx?id=32130`
   pristupljeno 8. siječnja 2017.
7. `http://hunspell.github.io/`
   pristupljeno 5. studenoga 2017.
8. `http://pravopis.hr/pravila/`
   pristupljeno 14. studenoga 2017.

9. http://wiki.languagetool.org/tips-and-tricks
pristupljeno 1. studenoga 2017.
10. http://www.elastic.co/
pristupljeno 30. studenoga 2017.
11. http://www.enciklopedija.hr/natuknica.aspx?ID=4149
pristupljeno 3. studenoga 2017.
12. http://www.enciklopedija.hr/natuknica.aspx?ID=68100
pristupljeno 3. studenoga 2017.
13. http://www.makeuseof.com/tag/software-vendor-lock-avoid/
pristupljeno 9. prosinca 2017.
14. http://www.utf8-chartable.de/
pristupljeno 9. studenoga 2017.
15. http://www.zdnet.com/article/almost-all-the-worlds-fastest-supercomputers-run-linux/
pristupljeno 5. studenoga 2017.
16. https://cgit.freedesktop.org/libreoffice/lightproof/
pristupljeno 14. studenoga 2017.
17. https://en.wikipedia.org/wiki/Spell_checker
pristupljeno 14. studenoga 2017.
18. https://github.com/akoncic/rjecnikstranihrijeci-android
pristupljeno 11. prosinca 2017.
19. https://github.com/elastic/elasticsearch;
pristupljeno 30. studenoga 2017.
20. https://github.com/hunspell/hunspell/tree/master/tests/v1cmdline
pristupljeno 13. studenoga 2017.
21. https://github.com/krunose/hr-hunspell
pristupljeno 13. studenoga 2017.
22. https://languagetool.org/
pristupljeno 14. studenoga 2017.
23. https://linux.die.net/man/1/hunspell
pristupljeno 5. studenoga 2017.
24. https://play.google.com/store/apps/details?id=com.dekoraktiv.android.rsr
pristupljeno 11. prosinca 2017.
25. https://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/
pristupljeno 9. studenoga 2017.
26. https://support.google.com/websearch/answer/106230
pristupljeno 9. prosinca 2017.
27. https://www.ancient.eu/writing/; pristupljeno
10. prosinca 2017.
28. https://www.britannica.com/technology/computer-programming-language
pristupljeno 3. studenoga 2017.

29. https://www.britannica.com/technology/digital-computer
pristupljeno 3. studenoga 2017.

30. https://www.britannica.com/technology/machine-language
pristupljeno 3. studenoga 2017.

31. https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/history.html
pristupljeno 3. studenoga 2017.

32. https://www.gnu.org/licenses/gpl-faq.html
pristupljeno 5. studenoga 2017.

33. https://www.gnu.org/philosophy/categories.html#FreeSoftware
pristupljeno 5. studenoga 2017.

34. https://www.gnu.org/philosophy/categories.html#OpenSource
pristupljeno 5. studenoga 2017.

35. https://www.gnu.org/philosophy/categories.html#ProprietarySoftware
pristupljeno 5. studenoga 2017.

36. https://www.gnu.org/philosophy/free-sw.html
pristupljeno 3. studenoga 2017.