TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 773

Optimizacija parametara ovisnosnog parsera za hrvatski jezik

Petra Bevandić

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2014.

DIPLOMSKI ZADATAK br. 773

Pristupnik: **Petra Bevandić**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Optimizacija parametara ovisnosnog parsera za hrvatski jezik**

Opis zadatka:

Strojna sintaktička analiza rečenice ili parsanje preduvjet je za više razine strojne obrade teksta, poput semantičke analize ili ekstrakcije informacija. Za parsanje morfološki složenih jezika poput hrvatskoga posebno se uspješnom pokazala paradigma temeljena na ovisnoj gramatici. Unatoč tome, zbog visoke morfološke složenosti, kvaliteta parsanja hrvatskoga jezika i dalje nije zadovoljavajuća. Točnost parsanja mogla bi se poboljšati odgovarajućom optimizacijom parametara ovisnosnog parsera.
U okviru diplomskoga rada potrebno je proučiti način rada postojećih ovisnosnih parsera temeljenih na postupcima strojnoga učenja, s naglaskom na parsere za morfološki složene jezike. Eksperimentalno ispitati kako različiti načini prikaza morfoloških značajki pojavnica utječu na točnost parsera. U obzir uzeti nekoliko javno dostupnih modela statističkih parsera, poput modela MSTParser, MaltParser i Mate. Razraditi postupak za heurističku optimizaciju parametara ovisnosnog parsera, uključivo morfoloških značajki na ulazu u parser, a koji je vođen točnošću parsera na ispitnome skupu kao ciljnom funkcijom. Razviti programsku implementaciju postupka i primijeniti ga na odgovarajuću, javno dostupnu banku stabala hrvatskoga jezika. Provesti eksperimentalno vrednovanje razvijenog modela, usporedbu s raspoloživim parserima te analizu pogreška. Ispitati primjenjivost i učinkovitost razvijenog modela na srodnim jezicima. Radu priložiti izvorni i izvršni kod razvijenog sustava, označene skupove podataka i potrebnu dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 30. lipnja 2014.

Mentor:

Doc. dr.sc. Jan Šnajder

Djelovođa:

Doc. dr.sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr.sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Unutar područja umjetne inteligencije jedno od najzanimljivijih zasigurno je obrada prirodnog jezika. Kada bi računalo moglo razumjeti prirodan jezik, s jedne bi se strane olakšala komunikacija korisnika s računalom, no i omogućilo računalu da dohvaća znanja i informacije iz istih izvora kao i čovjek - knjiga, članaka i drugih tekstualnih zapisa. Ovaj je problem težak ponajprije zato što je prirodni jezik gotovo nemoguće formalizirati – jezik se neprestano mijenja, a čak i pravila koja postoje mogu se ovisno o kontekstu prekršiti. Kako bi se omogućilo računalu da prati promjene u jeziku, a da se pritom izbjegne ručno definiranje i redefiniranje pravila, najčešće se primjenjuju metode strojnog učenja.

Jezik čine simboli, značenja te „kod“ koji povezuje značenje sa simbolom. Simboli pisanog prirodnog jezika jesu slova. Pravopis definira pravila povezivanja slova u riječi, gramatika definira kako se riječi povezuju u rečenice, dok semantika definira koje od mogućih, gramatički ispravnih rečenica imaju smisla u stvarnom svijetu. Analiza teksta na svakoj od prethodno navedenih razina rafinira razumijevanje sadržaja teksta - iz riječi koje se nalaze unutar nekog teksta moguće je razaznati temu i sadržaj, sintaksnom analizom moguće je primjerice prepoznati tko je izvršio neku radnju dok semantika može tog vršitelja radnje povezati sa osobom u stvarnom svijetu. Sve su ove razine međusobno isprepletene - da bi se prepoznalo radnju opisanu u tekstu potrebno je prepoznati da je neka riječ simbol radnje, no ova je veza dvosmjerna - viša razina može pomoći razriješiti višeznačnost niže razine.[1]

Gramatika je most između simbola i značenja. Dijeli se na morfologiju i sintaksu. Morfologija analizira i definira oblik riječi u rečenici, dok sintaksa analizira i definira pravila povezivanja i ulogu riječi u rečenici. Postupak sintaksne analize rečenice naziva se parsanje. Jedan od formalizama kojima se definira sintaksa jesu ovisnosne

[1] U isprepletenosti različitih razina analize teksta leži još jedan razlog za korištenje strojnog učenja u strojnoj obradi teksta - pravilima je možda jednostavno definirati gramatički ispravne konstrukcije, no teško je definirati koje su gramatički ispravne konstrukcije i smislene u stvarnom svijetu. Primjerima se ta pravila implicitno unose u postupak analize teksta.

gramatike koje su zadnjih godina postale popularne zbog svoje jednostavnosti, lakoće povezivanja s metodama strojnog učenja te dobrim rezultatima u radu s jezicima slobodnog poretka riječi u rečenici kao što je hrvatski. S obzirom na popularnost ovisnosnih gramatika, razvijen je niz univerzalnih parsera primjenjivih na bilo kojem jeziku. Svim je tim parserima zajedničko da u postupku treniranja i parsanja koriste riječi u rečenici, ali i dodatne podatke poput morfoloških karakteristika riječi u rečenici. Korisnik u određenoj mjeri može manipulirati podacima koji se nalaze na ulazu u parser kao i parametrima rada parsera. S obzirom na raznolikost postojećih jezika, jasno je da optimalni parametri rada parsera i optimalni podaci na ulazu u parser nisu isti za sve jezike. Kako su i značajke riječi u rečenici koja se parsa rezultat primjerice morfološke analize (koja nije nužno potpuno ispravna), u cijeli se postupak uvodi greška koja se propagira kroz cijeli postupak analize rečenice i narušava točnost izlaza korištenog parsera. S druge pak strane, čak i ako ima potpuno točnu informaciju na ulazu, parser i sam potencijalno može griješiti. Ovo dovodi do dva suprotstavljena zahtjeva - poželjno je parseru predati što više kvalitetnih podataka koji će doprinijeti točnosti parsanja, no s druge strane, što je manja količina pretprocesiranja, to je manja i greška koja se unosi na ulazu u parser. Također, optimalan skup parametara rada parsera ne mora se razlikovati od jezika do jezika; može se razlikovati i unutar istog jezika za rečenice različitih svojstava (primjerice dulje rečenice možda trebaju detaljniju morfološku analizu nego kratke rečenice). Treći je motiv za traženje optimalnog skupa podataka pokušati uočiti postoje li nepotrebne značajke. Kako je i pokazao ovaj rad, dio značajki niti narušava niti poboljšava rezultate rada parsera, ali može povećati vremenske i memorijske zahtjeve parsera te povećava cijenu i količinu posla koju obavljaju ljudi i u razvoju baza ovisnosnih rečenica za učenje modela ovisnosnog parsanja.

Cilj je ovog rada bio implementirati sustav koji bi temeljem rada postojećih, javno dostupnih parsera omogućio ispitivanje utjecaja morfoloških značaji na kvalitetu ovisnosnog parsanja, odnosno pronalazak optimalnog skupa morfoloških značajki za neki jezik uzimajući u obzir rezultate rada parsera te dodatno ispitati utjecaj reduciranog skupa značajki na cijeli lanac analize rečenice. Implementirani sustav zatim je upotrebljen za ispitivanje utjecaja morfoloških značajki na rad nekoliko najpopularnijih ovisnosnih parsera ne bi li pronašao onaj parser i one morfološke značajke koje daju najbolje rezultate za rečenice hrvatskog jezika. Najbolji dobiveni model (koji je ujedno i dosta manji i odnosu na potpuni skup morfoloških značajki) ispitan je na jezicima srodnim hrvatskom: slovenskom i češkom.

U prvom dijelu rad opisane su ovisnosne gramatike i ovisnosna stabla kao temeljna metoda prikaza rezultata ovisnosnog parsanja. Nakon toga ocrtani su temeljni pristupi

ovisnosnom parsanju rečenica temeljeni na podacima koje koriste i korišteni dostupni parseri. U drugom je dijelu opisan implementirani optimizator koji omogućava nekoliko različitih metoda za traženje potencijalno kvalitetnih skupova morfoloških atributa. Sam sustav ne daje odgovor na pitanje koji je skup atributa optimalan, no sustav omogućuje korisniku da nad odabranim skupovima atributa radi dodatna relevantna ispitivanja. Treći dio donosi detaljan opis korištenih parsera i baza ovisnosnih stabala te opisuje kako je implementirani optimizator upotrebljen za pronalazak optimalnog skupa značajki za hrvatski jezik. U zadnjem su dijelu izneseni dobiveni rezultati. Na samom se kraju nalazi zaključak, iznose moguća poboljšanja i potencijalni smjer budućih istraživanja.

2. Ovisnosne gramatike

Najlakši način za opisati ovisnosne gramatike (engl. *dependency grammar*) jest usporedbom s formalnim gramatikama (engl. *formal grammar*). Formalne gramatike definiraju pravila povezivanja riječi u cjeline, odnosno način povezivanja tih cjelina u neke složenije strukture i konačno rečenice. Nasuprot tome, ovisnosne gramatike definiraju veze koje mogu postojati među riječima u rečenici te pravila prema kojima te veze nestaju. U odnosu na formalne gramatike, ovaj je formalizam mnogo jednostavniji i lakše se nosi s jezicima koji imaju relativno slobodan poredak riječi u rečenici.

Postoji niz raznih pristupa definiranju ovisnosnih gramatika koje se razlikuju u načinu definiranja dozvoljenih veza koje postoje među riječima,[1] a u nastavku se daju temeljne pretpostavke koje su im svima zajedničke. Nakon toga se definira ovisnosno stablo kao osnovna metoda prikaza rezultata parsanja rečenica ovisnosnim gramatikama i njegova osnovna svojstva. Konačno, s obzirom da je jedan od osnovnih razloga za popularnost ovisnosnih gramatika činjenica da ih je jednostavno kombinirati s metodama strojnog učenja, opisana su dva osnovna pristupa ovisnosnom parsanju rečenica temeljenom na podacima: pristup temeljen na prijelazima (engl. *transition-based parsing*) i pristup temeljen na grafovima(engl. *graph-based parsing*). Na samom je kraju dan pregled najčešćih mjera za vrednovanje kvalitete ovisnosnih stabala dobivenih radom parsera.

Ovo se poglavlje u najvećoj mjeri oslanja na (Kubler et al., 2009).

2.1. Temeljne pretpostavke

Rečenica je niz riječi: $s = (w_1, .., w_k)$. Riječi unutar rečenice povezane su binarnim asimetričnim označenim vezama – ovisnosnim vezama (engl. *dependency relation*). Asimetrija veze proizlazi iz pretpostavke da je kod povezanih riječi jedna riječ uvijek važnija - jedna riječ je nadređena drugoj. U nastavku nadređena riječ nazivat će se

[1] Veze među riječima mogu biti i semantičke a ne samo sintaktičke, dok neke varijante ovisnosnih gramatika čak dozvoljavaju veze među skupovima riječi.

glavom (engl. *head*), a podređene riječ ovisnom riječi (engl. *dependent*). Dodatno, veza koja postoji među riječima može se imenovati.

Ako pretpostavimo da glava H i ovisna riječ D čine cjelinu C, ovisnosnu se vezu može prepoznati na sljedeći način (Zwicky, 1985):

1. H određuje sintaktičku kategoriju C i može se (uglavnom) umetnuti umjesto C. Tako u primjeru „Crn pas vidi psa.“ riječ „pas“ može zamijeniti cjelinu „crn pas“, a da rečenica i dalje zadržava sintaksnu i semantičku ispravnost - moguća je rečenica „Pas vidi psa.“, dok je „Crn vidi psa.“ neispravna rečenica. Dodatno, „pas“ je subjekt, a „crn pas“ subjektni skup;

2. H određuje semantičku kategoriju C, a D specificira H. Tako je u prethodnom primjeru „crn pas“ definira psa, dok pridjev „crn“ daje dodatnu informaciju o izgledu psa;

3. H je obvezan dio C, dok se D može izostaviti - ovo pravilo se zapravo nadovezuje na prvo pravilo o zamjeni C sa H. izostavljanjem H gubi se podatak o tome tko ustvari vidi psa;

4. H otvara mjesto D, i određuje je li ovaj potreban ili ne. Tako atribut „crn“ nije obavezan za glavu „pas‘, no određene glave mogu zahtijevati nadopunu. Tako primjerice u cjelini „u Zagrebu“ riječ „Zagrebu“ zahtjeva prijedlog „u“ da bi mogla označiti mjesto;

5. Oblik D ovisi o H. Tako se zahtjeva da pridjev „crn“ bude u nominativu jednine muškog roda;

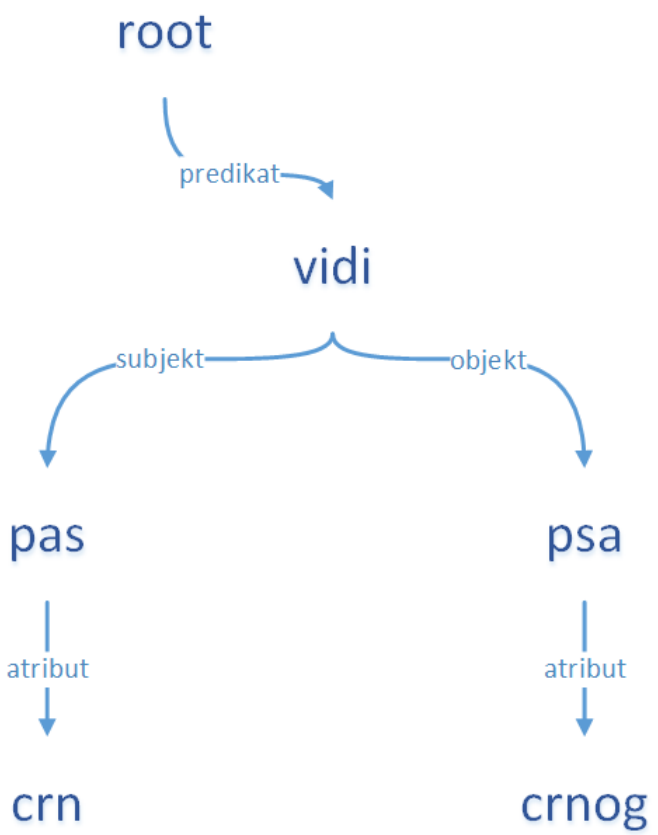
6. Položaj D ovisi o H. Očito je da kod jezika sa slobodnom strukturom postoje slabiji zahtjevi na položaj ovisne riječi, no u primjeru „u Zagrebu“ „u“ se uvijek mora nalaziti ispred „Zagrebu“.

Dana pravila imaju i određena ograničenja. Jedno od najvažnijih jest zahtjev da su sve riječi hijerarhijski uređene. Problem se očituje u primjeru „Pas vidi psa i laje“. U tom su primjeru glagoli „vidi“ i „laje“ jednako vrijedni, a riječ „pas“ je subjekt i jednog i drugog glagola. Način razrješavanja ovog problema različite ovisnosne gramatike definiraju na različite načine. Jedna ideja je da se između ta dva glagola ipak uvede hijerarhija uz proširenje mogućih oznaka veza, druga bi mogla biti da se na veznik „i“ povežu svi glagoli koji su koordinirani, kao i subjekt „pas“, dok je treća da se veze dozvole među grupama riječi, ne samo pojedinim riječima.

U nastavku će se veza označavati uređenom trojkom (w_i, r, w_j) gdje je w_i glava, w_j ovisna riječ, a veza koja postoji među njima označena je oznakom r.

2.2. Ovisnosna stabla

Kao što su se nizovi generirani regularnim gramatikama mogli prikazati parsnim stablima tako se i veze među riječima mogu prikazati pomoću stabla koje će se nazivati ovisnosno stablo (engl. *dependency tree*). Slika 2.1 prikazuje primjer jednog takvog stabla. Vidljivo je da se u čvorovima stabla nalaze riječi (i upravo je zbog toga prilikom izgradnje stabla nevažan međusobni položaj riječi u rečenici), dok su veze između riječi označene. Oznaka veze predstavlja sintaktičku funkciju ovisne riječi. „root“ je umjetno dodana riječ koja će uvijek predstavljati korijen rečenice.



Slika 2.1: Ovisnosno stablo za rečenicu „Crn pas vidi crnog psa“. Veze su usmjerene od glava prema ovisnim riječima i označene.

Matematički se stablo može definirati na sljedeći način: $G = (V, A)$, gdje je V skup svih riječi u rečenici, a A skup svih bridova.

Stabla imaju sljedeća svojstva:

1. Stablo ima samo jedan korijen. Ovo znači da samo jedna riječ u rečenici nema nadređenu riječ. Da bi se izbjeglo uvođenje neprirodnih ovisnosti (primjerice između koordiniranih riječi), umjetno se određuje da je korijen svake rečenice umjetno dodana nulta riječ „root“. Tako u rečenice „Pas laje, a mačka mijauče.“ niti jedan glagol neće biti podređen, nego će oba kao glavu imati korijen „root“;

2. Sve riječi neke rečenice moraju se nalaziti u stablu;

3. Svaka riječ u rečenici ima točno jednu glavu (pritom se smatra da korijen „root“ nije dio rečenice te da smije imati glavu);

4. Stablo nema cikluse;

5. Stablo ima n veza (gdje je n duljina rečenice, i smatra se da korijen nije dio rečenice). Veze su označene, te oznaka veze ujedno predstavlja i sintaktičku funkciju podređene riječi u rečenici.

Primjer 2.1 prikazuje zapis jednog ovisnosnog stabla.

Primjer 2.1 Ovisnosno stablo za rečenicu „Crn pas vidi crnog psa “.

$s =$ „$Crn\ \ pas\ \ vidi\ \ crnog\ \ psa$“
$V = \{crn,\ pas,\ vidi,\ crnog,\ psa\}$
$A = \{(pas,\ atribut,\ crn),\ (vidi,\ subjekt,\ pas),\ (vidi,\ objekt,\ psa),$
$(psa,\ atribut,\ crnog),\ (root,\ predikat,\ vidi)\}$

2.2.1. Projektivnost

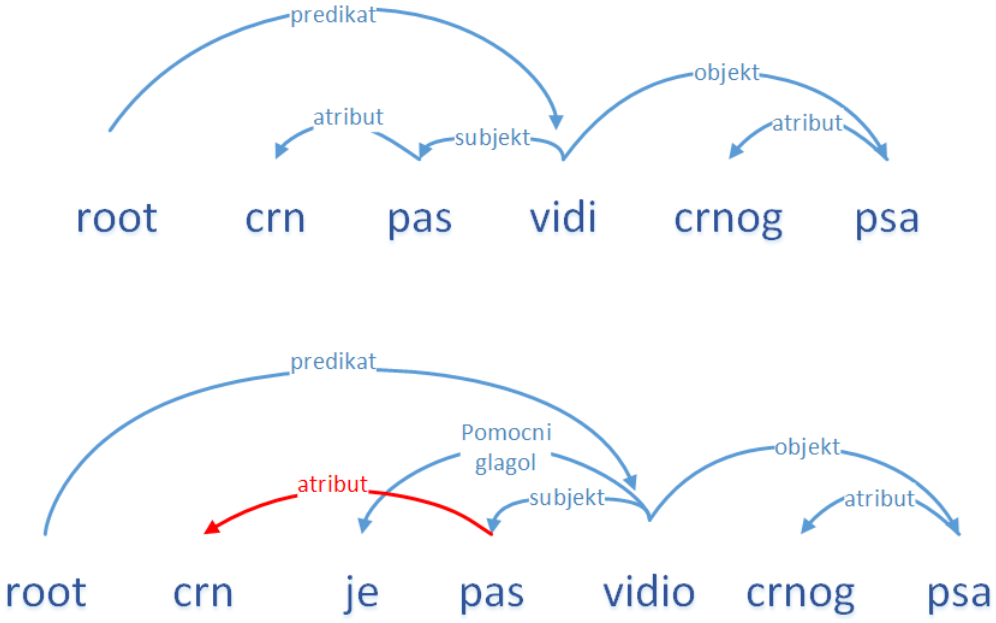
Zbog jednostavnosti, na stablo se može uvesti i jedno dodatno ograničenje koje se naziva projektivnost (engl. *projectivity*).

Projektivna veza je ona veza za koju vrijedi:

$$(w_i, r, w_j) \Rightarrow w_i \rightarrow^* w_k,\ \forall k((i < k < j \wedge i < j) \vee (j < k < i \wedge j < i))$$

$w_i \rightarrow^* w_k$ označava nadređenost koja ne mora biti izravna, odnosno vrijedi ili $w_i \rightarrow w_k$ ili $w_i \rightarrow w_j \wedge w_j \rightarrow^* w_k$. Uzme li se da podstablo ovisnosnog stabla čini jednu cjelinu, ovaj je zahtjev zapravo identičan onom koji imaju frazne gramatike – riječi unutar fraze ne smiju biti razdvojene.

Ovisnosno je stablo projektivno ako su mu sve veze projektivno, odnosno neprojektivno ako ima barem jednu neprojektivnu vezu.



Slika 2.2: Primjeri projektivnog i neprojektivnog stabla. Gornje stablo za rečenicu „Crn pas vidi crnog psa“ je projektivno, dok je donje stablo za rečenicu „Crn je pas vidio crnog psa“ neprojektivno. Neprojektivna je veza označena crvenom bojom.

Slika 2.2 prikazuje grafičku interpretaciju svojstva projektivnosti. Napišu li se riječi u rečenici jedna za drugom u svom izvornom redoslijedu, a sve se veze prikažu iznad rečenice tada je projektivno stablo planarno, odnosno može se nacrtati bez presijecanja bridova.

2.3. Metode ovisnosnog parsanja

Iako je moguće formalno definirati ovisnosne gramatike te parsirati temeljem definiranih pravila, popularnost ovisnosnih gramatika proizlazi upravo iz činjenice da je lako u postupak parsanja uklopiti postupke strojnog učenja (engl. *machine learning*). Metode parsanja temeljene na podacima mogu se podijeliti u dvije skupine: metode koje se temelje na grafovima (engl. *graph-based parsing*) i metode koje se temelje na prijelazima (engl. *transition-based parsing*). U nastavku su ukratko ocrtana oba ova pristupa.

2.3.1. Metode parsanja temeljene na prijelazima

Prvi pristup parsanju ovisnosnih gramatika parser promatra kao automat s konačnim brojem koji čita riječ po riječ s ulaza, i ovisno o ulazu mijenja svoje stanje. Nakon što su sve riječi pročitane s ulaza, automat na izlazu daje parsno stablo.

Algoritam 1 prikazuje opći algoritam rada prijelazničkog parsera. Početno stanje automata $c_0(s)$ neka je funkcija ulazne rečenice s. Zatim se sve dok parser ne dođe u konačno stanje C_T na temelju trenutnog stanja odabire prikladan prijelaz, a novo stanje je rezultat primjene odabranog prijelaza na trenutno stanje.

Algoritam 1: Opći algoritam parsera temeljenog na prijelazima

$c \leftarrow c_0(s)$
dok $c \neq C_T$ **radi**
    $t \leftarrow oracle(c)$
    $c \leftarrow t(c)$
$vrati\ G(s)$

Nekoliko je elemenata ovog algoritma koji su nedefinirani. Prva je pitanje prikaza stanja – što definira trenutno stanje parsera. Druga je kako izgledaju prijelazi iz stanja u stanje. Zadnje pitanje je kako se, jednom kada su prikaz stanja i mogući prijelazi odabrani, za trenutno stanje odabire najbolji prijelaz. U algoritmu 1 odluku o prijelazu u sljedeće stanje donosi proročište (engl. *oracle*) koje se se tipično implementira kao klasifikator, pri čemu je uzorak koji se klasificira trenutno stanje parsera, dok mogući prijelazi predstavljaju klase u koje se taj uzorak može klasificirati. Modeliranje proročišta moguće je nekom od metoda strojnog učenja. U nastavku je ilustrirano jedno moguće rješenje definicije stanja automata i prijelaza.

Algoritam arc-standard

Algoritam arc-standard (Nivre, 2008) za parsanje algoritam je koji se temelji na stogu i radi na skupu projektivnih rečenica.

Jedna konfiguracija rečenice $s = (w_1, .., w_n)$ predstavljena je uređenom trojkom (σ, β, A), gdje je:

1. σ stog s riječima $w_i \in s, i \leq k, k \leq n$;
2. β ulazna traka s koje automat čita $w_j \in s, j > k$;

3. A skup dotad dodanih ovisnosnih veza, dakle djelomično izgrađeno ovisnosno stablo.

Dozvoljena su tri moguća prijelaza:

1. $LEFT - ARC_r$ $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, j|\beta, A \cup \{(j, r, i)\})$ dodaje novu vezu s oznakom r u graf, riječ w_i s vrha stoga je ovisna o riječi w_j s početka niza. Ovisna se riječ uklanja s vrha stoga i više ne razmatra, dok glava može biti dio novih veza. Moraju biti zadovoljena dva preduvjeta: w_i ne smije biti korijen rečenice te ne smije imati već dodijeljenu glavu;

2. $RIGHT - ARC_r$ $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, i|\beta, A \cup \{(i, r, j)\})$ dodaje novu vezu s oznakom r u graf, riječ w_i s vrha stoga je glava riječi w_j s početka niza. Ovisna se riječ uklanja s početka niza i zamjenjuje s riječi s vrha stoga, dok glava može biti dio novih veza. Jedini je preduvjet u ovom slučaju da w_j nema već dodijeljenu glavu;

3. SHIFT $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$. uklanja riječ w_i s ulazne trake i postavlja ju na stog.

Početno stanje za neku rečenicu s duljine n je $c_0 = ([0], [1, .., n], \emptyset)$ Postupak završava kada se isprazni ulazna traka β. Ukoliko u tom trenutku postoje neke riječi ulazne rečenice koje nemaju glavu one se umjetno povezuju s korijenom.

Tablica 2.1 prikazuje postupak parsanja projektivne rečenice „Crn pas vidi crnog psa“.

Sam parser arc-standard predstavlja najjednostavniji mogući pristup definiciji stanja i prijelaza te ima nekoliko ograničenja. Jedan od problema u radu prethodno opisanog algoritma jest prijelaz $RIGHT - ARC_r$. Naime, on uklanja ovisnu riječ iz daljnjeg razmatranja i, ukoliko je ta uklonjena riječ bila nadređena jednoj ili nizu riječi koje dolaze poslije, može se dogoditi da parser propusti dio veza u rečenici. Drugo je ograničenje to što ovaj parser ne može parsati neprojektivna stabla. U promjeru „Crn je pas vidio crnog psa“ jedini što je moguće napraviti s riječima „crn“ i „je“ jest staviti ih na stog. S obzirom da će se u kasnijim koracima riječ „pas“ ukloniti iz daljnjeg razmatranja, riječ „crn“ nikada neće biti pridružena svojoj glavi. Navedeni se problemi mogu riješiti uvođenjem složenijih prijelaza: primjerice omogućavanjem skidanja riječi sa stoga ili omogućavanjem zamjene pozicija riječi na ulaznoj traci, no te se mogućnosti neće detaljnije razmatrati.[2]

[2]Primjer složenijeg prijelazničkog sustava moguće je naći u u potpoglavlju 4.2.1.

Prijelaz	Konfiguracija		
	$([0],$	$[1,2,3,4,5],$	$\emptyset)$
$SHIFT$	$([0,1],$	$[2,3,4,5],$	$\emptyset)$
$LEFT-ARC_{atribut}$	$([0],$	$[2,3,4,5],$	$A_1 = \{(2, atribut, 1)\})$
$SHIFT$	$([0,2],$	$[3,4,5],$	$A_1)$
$LEFT-ARC_{subjekt}$	$([0],$	$[3,4,5],$	$A_2 = A_1 \cup \{(3, subjekt, 2)\})$
$SHIFT$	$([0,3],$	$[4,5],$	$A_2)$
$SHIFT$	$([0,3,4],$	$[5],$	$A_2)$
$LEFT-ARC_{atribut}$	$([0,3],$	$[5],$	$A_3 = A_2 \cup \{(5, atribut, 4)\})$
$RIGHT-ARC_{objekt}$	$([0],$	$[3],$	$A_4 = A_3 \cup \{(3, objekt, 5)\})$
$RIGHT-ARC_{predikat}$	$([],$	$[0],$	$A_5 = A_4 \cup \{(0, predikat, 3)\})$
$SHIFT$	$([0],$	$[],$	$A_5)$

Tablica 2.1: Prikaz rada arc-standard parsera

2.3.2. Metode temeljene na grafovima

Dok pristup temeljen na prijelazima promatra parser kao automat koji postepeno gradi ovisnosno stablo, ne treba smetnuti s uma da je stablo zapravo graf. Ovo znači da je moguće iskoristiti teoriju grafova i dobiti matematički utemeljene algoritme za pronalazak najboljeg ovisnosnog stabla.

Definicije

Razapinjuće stablo (engl. *spanning tree*) nekog povezanog grafa G je onaj podgraf grafa G koji sadrži sve vrhove grafa G i istovremeno je stablo. U kontekstu parsanja graf G jest graf čiji su vrhovi riječi neke rečenice, a bridovi sve moguće,[3] odnosno dozvoljene veze za tu rečenicu.[4] Razapinjuće stabla tada je neko od mogućih ovisnosnih stabala za tu rečenicu.

Maksimalno razapinjuće stablo (engl. *maximum spanning tree*) nekog grafa jest ono stablo koje u skupu svih razapinjućih stabala nekog grafa ima najbolju ocjenu . U kontekstu parsanja to će biti ovisnosno stablo s najboljom ocjenom. Da bi se moglo naći maksimalno razapinjuće ovisnosno stablo pretpostavka je da postoji način ocjene kvalitete samog stabla. Modeliranje ocjenjivanja ovisnosnog stabla moguće je napraviti korištenjem strojnog učenja.

[3] Veze među riječima mogu biti gramatički ispravne čak i ako nisu semantički smislene.

[4] Inicijalni skup mogućih ovisnosnih veza koje će poslužiti za daljnje traženje ovisnosnog stabla moguće je smanjiti eliminiranjem besmislenih ili malo vjerojatnih veza.

Chu-Liu-Edmondsov algoritam

Chu-Liu-Edmondsov algoritam (Chu i Liu, 1965; Edmonds, 1967) algoritam je za pronalazak maksimalnog usmjernog razapinjućeg stabla nekog grafa. Usmjereno stablo ima sva svojstva koja imaju i ovisnosna stabla, pri čemu su najvažnija ona o jednom korijenu te postojanju samo jedne glave za svaki čvor stabla (osim korijena).

Algoritam 2 prikazuje pseudokod Chu-Liu-Edmondsovog algoritma. Algoritam kao ulaz prima potpuno povezano stablo u kojem je svaki čvor (riječ) povezana sa svakim čvorom dvostrukom vezom (gdje svaka od dvije veze predstavlja jedan smjer ovisnosti). Svaki od tih bridova mora biti na neki način bodovan. S obzirom da se svakom od bridova može dodijeliti bilo koja labela r, parser može u prvom koraku odabrati onu labelu za koju pojedini brid ima najbolju ocjenu.

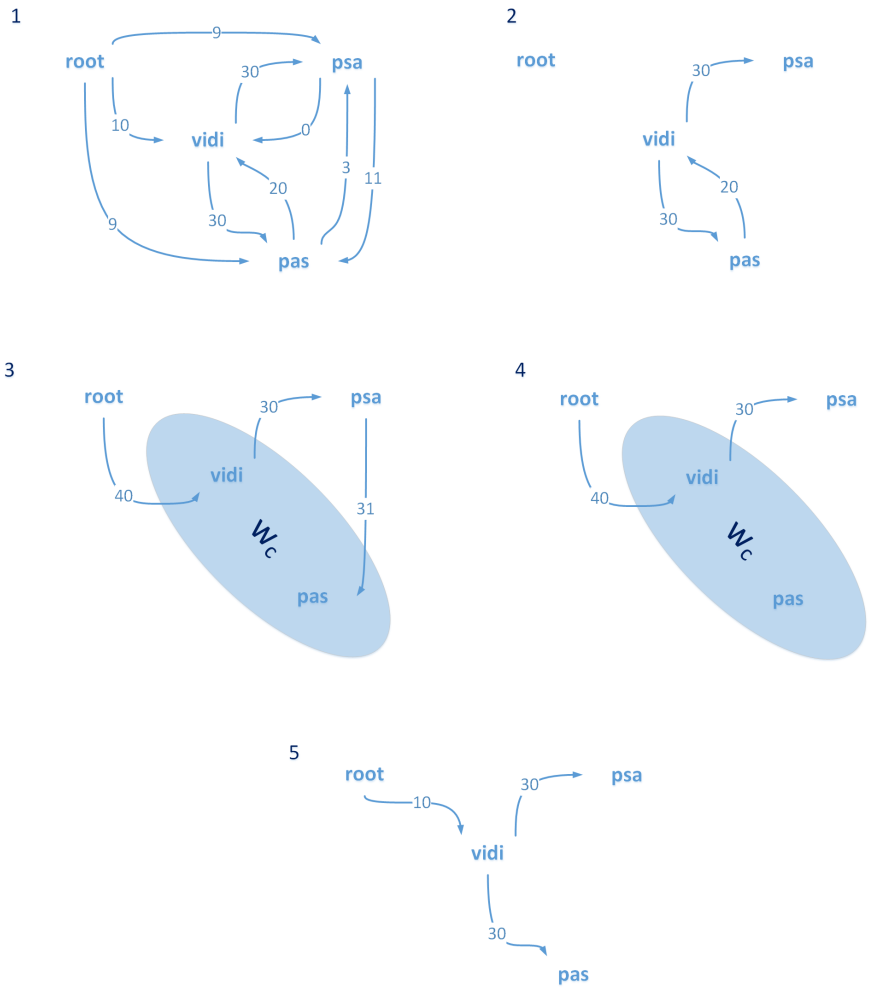
Algoritam radi na sljedeći način. Prvo za svaku točku grafa odabire maksimalan „ulazni“ brid - dakle za svaku točku pronalazi najbolju glavu. Ukoliko je tako dobiven graf stablo, postupak je gotov. Ukoliko nije, tada u stablu postoji neki ciklus. Taj se ciklus zamjenjuje jednom novom točkom w_c koja se s točkama izvan ciklusa povezuje na način definiran u funkciji **Kontrahiraj**. Veza između w_c i neke točke izvan ciklusa zapravo skriva najbolju vezu neke točke izvan ciklusa i neke točke unutar ciklusa. Točku unutar ciklusa za koju je ta najbolja veza postignuta pohranjuje se u matricu ep, koja će kasnije poslužiti za rekonstrukciju originalnog grafa. Za taj se novi graf opet rekurzivno poziva funkcija **Chu-Liu-Edmonds**. Nakon povratka iz rekurzije potrebno je rekonstruirati originalni graf.

Slika 2.3 prikazuje ovaj postupak za rečenicu „Pas vidi psa“. Dan je potpuno povezan graf za rečenicu „Pas vidi psa“. U prvom se koraku pronalaze najveći ulazni bridovi za svaku točku, odnosno za svaku se riječ u rečenici pronalazi najbolja glava. Pohlepnom je metodom dobiven jedan ciklus koji čine riječi „vidi“ i „pas“ . Taj se ciklus zamjenjuje jednom novom točkom te je na slici označeno na koje su originalne vrhove povezani ti novi bridovi (sadržaj matrice ep). Postupak se rekurzivno ponavlja s modificiranim skupom riječi. S obzirom da je dobiveno stablo, događa se povratak iz rekurzije. Pomoću matrice ep obavlja se rekonstrukcija konačnog, najboljeg ovisnosnog stabla.

Algoritam 2: Chu-Liu-Edmondsov algoritam

Chu-Liu-Edmonds(G,λ):
$G = (V, A), \lambda_{(w_i,w_j)} \in \lambda$
$A' = \{(w_i, w_j) : w_j \in V, w_i = argmax_{w_i} \lambda_{(w_i,w_j)}\}$
$G' = (V, A')$
ako G' *acikličan* **onda**
vrati G'
inače
pronađi ciklus A_c u grafu G'
(G_c, w_c, ep)= Kontrahiraj(G', A_c, λ)
G=(V, A)=Chu-Liu-Edmonds(G_c, λ)
za sve $(w_i, w_c) \in A$, $ep(w_i, w_c)$= w_j, pronađi $(w_k, w_j) \in A_c$ za neki w_k
pronađi sve veze $(w_c, w_l) \in A$
$A = \{A \cup \{(ep(w_c, w_l), w_l), \forall (w_c, w_l) \in A\} \cup A_c \cup (w_i, w_j)\} - (w_k, w_j)$
$V = V$
vrati G

Kontrahiraj(G,C,λ):
$G_c = G - C$
$G_c = G_c \cup w_c w_C \notin V$
za $w_j \in V - C, \exists (w_i, w_j) \in A, w_i \in C$ **radi**
dodaj (w_c, w_j) u G_c dako da vrijedi:
$ep(w_c, w_j) = argmax_{w_i \in C} \lambda_{(w_i,w_j)}$
$w_i = ep(w_c, w_j)$
$\lambda_{(w_c,w_j)} = \lambda_{(w_i,w_j)}$
za $w_i \in V - C, \exists (w_i, w_j) \in A, w_j \in C$ **radi**
dodaj (w_i, w_c) u G_c dako da vrijedi:
$ep(w_i, w_c) = argmax_{w_j \in C} [\lambda_{(w_i,w_j)} - \lambda_{(h(w_j),w_j)}]$
$w_j = ep(w_i, w_c)$
$\lambda_{(w_i,w_c)} = \lambda_{(w_i,w_j)} - \lambda_{(h(w_j),w_j)} + score(C)$
uz $a(w)$ =prethodnik od w u C i $score(C) = \sum_{w \in C} \lambda_{(a(w),w)}$



Slika 2.3: Prikaz rada Chu-Liu-Edmondsovog algoritma za rečenicu „Pas vidi psa“.

2.4. Vrednovanje različitih parsera

Prilikom vrednovanja izlaza parsera ispituju se dva elementa:

1. Točnost označavanja - mjeri se postotak veza s točno pridruženom oznakom (engl. *labeled score*);

2. Točnost pridruživanja - mjeri se postotak riječi s točno pridruženom glavom (engl. *attachment score*).

Najčešće korištene mjere kvalitete parsanja jesu: LAS (engl. *labeled attachment score*) , UAS (engl. *unlabeled attachment score*), LA (engl. *label attachment*) LEM (engl. *labeled exact match*) i UEM (engl. *unlabeled exact match*). Za pojedinu rečenicu $s = (w_1, .., w_k)$ ove se mjere računaju na sljedeći način:

$$
\begin{aligned}
UAS(s) &= \frac{1}{k}\sum_{i=1}^{k} 1\{h_t(w_i) = h_p(w_i)\} \\
LAS(s) &= \frac{1}{k}\sum_{i=1}^{k} 1\{h_t(w_i) = h_p(w_i) \wedge d_t(w_i) = d_p(w_i)\} \\
LA(s) &= \frac{1}{k}\sum_{i=1}^{k} 1\{d_t(w_i) = d_p(w_i)\} \\
UEM(s) &= \frac{1}{k}\prod_{i=1}^{k} 1\{h_t(w_i) = h_p(w_i)\} \\
LEM(s) &= \frac{1}{k}\prod_{i=1}^{k} 1\{h_t(w_i) = h_p(w_i) \wedge d_t(w_i) = d_p(w_i)\}.
\end{aligned}
$$

$h_t(w_i)$ predstavlja funkciju koja za riječ w_i vraća njezinu točnu glavu, dok $h_p(w_i)$ vraća glavu riječi koja je dobivane u postupku parsiranja. Analogno, $d_t(w_i)$ i $d_p(w_i)$ vraćaju točnu i predviđenu sintaktičku ulogu riječi w_i (odnosno točni i predviđeni r u vezi $(h(w_i), r, w_i)$).

Procjenjuje li se točnost parsanja nad nekim skupom rečenica A, tada se LAS, UAS i LA najčešće računaju nad skupom svih riječi u A, dok se UEM i LEM računaju kao postotak potpuno točno parsanih rečenica.

Dodatno se za svaki tip oznake veze (moguće sintaktičke funkcije riječi u rečenici) može se računati preciznost(engl. *precision*) (udio točno označenih veza nekog tipa u skupu svih veza tog tipa koje je parser vratio), odziv (engl. *recall*)(udio točno označenih veza nekog tipa u skupu svih veza tog tipa koje je parser trebao vratiti) i F_β-mjera, koje se definiraju na sljedeći način:

$$P(r) = \frac{\sum_{s_i \in A} \sum_{w_j \in s_i} 1\{d_p(w_j) = d_t(w_j) = r\}}{\sum_{i=1}^{n} \sum_{w_j \in s_i} 1\{d_p(w_j) = r\}}$$

$$R(r) = \frac{\sum_{s_i \in A} \sum_{w_j \in s_i} 1\{d_p(w_j) = d_t(w_j) = r\}}{\sum_{i=1}^{n} \sum_{w_j \in s_i} 1\{d_t(w_j) = r\}}$$

$$F_\beta(r) = \frac{(1+\beta^2)P(r)R(r)}{\beta^2 P(r)R(r)}.$$

3. Optimizator

Kako je rečeno još u uvodu, postoji niz javno dostupnih parsera za ovisnosno parsanje koji se mogu primjenjivati na bilo kojem jeziku. Svim je tim parserima zajedničko da na ulazu ne primaju samo rečenicu koju treba parsati već mogu primiti i dodatne podatke o riječima koje se nalaze unutar rečenice. Postavlja se pitanje koji je optimalni parser za neki jezik te u kojim uvjetima on najbolje radi. Na rad se parsera može utjecati na dva načina: kroz ugađanje parametara samog parsera ili manipulacijom podataka koji se parseru predaju na učenje. U prvom se slučaju može utjecati primjerice na odabir algoritma parsanja ukoliko ih postoji više ili ugađati koeficijente koji utječu na metodu strojnog učenja koja se koristi u postupku treniranja parsera. U drugom se slučaju može utjecati na značajke riječi koje će se analizirati prije provođenja postupka parsanja i koje će se predati parseru na ulazu. Skup odabranih značajki koje se predaju parseru mora poboljšavati kvalitetu parsanja, no istovremeno ne smije biti niti prevelik. Naime, i sama analiza značajki automatiziran je postupak, te neće biti potpuno točan te ga je potrebno što je više moguće pojednostavniti.

U nastavku je opisano idejno rješenja, a zatim i konkretna implementacija optimizatora. Nakon toga su izloženi formati zapisa ovisnosnih stabala koje optimizator podržava. Na kraju je ukratko opisan način korištenja samog optimizatora.

3.1. Idejno rješenje

U sklopu ovog rada bilo je potrebno implementirati sustav koji bi manipulacijom podataka na ulazu omogućio pronalazak idealnog skupa značajki riječi koji doprinosi kvaliteti ovisnosnog parsanja. Sam optimizator nije osmišljen kao alat koji na izlazu daje jedan optimalan skup značajki za neki jezik, već kao alat koji omogućava detekciju potencijalno zanimljivih skupova značajki te omogućava dodatno testiranje tih skupova značajki na zahtjev korisnika. Ta dodatna analiza prvenstveno se odnosi na ispitivanje kvalitete parsanja na podacima koji nisu potpuno točno, odnosno kod kojih postoji greška koja je posljedica automatizirane analize potrebnih značajki. Značajke

čiji se utjecaj na parsanje razmatra u sklopu ovoga rada jesu morfološke značajke riječi. S obzirom da se ne optimiziraju parametri korištenog parsera, implementirani je optimizatior relativno fleksibilan što se tiče parsera i morfoloških označivača s kojima može raditi, te omogućava nekoliko različitih pristupa definiciji skupa atributa i načinu pretraživanja prostora stanja mogućih skupova ulaznih atributa.

3.1.1. Metode pretraživanja prostora mogućih skupova značajki

Optimizatoru je u svakom trenutku poznata potpuna informacija o ulaznom skupu ovisnosnih stabala. Krećući od inicijalnog skupa značajki, moguće je dodavati nove (još neuvrštene) ili odbacivati postojeće značajke.[1] Nakon modifikacije skupa značajki, generiraju se dva skupa - skup za učenje kojim se trenira parser i gradi model za ispitivani skup morfoloških značajki te skup za testiranje kojim se ispituje dobiveni model. Na skupu za testiranje zatim je moguće procijeniti kvalitetu dobivenog modela (kao mjera kvalitete rada parsera odabran je LAS).

Implementirane su dva načina pretraživanja prostora stanja atributa: pretraživanje u širinu (engl. *breadth-first search*) s ograničenim faktorom grananja (engl. *branching factor*) te pretraživanje po slojevima (engl. *beam search*).

Prilikom pretraživanje u širinu s ograničenim faktorom grananja optimizator prima skup skupova atributa (svaki od tih skupova čini jedna čvor u stablu pretraživanja prostora stanja). Algoritam 3 prikazuje pseudokod pretraživanja u širinu s ograničenim faktorom grananja ukoliko se radi pretraživanje unatrag. Za svaki se čvor iz tog inicijalnog skupa vrši sljedeći postupak: iz skupa atributa izbriše se jedan atribut te se ispita rad parsera na modificiranom skupu atributa. Izbrisani se atribut zatim vrati u skup atributa i izbriše se sljedeći atribut. Opisani se postupak ponavlja za sve atribute u skupu atributa. Nakon što su svi čvorovi „otvoreni“, u sljedeću se iteraciju prosljeđuje ograničen broj najbolje „djece“ za svaki od čvorova (onih skupova za koje je parser radio najbolje) u kojoj se ovaj postupak ponavlja. Analogni se postupak izvodi i ukoliko se u inicijalne skupove dodaju atributi koji se još ne nalaze u njima. Algoritam na ulazu prima parametre parsera koji se ispituje, skup ovisnosnih stabala koje koristi za učenje i testiranje te rezultate prethodnih ispitivanja rada tog parsera na istim podacima. Prethodne je rezultate potrebno pamtiti tako da se prilikom pokretanja optimizatora uz različite početne uvjete ne radi višestruko ispitivanje istog modela. Slika 3.1 prikazuje pretraživanja prostora stanja u širinu unatrag za skup od četiri atributa uz maksimalni faktor grananja koji iznosi dva. Svakom je atributu pridružen indeks u bi-

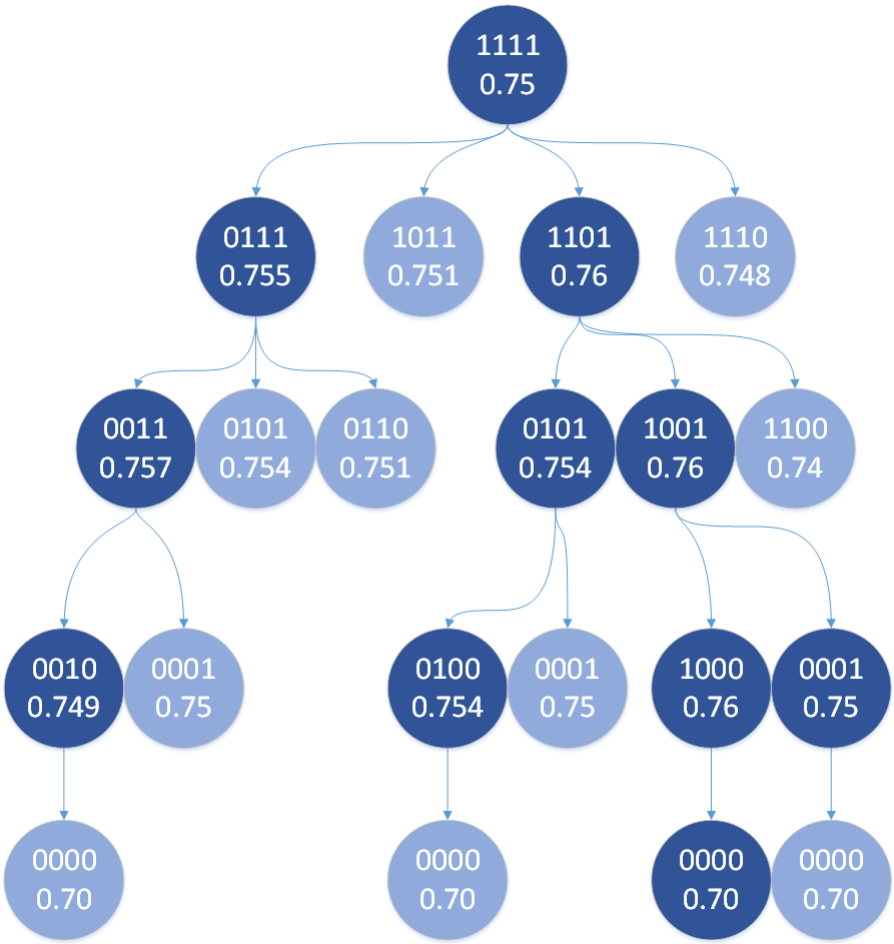
[1] Optimizator uvijek radi u jednom smjeru - ili samo dodaje ili samo briše značajke.

narnom vektoru pri čemu jedinica označava da se neki atribut koristi, a nula označava da se taj atribut ne predaje parseru. Ispod vektora odabranih atributa nalazi se ocjena rada parsera za pripadni skup atributa. Tamno plavom bojom označeni su čvorovi koji se dalje otvaraju. Otvaranje čvora znači razmatranje svih skupova atributa koji imaju jedan atribut manje od otvorenog čvora. Za svaki se čvor dalje otvara maksimalno dvoje djece. Taj broj može biti manji jer je moguće da čvor nema niti dvoje djece (što se dogodi kada čvor koji se otvara sam sadrži dva ili manje atributa) te zato što se isti skup atributa pojavljuje više puta unutar istog sloja (primjerice skup atributa 0101 može se dobiti na dva načina – iz skupa 0111 i 1101).

Algoritam 3: Unazadno pretraživanje u širinu s ograničenim faktorom grananja

Limited BFS back($initSet$, $treeBank$, $parserprevRes$):
$beam \leftarrow initSet$
za $i = 1..depth$ **radi**
$sloj \leftarrow []$
za svaki $set \in beam$ **radi**
$djeca \leftarrow []$
za svaki $atribut \in set$ **radi**
$noviSet = set - atribut$
ako $\neg tested(noviSet)$ **onda**
$[trnS, tstS] \leftarrow CreateTestTrain(noviSet,\ treeBank)$
$model \leftarrow Train(parser,\ trnS)$
$parsed \leftarrow Parse(parser,\ tstS)$
$LAS_{noviSet} \leftarrow Eval(parsed, tstS)$
$prevsRes \leftarrow prevRes + (ID_{noviSet}, LAS_{noviSet})$
$djeca \leftarrow InsertSortedBy(djeca, noviSet, LAS_{noviSet})$
za $j = 1..min(sizeOf(djeca), maxBranch)$ **radi**
ako $djeca[j] \notin sloj$ **onda**
$sloj \leftarrow sloj + djeca[j]$
$beam \leftarrow sloj$
$return\ prevRes$

Pretraživanje po slojevima radi na sličan način kao i ograničeno pretraživanje u širinu. Optimizator prima skup skupova atributa, i za svaki od tih skupova uklanja i vraća atribute jedan po jedan, pritom ispitujući kvalitetu modela parsanja na skupu za testiranje. Razlika je u tome da se u sljedeću iteraciju ne prosljeđuje određen broj najbolje djece za svaki o čvorova već samo određen broj najbolje "djece" u skupu svih



Slika 3.1: Slika prikazuje postupak ograničenog pretraživanja u širinu za četiri atributa, uz maksimalni faktor grananja koji iznosi dva.

generiranih čvorova djece.

Algoritam 4 prikazuje pseudokod pretraživanja po slojevima unatrag. Jedina razlika u odnosu na algoritam 3 jest u tome što se ne pamte djeca svakog otvorenog čvora zasebno već se skupovi atributa izravno umeću u listu čvorova„djece“ u „generaciji“ te se u sljedeću iteraciju prenosi određen broj najbolje djece u generaciji.

Algoritam 4: Unazadno pretraživanje po slojevima

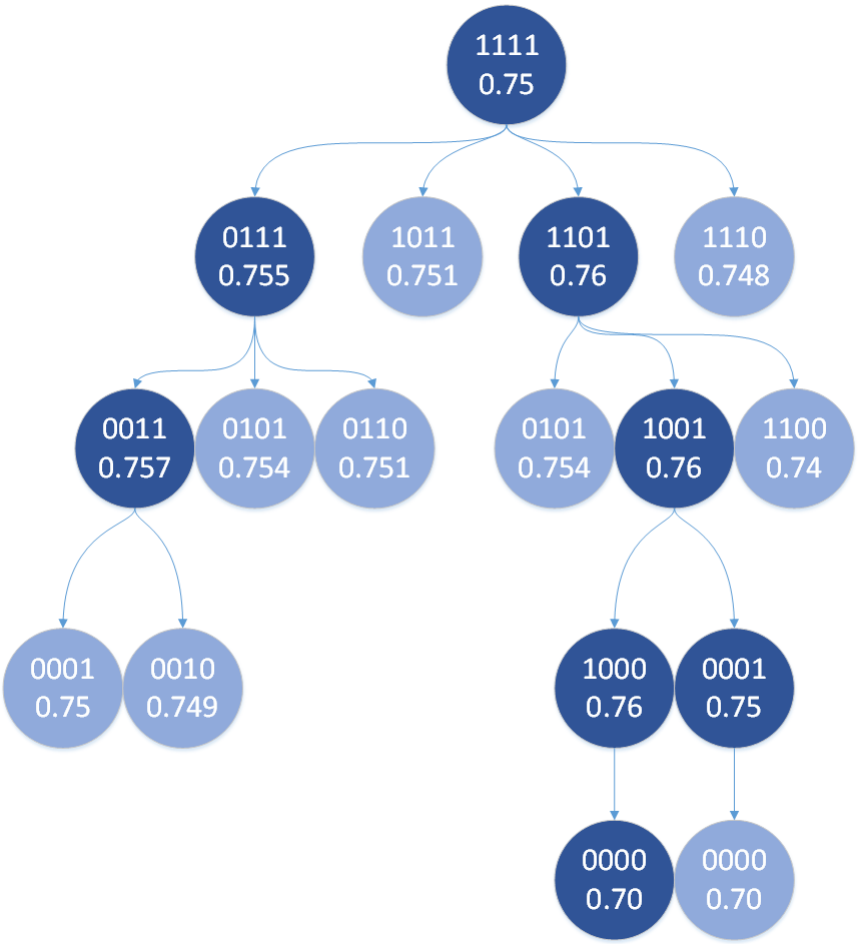
Beam back($initSet$, $treeBank$, $parserprevRes$):
  $beam \leftarrow initSet$
  za $i = 1..depth$ **radi**
    $sloj \leftarrow []$
    za svaki $set \in beam$ **radi**
      za svaki $atribut \in set$ **radi**
        $noviSet = set - atribut$
        ako $\neg tested(noviSet)$ **onda**
          $[trnS, tstS] \leftarrow CreateTestTrain(noviSet,\ treeBank)$
          $model \leftarrow Train(parser,\ trnS)$
          $parsed \leftarrow Parse(parser,\ tstS)$
          $LAS_{noviSet} \leftarrow Eval(parsed, tstS)$
          $prevsRes \leftarrow prevRes + (ID_{noviSet}, LAS_{noviSet})$
        ako $noviSetnijeusloj$ **onda**
          $sloj \leftarrow InsertSortedBy(sloj, noviSet, LAS_{noviSet})$
    $beam \leftarrow BestK(sloj, k)$
  $return\ prevRes$

Oba pristupa kao rezultat svog rada vraćaju rezultate kvalitete rada parsera za sve ispitane skupove atributa.

3.1.2. Definiranje skupa značajki

Govorili se o morfološkim atributima riječi moguće je da različite vrsti riječi dijele iste atribute (tako se primjerice u hrvatskom jeziku padež može definirati i za imenice i za pridjeve). Ovo znači da se atrbuti iz skupa atributa mogu uklanjati na dva načina: određeni se atribut može ukloniti u potpunosti (dakle za sve riječi za koje je definiran) ili se može ukloniti, ali samo za jednu od svih vrsta riječi za koje je definiran. U nastavku će se prvi način uklanjanja atributa nazivati skupnim uklanjanjem atributa, dok će se drugi način nazivati individualnim uklanjanjem atributa.

Osim morfoloških značajki, parseru se mogu na ulazu predati i riječi u rečenici te



Slika 3.2: Slika prikazuje postupak pretraživanja po slojevima za skup od četiri atributa uz veličinu prozora jednaku dva.

njihove leme. Uzme li se u obzir da broj različitih riječi i njihovih pojavnica može biti velik, njihovo korištenje može učiniti postupak učenja složenijim. Ispitivanje uklanjanja pojedinih riječi nema pretjeranog smisla,[2] no zanimljivo je promotriti što se događa ako su parseru poznate samo morfološke značajke riječi u rečenici, ali ne i same riječi koje su se pojavile u rečenici. Pristup parsanju koji ne koristi pojavnice ili njihove leme nazva se deleksikaliziranim parsanjem (engl. *delexicalized parsing*). Dok je deleksikalizirano parsanje zanimljivo samo po sebi zbog svojih mogućih primjena,[3] sa stajališta optimizacije skupa morfoloških značajki zanimljivo je jer može dati realniju procjenu kvalitete pojedinih morfoloških značajki. Naime, u takvoj deleksikaliziranoj okolini dobro će raditi samo korisne morfološke značajke, dok se u leksikaliziranoj varijanti može dogoditi da riječi nadoknade nedostatke pojedinih morfoloških značajki.

Konačno, značajke koje se predaju parseru mogu biti potpuno točne – dobivene ljudskim označavanjem – no mogu biti i dobivene strojnom analizom te sadržavati određenu pogrešku. U nastavku će se skup značajki dobiven ljudskim označavanjem nazivati zlatnim skupom značajki, dok će se skup značajki dobiven strojnom analizom nazivati sistemskim skupom značajki.[4]

Konačno, optimizator samostalno ne može dati odgovor koji je skup atributa najbolji (s obzirom da može uspoređivati samo performanse parsera na određenom skupu za testiranje) te se može dogoditi da dođe do prenaučenosti modela. Ne bi li se izbjegla prenaučenost, sustav omogućava dodatno testiranje potencijalno zanimljivih kombinacija atributa primjenom unakrsne provjere (engl. *cross-validation*) dakle provjere modela uz korištenje više različitih skupova za treniranje i testiranje.

3.2. Formati zapisa skupa ovisnosnih stabala

Format CoNLLx

Format zapisa CoNLLx definiran je za potrebe CoNLL (engl. *Conference on Computational Natural Language Learning*) natjecanja iz 2006 i 2007 godine (Buchholz i Marsi, 2006). Jedna rečenica u ovom formatu definirana je na sljedeći način: svaka

[2] Nema smisla ispitivati kako na rad parsera utječe ako se nikada ne preda pojavnica „jesu“.

[3] Model naučen na jednom setu rečenica izvučenih iz novinskih izvora, ne mora nužno dobro raditi na setu rečenica iz znanstvenih članaka. Pretpostavka je da se leksikoni ovih dviju skupova rečenica razlikuju, no da je gramatika konzistentna unutar jezika.

[4] Jedna od motivacija za izradu ovog reda jest smanjiti kompleksnost pretprocesiranja rečenica koje se parsaju - pretpostavka je da se smanjenjem skupa morfoloških atributa poboljšava automatska morfološko označavanje riječi, a time i smanjuje greška u podacima na ulazu u parser.

riječ rečenice napisana je u jednom retku, dok su rečenice odvojene praznim retkom. Svaka je riječ obilježena sljedećim podacima:

1. ID - redni broj riječi u rečenici koji predstavlja identifikator te riječi;
2. FORM - pojavni oblik riječi u rečenici;
3. LEMMA - lemom riječi;
4. CPOSTAG - jednostavna morfosintaktička oznaka (engl. *coarse part-of-speech tag*), primjerice vrsta riječi;
5. POSTAG - morfosintaktička oznaka (engl. *part-of-speech tag*), može biti jednaka prethodnoj, a može i sadržavati dodatnu informaciju definiranu standardom morfosintaktičkih oznaka za ulazni jezik;
6. FEATS - sintaktičke i semantičke značajke riječi definirane standardom za ulazni jezik u formatu „feat1=val1|feat2=val2 “;
7. HEAD - glava riječi unutar ovisnosnog stabla;
8. DEPREL - sintaktička funkcija riječi, odnosno labele veze između glave riječi i same riječi;
9. ... - dodatni parametri.

Primjer 3.1 prikazuje dvije rečenice zapisane u ovom formatu.

Primjer 3.1 Primjer zapisa rečenica u formatu CoNLLx

1	Proces	proces	N	N	nounT=c	0	Elp	_	_
2	privatizacije	privatizacija	N	N	nounT=c	1	Obj	_	_
3	na	na	S	S	_	1	Prep	_	_
4	Kosovu	Kosovo	N	N	nounT=p	3	Adv	_	_
5	pod	pod	S	S	_	0	Prep	_	_
6	povećalom	povećalo	N	N	nounT=c	5	Elp	_	_
1	Kosovo	Kosovo	N	N	nounT=p	3	Sb	_	_
2	ozbiljno	ozbiljno	R	R	advT=g	3	Adv	_	_
3	analizira	analizirati	V	V	verbT=m	0	Pred	_	_
4	proces	proces	N	N	nounT=c	3	Obj	_	_
5	privatizacije	privatizacija	N	N	nounT=c	4	Atr	_	_
6	u	u	S	S	_	3	Prep	_	_
7	svjetlu	svjetlo	N	N	nounT=c	6	Obj	_	_
8	učestalih	učestao	A	A	adjT=g	Atr	_	_	
9	pritužbi	pritužba	N	N	nounT=c	7	Atr	_	_
10	.	.	Z	Z	_	0	Punc	_	_

Format MST

Format MST predstavlja format koji se koristi isključivo kod parsera MST i opisan je u uputstvima za korištenje parsera. U ovom se formatu svaka rečenica proteže kroz četiri retka, a same su rečenice međusobno odvojene praznim retkom. Ta četiri retka sadržavaju:

1. Samu rečenicu u kojoj je svaka pojavnica odvojena razmakom;
2. POS oznake za svaku riječ u rečenici odvojeni razmakom, pri čemu redoslijed tagova odgovara redoslijedu pripadajućih pojavnica u retku 1;
3. Sintaktičku funkcije riječi u rečenici (dakle labelu veze između riječi i njene glave) odvojene razmakom; i ovdje je bitno da redoslijed labela odgovara redoslijedu pripadajućih riječi u rečenici; može biti oznaka vrste riječi, a može i

sadržavati dodatnu informaciju definiranu standardom morfosintaktičkih oznaka za ulazni jezik;

4. Identifikatore glave pripadajuće riječi u rečenici odvojene razmakom; identifikator odgovara rednom broju riječi u rečenici brojeći od 1, dok korijen ima oznaku 0.

Primjer 3.2 Zapis rečenica u formatu MST

```
Proces privatizacije na Kosovu pod povećalom
N N S N S N
Elp Obj Prep Adv Prep Elp
0 1 1 3 0 5

Kosovo ozbiljno analizira proces privatizacije u svjetlu učestalih pritužbi .
N R V N N S N A N Z
Sb Adv Pred Obj Atr Prep Obj Atr Atr Punc
3 3 0 3 4 3 6 9 7 0
```

Format CoNLL

Format CoNLL predstavlja nadogradnju CoNLLx formata korištenu za potrebe CoNLL natjecanja iz 2009. godine (Hajič et al., 2009) uz napomenu da je zapravo namijenjen ulančavanju više faza analize teksta – od lematizacije, preko morfološke analize do samog ovisnosnog parsanja. Sadrži sljedeće stupce:

1. ID - redni broj riječi u rečenici koji predstavlja identifikator te riječi;
2. FORM - pojavni oblik riječi u rečenici;
3. LEMMA - lemu riječi;
4. PLEMMA - predviđenu lemu riječi;
5. POSTAG - morfosintaktičku oznaku (engl. *part-of-speech tag*),može biti oznaka vrste riječi, a može i sadržavati dodatnu informaciju definiranu standardom morfosintaktičkih oznaka za ulazni jezik;
6. PPOSTAG - predviđenu morfosintaktička oznaka (engl. *part-of-speech tag*),

7. FEATS - sintaktičke i semantičke značajke riječi definirane standardom za ulazni jezik u formatu „feat1=val1|feat2=val2“;

8. PFEATS -predfiđene sintaktičke i semantičke značajke riječi definirane standardom za ulazni jezik u formatu „feat1=val1|feat2=val2“;

9. HEAD - glava riječi unutar ovisnosnog stabla;

10. PHEAD - predviđena glava riječi unutar ovisnosnog stabla;

11. DEPREL - sintaktička funkcija riječi, odnosno labele veze između glave riječi i same riječi;

12. PDEPREL - predviđena sintaktička funkcija riječi, odnosno labele veze između glave riječi i same riječi;

13. ... - dodatni parametri.

Dakle, CoNLL format sadrži gotovo sve stupce kao i CoNLLx format, osim što iza svakog originalnog ostavlja stupac u koji se upisuje rezultat rada lematizatora, POS, odnosno morfološkog označivača te ovisnosnog parsera (P-stupci).

3.3. Implementacija optimizatora

Optimizator je implementiran u programskom jeziku Java. Sastoji se od dvije ključne klase: ***Utility*** i ***Experiments***.

3.3.1. Klasa Utility

Klasa ***Utility*** sadrži funkcije za rad s različitim formatima zapisa ovisnosnih rečenica te omogućuje delematizaciju banke stabala. Sadrži strukture podataka i metode navedene u nastavku.

public static List> sentFeatsV - Lista rečenica, pri čemu je svaka rečenica lista riječi, a svaka je riječ definirana svojim morfološkim atributima. Vektor znakovnih nizova predstavlja vektor svih morfoloških atributa neke riječi. Pozicija na koju se upisuje vrijednost atributa neke riječi funkcija je vrste riječi i vrste atributa (primjerice na prvom se mjestu nalazi vrijednost padeža imenice, na drugom vrijednost padeža pridjeva itd.).

public static List> sentFeatsC - Lista rečenica, pri čemu je svaka rečenica lista riječi, a svaka je riječ definirana atributima koji nisu morfološki (oblik riječi u rečenici, njezina lema itd.). Zajedno sa `sentFeatsV` čini potpuni opis neke banke stabala.

public static Map featsV - preslikava kombinaciju riječi i morfoloških atributa na poziciju u vektoru atributa (onog koji se nalazi u `sentFeatsV`).
public static Map featsVComb - preslikava morfološki atribut na poziciju u vektoru atributa. Koristi se prilikom skupnog uklanjanja atributa.
public static Map featsC - preslikava nemorfološke atribute na poziciju u vektoru nemorfoloških značajki (vektoru koji se koristi u `sentFeatsC`).
public static Map deprels - preslikava moguće sintaktičke funkcije (oznake veza) u poziciju u vektoru.

static void ReadFeats(String path) - učitava postojeće morfološke atribute i vrste riječi za koje mogu biti definirani te stvara `featsV` i `featsVComb`.
static void ReadConllxSentences(String path) - učitava banku stabala u CoNLLx formatu nad kojom će se raditi eksperimenti i pohranjuje ju u `sentFeatsV`, odnosno `sentFeatsC`. Ignorira POS oznaku.
static Experimens ReadExperimentParams(String path, String experiment) - učitavaju se parametri eksperimenta ovisno o vrsti eksperimenta.
static void ReadDeprel(String path) - učitavaju se postojeće sintaktičke funkcije.
static void ReadPreviousResults(String path) - učitavaju se rezultati prethodnih eksperimenata tako da se parser ne bi više puta trenirao isti model.

static void Delemmatize() - uklanja lemu i riječ iz `sentFeatsC`.
static void Delemmatize(List> sfc) uklanja lemu i riječ iz `sfc`.
static void ChangeType(String path, string inType, String outType, Boolean delemmatize) - učitava ovisnosna stabla koja se nalaze u datoteci `path` koja je u formatu `inType`, te ju zapisuje opet u `path`, ali u formatu `outType`, te ukoliko treba delematizira učitana ovisnosna stabla.
static void ReadConllx(String p, List> sfv, List> sfc) - učitava ovisnosna stabla iz verb|p| (koji mora biti u CoNLLx formatu) te učitane rečenice pohranjuje u `sfv` (morfološki atributi) i `sfc` (svi ostali atributi). Riječi i rečenice su pohranjene istim redoslijedom u oba skupa.
static void ReadConll(String p, List> sfv, List> sfc) - učitava ovisnosna stabla iz `p` (koji mora biti u Conll formatu) te učitane rečenice pohranjuje u `sfv` i `sfc`. Vrijed-

nosti se učitavaju iz P-stupaca.

static void ReadMST(String p, List> sfv, List> sfc) - učitava ovisnosna stabla iz `p` (koji mora biti u MST formatu) te učitane rečenice pohranjuje u `sfv` i `sfc`. Pretpostavlja da je oznaka u retku s POS oznakama jednaka formatu oznake u FEATS stupcu formata CoNLLx.

static void WriteToConll(String p, List> sfv, List> sfc) - zapisuje ovisnosna stabla iz `sfv` i `sfc` u `p` u CoNLL formatu. `sfv` treba sadržavati morfološke, a `sfc` sve ostale atribute te poredak riječi i rečenica u oba skupa mora biti jednak. Zlatni i P-stupci sadržavaju iste vrijednosti.

static void WriteToConllx(String p, List> sfv, List> sfc) - zapisuje ovisnosna stabla iz `sfv` i `sfc` u `p` u CoNLLx formatu analogno prethodnoj funkciji. Vrijednost POS stupca jednaka je CPOS stupcu.

static void WriteToMST(String p, List> sfv, List> sfc) - zapisuje ovisnosna stabla iz `sfv` i `sfc` u `p` u MST formatu. S obzirom da format MST nema FEATS redak, u redak s POS oznakom smješta se FEATS tag u obliku u kojem se upisuje u FEATS stupcu u CoNLL formatu.

static void Write(Integer f, Integer l, String[] sf, BufferedWriter writer, String type) - zapisuje rečenice pohranjene u `sentFeatsV` i `sentFeatsC` u `writer`. Ne zapisuje cijelu banku stabala već samo rečenice između `f` i `l`. `sf` predstavlja skup odabranih odabranih atributa – zapravo je to binarni vektor koji ima vrijednost "1" na pozicijama atributa koji će se zapisati u modificiranu varijantu skupa stabala. Pretpostavlja se da je dužina ovog vektora jednaka dužini vektora u `sentFeatsV`. Koristi se za stvaranje skupova za testiranje i treniranje s modificiranim skupom atributa koji se koriste prilikom rada optimizatora. ovisno o tipu skupa za treniranje, odnosno testiranje, zove s neka od sljedećih funkcija.

static void WriteToConll(Integer f, Integer l, String[] sf, BufferedWriter writer) - zapisuje rečenice pohranjene u `sentFeatsV` i `sentFeatsC` u `writer` u formatu CoNLL. P stupac ima iste vrijednosti kao i stupac sa zlatnim vrijednostima.

static void WriteToConllx(Integer f, Integer l, String[] sf, BufferedWriter writer) - zapisuje rečenice pohranjene u `sentFeatsV` i `sentFeatsC` u `writer` u formatu CoNLLx. POS i CPOS oznaka su jednake.

static void WriteToMST(Integer f, Integer l, String[] sf, BufferedWriter writer) - zapisuje rečenice pohranjene u `sentFeatsV` i `sentFeatsC` u `writer` u formatu MST. U redak s POS oznakom smješta se FEATS tag u obliku u kojem se upisuje u FEATS stupcu u CoNLL formatu.

static void WriteStats(Experiment exp) - zapisuje rezultate rada u datoteku koja je definirana kao jedan od parametara eksperimenta.

3.3.2. Klasa Experiments

Klasa ***Experiments*** sadrži strukture podataka i funkcije koje implementiraju funkcionalnosti optimizatora opisane u prethodnom potpoglavlju.
public String trainPath, testPath, outPath, tmpFile, resultsPath - pohranjuju puteve do datoteka koje će se koristiti za treniranje i testiranje rada parsera, odnosno u koju će se pohraniti rezultati rada optimizatora.
public String fileType, morphType - definiraju format zapisa ovisnosnih stabala s kojima rade parser odnosno morfološki označivač.
public String trainCmd, testCmd, trainMorphCmd, testMorphCmd, projCmd, deprojCmd - definiraju naredbe za treniranje i testiranje parsera, treniranje i testiranje morfološkog označivača, te projektivizaciju i deprojektivizaciju seta ovisnosnih stabala ukoliko se radi pseudoprojektivno parsanje.
public String selectedFeatures - predstavlja inicijalni skup odabranih morfoloških atributa od kojih kreće rad parsera.
public String featsModType - definira dodaju li se ili uklanjaju atributi iz inicijalnog skupa atributa.
public Integer NUMBEST, MAXTEST,TESTSIZE, depth - definiraju broj najbolje djece koja se prenose u sljedeću iteraciju, maksimalan broj testova koji se može izvesti, veličinu skupa za testiranje (ostatak se koristi za treniranje) i dubinu pretraživanja (maksimalan broj iteracija koji se može izvesti).
public static Map visited, visitedUas, visitedLa - pohranjuju vrijednosti LAS, UAS i LA za skup atributa.

public void Beam() - implementira pretraživanje po slojevima.
public void BWS() - implementira pretraživanje u širinu s ograničenim faktorom grananja.
public void CrossValidate() - vrši unakrsnu provjeru za neki skup atributa.
public void TestPipe() - vrši unakrsnu provjeru za neki skup atributa, ali ulančava morfološku analizu i testiranje rada parsera. Ovo znači da se za parsanje testnih primjera ne koriste zlatne, već sistemske morfološke oznake. Morfološka analiza radi se korištenjem zlatnih POS oznaka i zlatnih lema. Treniranje parsera i morfološkog označivača

radi se na zlatnim oznakama.

private void ExecuteCommand(String cmd) - izvršava naredbu `cmd` (primjerice naredbu za treniranje parsera) i čeka njezin završetak..
public static Hash(String[] featsVector - računa jedinstveni ključ za odabrani skup atributa koji se može iskoristiti za pohranu vrijednost LAS, UAS i LA.
public String[] ModifyComb (String[] featsVector) - ukoliko se pretraživanje skupa značajki vrši uz skupnu modifikaciju atributa, onda je prije zapisivanja tako modificiranog skupa atributa potrebno transformirati vektor odabranih atributa u vektor odabranih atributa koji se koristi prilikom individualnog načina modifikacije skupa atributa i prilikom zapisivanja u datoteku u klasi `Utility` (novonastali vektor mora biti jednake dužine kao i vektor koji se koristi u `sentFeatsV`).

public void prepareTestTrain(String[] featsVector) - stvara skup za treniranje i testiranje, pri čemu za testiranje korist prvih `TESTSIZE` rečenica iz `sentFeatsC` i `sentFeatsV`.
public void prepareTestTrain(String[] featsVector, Integer beg, Integer end) - stvara skup za treniranje i testiranje, pri čemu za testiranje koristi rečenice iz `sentFeatsC` i `sentFeatsV` između `beg` i `end`.
public void prepareTestTrain(String[] featsVector, Integer beg, Integer end, String type) - stvara skup za treniranje i testiranje, pri čemu za testiranje koristi rečenice iz `sentFeatsC` i `sentFeatsV` između `beg` i `end`, s tim da ova dva skupa zapisuje u formatu `type`.

public void Eval(String featsVectorHash) - uspoređuje `testFile` i `outFile` te računa LAS, UAS i LA. Poziva neku od sljedećih funkcija:
public void EvalConll(String featsVectorHash) - uspoređuje `testFile` i `outFile` koji su u formatu CoNLL;
public void EvalConllx(String featsVectorHash) - uspoređuje `testFile` i `outFile` koji su u formatu CoNLLx;
public void EvalMSt(String featsVectorHash) - uspoređuje `testFile` i `outFile` - uspoređuje `testFile` i `outFile` koji su u formatu MST.

public void Confusion() - ukoliko se radi unakrsna provjera, tada ispunjava matricu zabune, i pritom ovisno o formatu zapisa s kojim radi parser zove neku od sljedećih funkcija:

public void ConfusionConll() - uspoređuje `testFile` i `outFile` koji su u formatu CoNLL;
public void ConfusionConllx() - uspoređuje `testFile` i `outFile` koji su u formatu CoNLLx;
public void ConfusionMST()- uspoređuje `testFile` i `outFile` koji su u formatu MST;

3.4. Korištenje optimizatora

Optimizator se nalazi u arhivi DepParsOpt.jar. Poziva se naredbom:

java -cp DepParsOpt.jar Program Experiment ExpConfig Feats Deprels DTBank Type PrevResults

Experiment predstavlja tip eksperimenta koji se želi izvesti i može poprimiti jednu od 4 vrijednosti:

- `beam` - izvodi pretraživanje po slojevima;
- `BWS` - izvodi pretraživanje u širinu s ograničenim faktorom grananja;
- `cross` - izvodi se unakrsna provjera;
- `pipe` - izvodi se unakrsna provjera, s tim da se trenira i morfološki označivač.

`ExpConfig` predstavlja put do datoteke s parametrima eksperimenta i mora biti u sljedećem obliku navedenom u primjeru 3.3.

`Feats` predstavlja put do datoteke s postojećim atributima i vrstom riječi za koju može biz definiran u formatu „feat:POS1|POS2...“. Primjer 3.4 prikazuje primjer dijela `Feats` datoteke za MSTv4 skup morfoloških oznaka.

`Deprels` predstavlja put do datoteke s navedenim mogućim sintaktičkim funkcijama u ovisnosnoj banci stabala pri čemu se svaka moguća oznaka veze navodi u novom redu.

Primjer 3.3 Format datoteke ExpConfig

1. format zapisa ovisnosnih stabala s kojima radi parser koji će se koristiti (`mst`, `conll` ili `conllx`)
2. naredba za treniranje parsera
3. naredba za testiranje parsera
4. oznaka treba li projektivizirati stabla prije učenja, odnosno deprojektivizirati nakon parsanja (`proj` ako treba, `nonproj` inače)
(4.a ako `proj`, naredba za projektivizaciju skupa za treniranje)
(4.b ako `proj`, naredba za deprojektivizaciju skupa za testiranje)
(5.a ako je `Experiment` jednak `pipe` tada naredba za treniranje morfološkog označivača)
(5.b ako je`Experiment` jednak `pipe` tada naredba za morfološko označavanje skupa za testiranje)
(5.c ako je `Experiment` jednak `pipe` tada format zapisa ovisnosnih stabala s kojima radi morfološki označivač)
(5.d ako je `Experiment` jednak `pipe` tada put do datoteke u koju će se pohraniti izlaz morfološkog označivača)
6. put do datoteke koja sadrži skup za treniranje
7. put do datoteke koja sadrži skup za testiranje
8. put do datoteke u koju će se pohraniti izlaz rada parsera na skupu za testiranje
9. put do datoteke u koju će se upisati rad optimizatora 10. inicijalni skup atributa odvojen s „|“(ključna riječ `all` ukoliko se želi uvrstiti sve atribute)
11. tip izbacivanja atributa (`comb` za skupno izbacivanje atributa, `ind` inače; također utječe na zapis inicijalnog skupa atributa: za `comb` se navodi samo naziv atributa, za `ind` se navodi vrsta riječi i ime atributa u obliku „POS::feat“)
12. maksimalni faktor grananja (`Experiment` jednak `BWS`) / broj čvorova djece koji se prenosi u sljedeći sloj (`Experiment` jednak `beam`)
13. maksimalan broj testova koji se može izvesti
14. veličina skupa za testiranje (ostatak banke stabala koristi se za treniranje)
15. dubina pretraživanja (`Experiment` jednak `BWS`) / broj iteracija (`Experiment` jednak `beam`) / broj preklopa (`Experiment` jednak `cross` ili `Experiment` jednak `pipe`)
(16. `add` ako se u inicijalni skup trebaju dodavati atributi, `rmv` ako se iz inicijalnog skupa trebaju uklanjati atributi.)

Primjer 3.4 Format datoteke Feats

nounT:N
gen:N|V|A|P|M
num:N|V|A|P|M
cs:N|A|P|S|M
anim:N|A|P|M
verbT:V
...

`DTBank` je put do datoteke s bankom ovisnosnih stabala u CoNLLx formatu. Treba napomenuti da će optimizator prilikom rada zanemariti CPOS stupac, odnosno na to mjesto staviti POS oznaku, te pretpostavlja da se morfološki atributi nalaze u FEATS stupcu (ne analizira CPOS oznaku). `Type` predstavlja zastavicu koja označava redi li se o deleksikaliziranom parsanju (oznaka `delex`) ili ne (oznaka `lex`). `PrevResults` označava put do datoteke u kojoj su pohranjeni rezultati nastali prilikom prethodnih pokretanja optimizatora. Primjerice, želi li se optimizator pokrenuti više puta s različitim početnim skupom atributa, dogodi li se da u pretraživanju optimizator naiđe na prethodno ispitivane modele, nema smisla da se iznova ponavlja treniranje parsera (vremenski najzahtjevniji dio rada optimizatora). Pritom treba paziti da su rezultati doista usporedivi.[5]

Kao rezultat rada optimizator stvara datoteku u koju za sve ispitivane skupove atributa redak po redak navodi: korištene atribute te dobiveni LAS, UAS i LA. Na kraju datoteke ispisuje se matrica zabune za sintaktičke funkcije i to tako da se najprije navede sintaktička funkcija, a zatim izlaz koji je dao parser. Ova se matrica računa samo ako je tip eksperimenta `pipe` ili `cross`.

[5] U $ExpConfig$ moraju biti ista podjela seta na skup za testiranje i treniranje, mora se koristiti isti poziv za treniranje i testiranje parsera, ne mogu se uspoređivati lematizirani i delematizirani model itd.

4. Testiranje

Opisani optimizator upotrebljen je ne bi li se pronašao optimalan skup morfoloških značajki za hrvatski jezik. U nastavku su detaljno opisani parseri koji su korišteni tijekom ispitivanja te svi parametre koje je za njih moguće podešavati, a da nemaju veze s morfološkim značajkama. Nakon toga je opisana korištena ovisnosna banka stabala za hrvatski jezik, ali i banke stabala za češki i slovenski jezik koje su poslužile da bi se dobiveni model ispitao na hrvatskom srodnim jezicima. Na kraju je opisan tijek samog testiranja i izneseni korišteni parametri parsera.

4.1. Parseri

Za testiranje rada optimizatora odnosno optimizaciju skupa morfoloških značajki za hrvatski jezik korišteni su parser Malt, parser MST te parser MATE koji dolazi u dvije varijante – prijelazničkoj i varijanti temeljenoj na grafovima.

4.1.1. Malt parser

Malt parser[1] predstavlja parser koji je temeljen na prijelazima. Trenutno korisniku nudi mogućnost odabira između 9 različitih deterministčkih algoritama parsanja (Covington; Nivre et al., 2006; Nivre, 2009): `Nivre arc-eager`, `Nivre arc-standard`, `Covington projective`, `Planar` i `2-planar` za projektivno parsanje te `Covington non-projective`, `Stack projective`, `Stack swap-eager` i `Stack swap-lazy` za neprojektivno parsanje.

Dodatno omogućava i pseudoprojektivno parsanje(Nivre i Nilsson, 2005). Pseudoprojektivna stabla jesu projektivna stabla nastala iz neprojektivnih postupkom projektivizacije. Ovaj postupak omogućava da se za parsanje neprojektivnih stabala koriste algoritmi za projektivno parsanja, a u praksi pokazuje bolje rezultate nego neprojektivno parsanje. Postupak učenja i parsanja tada se vrši na sljedeći način:

[1]Dostupan na adresi `http://www.maltparser.org/`

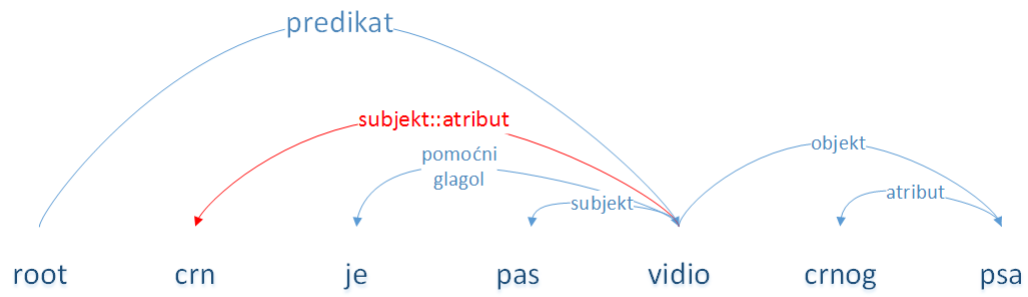
1. Stabla iz skupa za učenje se projektiviziraju;
2. Model se uči na projektiviziranim stablima iz skupa za učenje;
3. Parser radi s modelom naučenim na projektiviziranim stablima i generira projektivno stablo;
4. Primjenom inverza funkcije projektivizacije dobiveno projektivno stablo transformira se u neprojektivno stablo.

Projektivizacija se provodi na sljedeći način: u neprojektivnom se stablu detektira neprojektivni brid (i, r, j). Taj se neprojektivni brid briše, a u graf se dodaje novi brid $(h(i), r', j)$, gdje je $h(i)$ funkcija koja za primljeni čvor vraća njegovu glavu, a r' nova labela - stvara se nova veza u kojoj se glava ovisne riječi postaje riječ koja joj je posredno nadređena. Nova labela koja povezuje $h(i)$ i j u sebi nosi informaciju o vezi između i i $h(i)$, kao i informaciju o originalnoj vezi r i omogućava da se postupak invertira. Slika 4.1 prikazuje ovaj postupak.

Primjer 4.1 prikazuje zapis neprojektivnog i projektiviziranog stabla za rečenicu „Crn je pas vidio crnog psa“.

Primjer 4.1 Projektivizacija ovisnosnog stabla

$s = "Crn\ je\ pas\ vidio\ crnog\ psa"$
$V = \{crn, pas, je, vidio, crnog, psa\}$
$A = \{(pas, atribut, crn), (vidio, subjekt, pas), (vidi, objekt, psa),$
$(vidio, pom.\ glagol, je)(psa, atribut, crnog), (root, predikat, vidio)\}$
$A' = \{(pas, subjekt :: atribut, crn), (vidio, subjekt, pas), (vidi, objekt, psa),$
$(vidio, pom.\ glagol, je)(psa, atribut, crnog), (root, predikat, vidio)\}$



Slika 4.1: Primjer projektiviziranog ovisnosnog stabla za rečenicu „Crn je pas vidio crnog psa“. Crvenom je bojom označena veza koja je prehodno bila neprojektivna.

Primjer 4.2 primjer *.par* datoteke

```
POS STACK
POS INPUT
POS INPUT 1
POS INPUT 2
POS INPUT 3
POS INPUT 4
POS STACK 1
POS INPUT 0 -1
CPOS STACK
CPOS INPUT
CPOS INPUT 0 -1
CPOS STACK 1 0 1
DEP STACK
DEP STACK 0 0 0 -1
DEP STACK 0 0 0 1
DEP INPUT 0 0 0 -1
LEX STACK
LEX INPUT
LEX INPUT 1
LEX STACK 0 0 1
FEATS STACK
FEATS INPUT
FEATS INPUT 1
FEATS INPUT 2
LEMMA STACK
LEMMA INPUT
LEMMA STACK 0 -1
```

Malt parser omogućava nekoliko različitih strategija projektivizacije koje se razlikuju po načinu bilježenja projektivizirane veze.

S obzirom da je nemoguće koristiti potpunu informaciju o konfiguraciji stanja,[2] Malt parser za opis stanja koristi samo neki podskup značajki ulaznog niza, kao i dotad izgrađenog stabla. Može se reći da se stanje opisuje korištenjem dvije funkcije: adresne funkcije $a(c)$ koja odabire riječi koje su relevantne za opis trenutnog stanje (primjerice, riječ s vrha stoga, riječ koja je trenutno na ulazu u automat i sl.) i funkcije $f(a(c))$ koja zatim vadi značajke odabranog podskupa riječi. Značajke koje se mogu definirati za niz riječi tada mogu biti položaj riječi u rečenice i, udaljenost među tim

[2]Broj mogućih stanja koje može poprimiti parser je beskonačan jer je broj različitih rečenica koje se mogu pojaviti na ulazu u parser također beskonačan

riječima, oblik riječi (npr. rod, broj i padež), lema riječi, ima li riječ glavu ili je i sama glava itd. Ove značajke definira korisnik unutar *.par* datoteke koju zatim predaje parseru prilikom pokretanja. Primjer 4.2 prikazuje primjer izgleda *.par* datoteke, dok je definiciju jezika zadavanja željenih značajki moguće moguće pronaći na stranicama Malt parsera.

Samo proročište kod Malt parsera koje na temelju značajki stanja odabire prijelaz implementirano je kao klasifikator kojem su moguće klase definirani prijelazi. Ovaj klasifikator moguće je učiti na dva načina: korištenjem biblioteke LIBSVM (Chang i Lin, 2011) ili biblioteke LIBLINEAR (Fan et al., 2008), te je za svaku od ovih opcija moguće definirati dodatne parametre.

Malt parser radi vrši učenje i parsanje nad ovisnosnim stablima zapisanim u CoNLLx formatu. Određene parametre Malt parsera moguće je optimizirati korištenjem alata MaltOptimizer [3] (Ballesteros i Nivre, 2012) kojim je moguće odabrati optimalan algoritam parsanja te definirati optimalan skup značajki stanja.

4.1.2. mst parser

Parser MST[4] predstavlja parser temeljen na grafovima te implementira dvije različite varijante parsanja ovisno o tome radi li s neprojektivnim ili projektivnim stablima. Ukoliko radi s neprojektivnim stablima tada radi s Chu-Liu-Edmondsovim algoritmom, a radi li s projektivnim stablima tada se izvodi Eisenrov algoritam (Eisner, 1996; McDonald et al., 2005). Naime, projektivna ovisnosna stabla mogu se promatrati kao ugnježđena manja ovisnosna stabla, te se može povući paralela između projektivnog ovisnosnog parsanja i parsanja kontekstno neovisnim gramatikama. Ovo znači da se u projektivnom ovisnosnom parsanju mogu iskristiti postojeći alogritmi za parsanje ovisnosnim gramatikama. Jedan od tih algoritama jest algoritam CYK (Sakai, 1962; Younger, 1967; Kasami i Torii, 1969), dok Eisnerov algoritam predstavlja vremensku i memorijsku optimizaciju CYK algoritma koje se temelji na specifičnostima projektivnih ovisnosnih veza.

[3] Dostupan na adresi `nil.fdi.ucm.es/maltoptimizer`.

[4] Dostupan na adresi `http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html`.

Eisnerov algoritam

Algoritam 5: Pseudokod algoritma CYK

$s = (w_1..w_n), w_i \in \Sigma, 1 \leq i \leq n$

$\lambda_{w_i,w_j} \in \lambda$

$E[s, s, d, c] = 0,\ s = 1..n, d = 1..2, c = 1..2$

za *m = 1..n* **radi**

  za *s = 1..n* **radi**

    $t = s + m$

    ako $t > n$ **onda**

      $break$

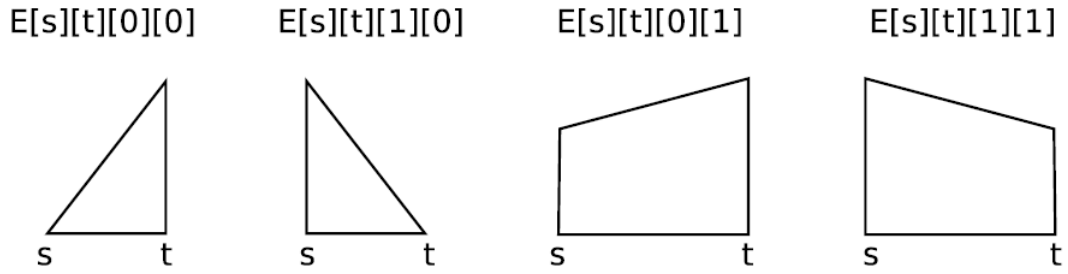
    $E[s, t, 0, 1] = max_{s \leq q < t}(E[s, q, 1, 0] + E[q + 1, t, 0, 0] + \lambda_{w_t,w_s})$

    $E[s, t, 1, 1] = max_{s \leq q < t}(E[s, q, 1, 0] + E[q + 1, t, 0, 0] + \lambda_{w_s,w_t})$

    $E[s, t, 0, 0] = max_{s \leq q < t}(E[s, q, 0, 0] + E[q + 1, t, 0, 1])$

    $E[s, t, 1, 0] = max_{s < q \leq t}(E[s, q, 1, 1] + E[q + 1, t, 1, 0])$

Algoritam zapravo za svaki mogući podniz $(w_s, ..w_t)$ neke rečenice $(w_1, .., w_n)$ pronalazi najbolje stablo kojemu je korijen[5] ili w_s (za $d = 1$) ili w_t(za $d = 0$). Dodatna opcija je da podređena riječ može i sama biti korijen nekog kraćeg podstabla ($c = 1$) ili ne ($c = 0$). Mogući podgrafovi koje koristi Eisnerov algoritam prikazani su na slici 4.2.Ocjena najboljeg ovisnosnog stabla nalazi se u $P[0, n, 1, 0]$.



Slika 4.2: Prikaz podstabala koja se koriste tijekom rada Edmondsovog algoritma.

Već je rečeno kog algoritma CYK, a može vidjeti i kod Eisnerovog algoritma, da bi se moglo pronaći maksimalno razapinjuće stablo potrebno je na neki način ocijeniti kvalitetu veze između dvije riječi, a zatim i dodijeliti oznaku toj vezi (McDonald et al., 2006). Općenito je moguće reći da je bodovanje grafa moguće prema sljedećoj formuli:

$$score(G) = f(\psi_1, .., \psi_n),\ \forall \psi_i \in \Psi_G$$

[5]Pojam "korijen" ovdje se koristi kao bilo koji čvor stabla koji još nema glavu, ne u smislu umjetno dodane riječi koja čini korijen svakog ovisnosnog stabla

Dakle, ocjena grafa jest neka funkcija svih njegovih podgrafova. Prethodni se izraz može pojednostavniti tako da se ocjena pojedinog grafa definira kao:

$$score(G) = \sum_{\psi_i \in \Psi_G} \lambda_\psi$$

gdje je λ_ψ ocjena svakog mogućeg podgrafa grafa G.

Ono što je još potrebno razriješiti jest što čini Ψ_G, i kako definirati λ_{ψ_i}.Kod MST parsera pretpostavlja se da Ψ_G čine bridovi, odnosno da se ocjenjuje samo kvaliteta veza između pojedinih parova riječi. Tada se ocjena pojedinog brida računa na temelju značajki incidentnih vrhova (npr. labele, položaja i svojstava incidentnih riječi):

$$\lambda_\psi = score(w_i, w_j) = w \cdot f(i, j)$$

Prethodno opisana varijanta Eisnerovog i Chu-Liu-Edmondsovog algoritma pretpostavlja da se prilikom bodovanja koriste samo značajke incidentnih vrhova – tzv. model prvog reda (engl. *first-order features*). MST parser omogućava i korištenje modela drugog reda koje bodovanje stabla proširuju na sljedeći način:

$$\lambda_\psi = score(w_i, w_j, w_k) = w \cdot f(i, j, k)$$

gdje su w_i i w_k susjedna djeca koja se nalaze s iste strane glave w_i. Model drugog reda moguće je koristiti prilikom oznalavanja veza pro čemu se označavanje sintaktičke funkcije djece neke glave može promatrati kao označavanje niza za što se onda može iskoristiti neki od poznatih algoritama za labeliranje nizova (primjerice Viterbijev algoritam).

Eisnerov algoritam moguće je modificirati tako da prilikom pronalaska najboljeg nelabeliranog ovisnosnog stabla koristi model drugog reda, dok bi Chu-Liu-Edmondsov algoritam postao NP-težak (Mcdonald i Pereira, 2006).

Učenje na temelju promjera MST parser radi korištenjem algoritma MIRA (Crammer i Singer, 2003; Crammer et al., 2006; **?**) koji mijenja vektor težina iz iteracije u iteraciju na temelju svake rečenice zasebno i to tako da minimizira razliku razliku između vektora iz prethodne i trenutne iteracije uz uvjet da razlika između bodova grafa za treniranje i bodova dobivenog grafa bude proporcionalna broju riječi kojima je netočno dodijeljena glava.

$$min\|w^{(i+1)} - w^i\|, s(x_t, y_t; w^{(i+1)}) - s(x_t, y_p; w^{(i+1)}) \geq L(y_t, y_p)$$

gdje je x_t ulazna rečenica, y_t točno ovisnosno stablo, y_p stablo dobiveno na izlazu, w^i trenutni vektor težina, $w^{(i+1)}$ novi vektor težina, a $L(y_t, y_p)$ gubitak dobivenog u

odnosu na dobiveno stablo. Konačno stablo dobije se kao prosječna vrijednost zbroja vektora težina kroz iteracije, čime se izbjegava prenaučenost parsera (Collins, 2002). Kao dodatnu opciju, MST parser dopušta proširenje uvjeta minimizacije tako da uključuju k najboljih stabala koje je dao parser te broj epoha (prolazaka kroz cijeli skup za treniranje).

MST parser za i učenje zahtjeva skupove ovisnosnih stabala zapisane u formatu MST.

4.1.3. MATEparser

Parser MATE[6] najnoviji je od korištenih parsera te predstavlja nadogradnju na parsere Malt i MST. Dolazi u dvije varijante – onoj temeljenoj na prijelazima i onoj temeljenoj na grafovima. U nastavku su opisane obje arijante parsera.

Graf varijanta

Graf varijanta parsera MATE u parsanju koristi Eisnerov algoritam parsanja modificaran da radi sa sintaktičkim modelima višeg reda (Carreras, 2007). Neprojektivno parsanje vrši se aproksimacijom i to tako da se najprije pronađe najbolje projektivno stablo za neku rečenicu, a zatim se dobiveno stablo transformira dok moguće transformacije donose poboljšanja u kvaliteti ovisnosnog stabla (Mcdonald i Pereira, 2006). Pritom je moguće definirati prag zahtijevanog poboljšanja u kvaliteti stabla prilikom dodavanja nove veze čime se ubrzava postupaka parsanja te poboljšava konačan rezultat parsanja (Bohnet, 2009).

Učenje vektora težina vrši se korištenjem perceptrona pasivno-agresivnog perceptrona koji mijenja vektor težina točno onoliko koliko je potrebno da bi se trenutni primjer točno klasificirao. Modifikacija težina kroz iteracije vrši se na sljedeći način:

$$\gamma = EI - ((n-1)I + i) + 2$$
$$u = (f(x_i, y_i) - f(x_i, y_p)$$
$$\eta = \frac{e - (F(x_i, y_i) - F(x_i, x_p))}{u^2}$$
$$w = w + \eta u$$
$$u = v + \gamma \eta u$$

[6]Obje inačice parsera MATE dostupne su na adresi `https://code.google.com/p/mate-tools/`

gdje je E broj epoha (prolazaka kroz cijeli skup za treniranje), n trenutna epoha, I broj primjera za učenje, i oznaka trenutnog primjera za učenje, $f(x_i, y_i) - f(x_i, y_p)$ razlika između vektora značajki točnog stabla za rečenicu $x_i$te vektora značajki stabla dobiveno trenutnom konfiguracijom parsera za rečenicux_i„ e broj pogrešnih labela u dobivenom stablu, w vektor težina a u zbroj dotad napravljenih promjena vektora težina. Pritom se promjene vektoru u pribrajaju s faktorom γ, koji se kroz iteracije smanjuje. Konačni vektor predstavlja prosječnu vrijednost vektora u čime se smanjuje prenaučenost.

Algoritam koji koristi MATE parser (Bohnet, 2010) prvenstveno je razvijen ne bi li se napravila vremenska i prostorna optimizacija rada Carrerasovog algoritma. Osim paralelizacije postupka parsanja, najveća je ušteda postignuta redukcijom korištenih značajki, koja se provela korištenjem hash-funkcije. Veličina vektora težina nije jednaka veličini vektora značajki, već se za svaku značajku koristi hash funkcija koja za danu značajku vraća indeks pripadajuće težine. Veličinu vektora značajki može definirati sam korisnik prilikom pokretanja. Optimalna veličina vektora značajki treba biti takva da maksimizira broj elemenata vektora težina koji poprimaju vrijednost različitu od nule, te istovremeno minimizira broj kolizija. [7]

4.1.4. Prijelaznička varijanta parsera

Prijelaznička varijanta temelji se na Nivreovom algoritmu za neprojektivno parsanje. Ovaj model nadograđen je tako da umjesto determinizma (i pohlepnog odabira najboljeg prijelaza za neko stanje) pretražuje prostor mogućih stanja parsera (Bohnet i Kuhn, 2012). S obzirom da je prostor stanja prevelik, ovo je pretraživanje ograničeno – uvijek se proširuje samo određen broj najboljih stanja iz prethodnog koraka (engl. *beam search*).

Prijelaz sa deterministčkog na determinističko parsanje unosi sasvim novi pristup ocjenjivanju stanja. Više se ne promatra kvaliteta pojedinog prijelaza, već je potrebno modelirati ocjenu kvalitete samog stanja i dotad izgrađenog ovisnosnog stabla. Bodovanje djelomično izgrađenog ovisnosnog stabla dobiva se na temelju kvalitete njegovih podstabla (kao i kod MST parsera), s tim da je moguće uvesti sintaktičke modele višeg reda - u slučaju parsera MATE to su modeli drugog i trećeg reda. Kao i kod graf

[7] S obzirom da je vektor težina mnogo manji od vektora značajki, više će značajki dijeliti istu težinu. Kolizije ne moraju nužno biti pogubne za točnost parsanja jer se može reći da značajke koje dijele težinu čine jednu kompleksnu značajku te tijekom rada parsera omogućuju rad sa značajkama koje se nisu pojavile tijekom treniranja.

varijante MATE parsera, i prijelaznička varijanta ovog parsera koristi hash funkciju te koristi manji vektor težina od vektora značajki.

Druga razlika u odnosu na sve dosad opisane parsere jest da prijelaznička varijanta MATE parsera modificira Nivreov algoritam za neprojektivno parsanje (Nivre, 2009)tako da istovremeno s ovisnosnim parsanjem vrši i POS označavanje riječ (Bohnet i Nivre, 2012)i. Konfiguraciju stanja čine:

1. σ stog s riječima;
2. β ulazna traka s koje automat čita;
3. A skup dotad dodanih veza;
4. π uređeni parovi riječi i pripadajućih POS oznaka; svaka riječ mmože imati samo jednu POS oznaku.

Mogući prijelazi tada su sljedeći:

1. $LEFT - ARC_r\ ([\sigma|i, j], \beta, A, \pi) \Rightarrow ([\sigma|j], \beta, A \cup \{(j, r, i)\}, \pi), i \neq 0$ dodaje novu vezu s oznakom r u graf, riječ i s vrha stoga ovisna je o riječi j s početka niza. Riječ i uklanja se iz daljnjeg razmatranja;
2. $RIGHT - ARC_r\ (\sigma|i, j|\beta, A, \pi) \Rightarrow ([\sigma|i], \beta, A \cup \{(i, r, j)\}, \pi)$ dodaje novu vezu s oznakom r u graf, riječ i s vrha stoga je glava riječi j s početka niza. Riječ j uklanja se iz daljnjeg razmatranja;
3. $SHIFT_p\ (\sigma, [i|\beta], A, \pi) \Rightarrow ([\sigma|i], \beta, A, \pi \cup (i, p))$ Postavlja riječ na stog i toj riječi pridaje POS oznaku;
4. $SWAP\ ([\sigma|i, j], \beta, A, \pi) \Rightarrow ([\sigma|j], [i|\beta], A\}, \pi), 0 < i < j$ zamjenjuje pozicije dvije riječi na vrhu stoga, te novi vrh stoga postavlja na ulaznu traku. Da bi ovaj prijelaz bio moguć, dvije riječi na vrhu stoga moraju biti u ispravnom redoslijedu (redoslijedu kojim su se pojavile u rečenici).

Dvije su promjene vidljive u odnosu na originalni algoritam: u definiciju stanja dodani su i parovi riječi i njihovih POS oznaka, a te se POS oznake dodaju prilikom postavljanja riječi na vrh stog. Ova promjena ujedno i povećava broj mogućih konfiguracija parsera. Da bi se ovaj prostor stanja smanjio, prije započinjanja samog postupka parsanja provodi se tagiranje koje daje listu mogućih POS oznaka za neku riječ te bodove za svaki mogući par riječi i POS oznake. Moguće je definirati maksimalan broj POS oznaka koje će se razmatrati u radu parsera, te dodatno definirati koliko manji

smiju biti bodovi razmatranih POS oznaka u odnosu na najbolju dobivenu POS oznaku za neku riječ.

Najbolja stanja koja će se razmatrati u sljedećoj iteraciji rada parsera biraju se u odnosu na dva parametra. Najprije se odabere određen broj najboljih hipoteza koje imaju različita ovisnosna stabla (u smislu topologije). Nakon toga se dodaju preostala najbolja stabla koja će biti varijante već dodanih stabala, koja će imati jednaku topologiju, ali različite labele u odnosu na već dodana stabla.

I prijelaznička i graf varijanta parsera primaju prilikom učenja ovisnosna stabla u CoNLL formatu. S obzirom da CoNLL format sadrži stupce sa zlatnim oznakama i oznakama dobivenim kroz korake analize rečenice, parser za učenje i rad značajke koje su mu potrebne za rad uzima P-stupaca (stupaca koji ne sadržavaju nužno zlatne oznake; PLEMMA, PPOS, PFEATS), dok se ono što se želi istrenirati uzima iz stupca za zlatnim oznakama (HEAD, DEPREL, POS kod prijelazničke varijante prasera). Parser rezultate svog rada zapisuje u P- stupce, dok ostale stupce prepisuje.

4.2. Banka stabala

Za optimizaciju skupa morfoloških atributa za hrvatski jezik korišten je podskup SETimes banke ovisnosnih stabala, dok je za testiranje modela na češkom i slovenskom jeziku korišteni podskupovi PDT, odnosno jos100k banke stabala. U nastavku su ukratko opisane statističke značajke korištenih skupova podataka.

4.2.1. Hrvatski jezik

SETimes banka ovisnosnih stabala (Agić i Merkler, 2013)[8] izgrađena je na temelju rečenica iz SETimes korpusa novinskih članaka koje su ručno lematizirane i morfološki označene. Morfološko označavanje napravljeno je prema Multex East standardu, verziji 4 (Agić et al., 2013).[9] Sama banka stabala dolazi u CoNLLx formatu, dok se MSD oznake (engl. *morphosyntactic description*) nalaze u POS stupcu zapisa. Da bi se mogla koristiti u radu optimizatora, banka je prethodno transformirana tako da su MSD oznake prebačene iz POS stupca u FEATS stupac uz imenovanje pojedinih atributa. U tablicama 4.1, 4.2, 4.3 prikazane su osnovne statističke značajke korištenog skupa ovisnosnih stabala.

[8]Dostupna na adresi `http://nlp.ffzg.hr/resources/corpora/setimes-hr/`

[9]Specifikacije standarda dostupne na `http://nlp.ffzg.hr/data/tagging/msd-hr.html`

Tablica 4.1: Osnovne statističke značajke SETimes banke stabala

značajka	broj
rečenica	2490
riječi	56424
različitih pojavnih oblika riječi	13497
različitih lema	7030
različitih MSD oznaka	628

Tablica 4.2: Raspodjela vrste riječi unutar SETimes banke stabala

vrsta riječi	posotak
imenica	33.62
glagol	14.72
pridjev	11.49
zamjenica	5.10
broj	2.38
prilog	3.34
prijedlog	9.72
veznik	5.29
usklik	0
čestica	0.40
kratica	0
interpunkcija	13.63
ostale	0.31

4.2.2. Slovenski jezik

Banka slovenskih ovisnosnih stabala jos100k (Erjavec et al., 2010) predstavlja bazu ručno morfosintaktičkih označenih rečenica slovenskog jezika.[10] Korištene morfološke značajke odgovaraju hrvatskima, dok je skup korištenih sintaktičkih funkcija reduciran na samo deset. Sama baza sadrži preko šest tisuća rečenica i dolazi u CoNLLx formatu, no za potrebe ovog rada korišten je samo jedan njihov podskup. Dodatno, zanemaren je sadržaj POS stupca. U tablicama 4.4, 4.5, 4.6 prikazane su osnovne statističke značajke korištenog skupa slovenskih ovisnosnih stabala.

[10]Dostupno na adresi `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0022-F59C-8?show=full`

Tablica 4.3: Raspodjela sintaktičkih oznaka unutar SETimes banke stabala

sintaktička funkcija	oznaka	posotak
objekt	Obj	7.42
prijedlog	Prep	9.63
predikat	Pred	9.31
priložna oznaka	Adv	4.97
imenski predikat	Pnom	1.65
atribut	Atr	26.39
ostalo	Oth	1.79
vanjska ovisnost	Elp	0.57
interpunkcija	Punc	13.24
nezavisna reč.	Co	3.37
pom. glagol	Aux	6.49
apozicija	Ap	2.90
zavisna reč.	Sub	3.49
subjekt	Sb	7.09
komplement	Atv	1.69

Tablica 4.4: Osnovne statističke značajke reduciranog skupa slovenskih ovisnosnih stabala

značajka	broj
rečenica	2674
riječi	50822
različitih pojavnih oblika riječi	16100
različitih lema	9374
različitih MSD oznaka	594

Tablica 4.5: Raspodjela vrste riječi unutar reduciranog skupa slovenskih ovisnosnih stabala

vrsta riječi	posotak
imenica	24.39
glagol	15.93
pridjev	9.83
zamjenica	6.36
broj	2.46
prilog	5.14
prijedlog	8.84
veznik	7.80
usklik	0.02
čestica	2.83
kratica	0.36
interpunkcija	15.43
ostale	0.58

Tablica 4.6: Raspodjela sintaktičkih oznaka unutar reduciranog skupa slovenskih ovisnosnih stabala

sintaktička funkcija	oznaka	posotak
ostale priložne oznake	štiri	6.03
konjunkcija	vez	8.11
atribut	dol	32.67
objekt	dve	6.73
veza s korijenom	modra	29.18
priložna oznaka načina	tri	2.42
dio predikata	del	6.79
subjekt	ena	5.05
skup od više riječi	skup	0.31
koordinarane veze	prir	2.71

4.2.3. Češki jezik

Za češki je jezik korištena Praška banka ovisnosnih stabala (Böhmová et al., 2003), i to zapis te banke stabala u CoNLL formatu korišten za potrebe CoNLL natjecanja iz 2009. godine.[11] Iako dostupni zapis sadrži preko trideset tisuća rečenica koje imaju i zlatne i sistemske POS i morfološke oznake, za potrebe ovog rada korištene su samo zlatne oznake te jedan manji podskup rečenica. I ova baza dijeli postojeće morfološke značajke sa slovenskom i hrvatskom bankom stabala. U tablicama 4.7, 4.8, 4.9 prikazane su osnovne statističke značajke korištenog skupa čeških ovisnosnih stabala.

Tablica 4.7: Osnovne statističke značajke reduciranog skupa čeških ovisnosnih stabala

značajka	broj
rečenica	2719
riječi	45865
različitih pojavnih oblika riječi	14703
različitih lema	8320
različitih MSD oznaka	975

Tablica 4.8: Raspodjela vrste riječi unutar reduciranog skupa čeških ovisnosnih stabala

vrsta riječi	posotak
imenica	30.61
glagol	12.38
pridjev	12.17
zamjenica	6.66
broj	3.06
prilog	5.08
prijedlog	9.91
veznik	5.59
usklik	0.01
čestica	0.41
interpunkcija	14.13
ostale	0

[11] Dostupan na adresi `https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0001-C6D1-9`

Tablica 4.9: Raspodjela sintaktičkih oznaka unutar reduciranog skupa čeških ovisnosnih stabal

sintaktička funkcija	posotak	sintaktička funkcija	posotak
Apos	0.39	Adv	9.46
Coord	3.52	AtrAdv_M	0.03
ObjAtr	0	Sb_M	0.89
Coord_M	0.43	AtvV	0.07
AtrObj_M	0.01	ExD_M	1.02
Pnom	1.37	Apos_M	0.05
Atv	0.17	AuxR	0.34
AuxP	10.19	AuxO	0.03
AtrObj	0.01	Pred	3.91
AuxK	5.24	Adv_M	0.89
Atr	26.54	AuxG	2.09
Atv_M	0	AuxZ	1.51
Atr_M	2.46	AuxY	0.74
AuxX	4.88	AtrAtr	0.04
Sb	6.51	AuxV	1.32
AdvAtr	0.01	AuxT	1.22
Obj	7.24	AuxZ_M	0
Pnom_M	0.14	AtrAdv	0.14
Obj_M	1.07	AuxC	1.76
Pred_M	2.26	ExD	2.01

4.3. Provedeni eksperiment

Implementirani parser iskorišten je da bi se pronašao onaj skup atributa koji daje najbolje rezultate za hrvatski jezik.

Samo ispitivanje odvijalo se u nekoliko koraka. Najprije je za MST, Malt i MATE parser (graf varijantu) napravljeno pohlepno pretraživanje unatrag i unaprijed uz skupno uklanjane atributa pri čemu se koristilo pohlepno pretraživanje po slojeva – dakle u sljedeću se u sljedeću iteraciju prenosi samo jedno, najbolje dijete. Unaprijedno pretraživanje znači da se kreće iz praznog skupa atributa te se postepeno dodaje jedan po jedan atribut, dok pretraživanje unatrag kreće od svih mogućih atributa uklanja ih. Skup za testiranje sadržavao je 301 rečenicu, dok su sve ostale rečenice iz banke stabala poslužile za treniranje.

Ova su dva pretraživanja napravljena u dvije varijante – klasičnoj i delematiziranoj. Nakon toga je su za najbolji parser odabrani najveći i najmanji set koji su poslužili kao ishodišne točke za pretraživanje skupa atributa na "finijoj" razini (uz uklanjanje atributa iz jedne po jedne vrste riječi), opet po slojevima, ali uz veći broj djece koja se prenose su sljedeću iteraciju (5). Iz ukupnog je seta na kraju ručno izabrano nekoliko potencijalno zanimljivih setova atributa (različitih veličina, ali sa vrlo dobrim rezultatima rada parsera) te je za njih provedena obična unakrsna provjera, kao i unakrsna provjera uz ulančavanje morfološkog označavanja i parsanja. Pritom su obje provjere izvedene u klasičnoj i delematiziranoj varijanti. Unakrsna se validacija radila uz 10 preklopa. Morfološko označavanje napravljeno je korištenjem morfološkog označivača koji dolazi uz parser MATE uz korištenje automatskih postavki.

MST parser je korišten u varijanti parsanja Chu-Liu-Edmondsovim algoritmom, uz korištenje sintaktičkog modela drugog reda pri čemu je za broj iteracija treniranja iznosio 10, a broj uvjeta minimizacije iznosio je 5. Što se tiče Malt parsera, najprije je iskorišten MaltOptimizer koji je utvrdio sa je najbolje koristiti pseudo-projektivno parsanje (uz $baseline$ projektivizaciju) korištenje algoritma $stack-projective$. MaltOptimizer je također kreirao i datoteku s optialnim skupom značajki za opis konfiguracije (primjer 4.1) prijelazničkog automata koji je korišten prilikom treniranja i testiranja. Kao klasifikator korišten je linearni klasifikator LIBLINEAR uz opciju $s\_4\_c\_0.1$. MATE parser je u graf varijanti radio s automatskim postavkama. Prijelaznički MATE parser nije detaljno ispitan jer učenje parsera traje predugo, nego je samo ispitan rad za najzanimljivije skupove atributa i to samo jednom (postojeća je baza podijeljena na skup za treniranje i testiranje analogno podjeli koja se koristila prilikom treniranja i testiranja graf varijante MATE parsera). Korištena je veličina prozora od 60 konfiguracija, 25 iteracija učenja, korištenje svih mogućih značajki, 10 POS označivača koji su trenirani kroz 15 iteracija te su korištene najbolje 3 POS oznake u radu samog parsera uz dopušten prag korištenih POS oznaka 0.2 uz automatsku veličinu vektora značajki.

Na kraju je najbolji model ispitan na bazi slovenskih stabala jos1000k i bazi praških stabala PDT.

5. Rezultati

U nastavku se navode rezultati za sve provedene testove opisane u prethodnom poglavlju uz kratki komentar i usporedbu rada korištenih parsera.

5.1. Rezultati pohlepnog pretraživanja

Parser MATE

Tablica 5.1 prikazuje 10 najboljih setova atributa za klasičnu i delematiziranu varijantu pohlepnog pretraživanja za parser MATE te dobivene rezultate za svaki od njih.

Zanimljivo je da najbolji rezultati u oba slučaja nisu sačinjeni od potpunog seta atributa, kao i da je za delematizirano parsanje potrebno imati što više atributa. U slučaju klasičnog parsanja rezultati mogu biti vrlo dobri već za male setove atributa. Važno je napomenuti da se kod klasičnog parsanja već sa samo dva atributa može postići uspješnost parsanja koja ima LAS samo 1% lošiji od najbolje varijante te da se većina rezultata nalazi između 76 i 79% uspješnosti LAS-a. Delematizirano parsanje radi vrlo loše za setove atribut male veličine, no prihvatljivo za setove srednje veličine.

Tablica 5.2 prikazuje rezultate ispitivanja rada parsera za odabrane skupove atributa različite veličine. Vidljivo je da smanjenjem korištenog skupa atributa pada i točnost rada parsera. Postoji jedna devijacija i to za skup atributa „S::cs, A::adjT, N::cs, N::nounT“, koja je po uspješnosti usporediva s korištenjem potpunog skupa atributa. Rezultati su lošiji u usporedbi s varijantom parsera MATE koji se temelji na grafovima, no treba uzeti u obzir da je ovaj parser istovremeno radio i POS označavanje. Deleksikalizirana varijanta nije napravljena s obzirom da nema pretjeranog smisla uzme li se u obzir parser radi i POS označavanje za što mu trebaju značajke riječi koja se pojavljuje u rečenici i značajke njezine leme.

Tablica 5.1: Rezultati pohlepnog pretraživanja za varijantu parsera MATE temeljenu na grafovima

Klasična varijanta			
Atributi	LAS	UAS	LA
nounT, cs, anim, verbT, verbF, per, adjT, deg, pronT, refT, advT, numF, numT	0.7883	0.8477	0.8941
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, refT, frmtn, numF, numT, resT	0.7872	0.8498	0.8890
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, refT, synT, frmtn, numF, numT, parT, resT	0.7870	0.8470	0.8887
nounT, cs, anim, verbF, per, adjT, numF	0.7870	0.8472	0.8914
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, refT, conT, frmtn, numF, numT	0.7868	0.8472	0.8919
nounT, cs, anim, verbT, verbF, per, adjT, deg, refT, advT, numF	0.7867	0.8480	0.8901
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, refT, frmtn	0.7864	0.8474	0.8881
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, ownN, ownG, cut, refT, synT, advT, frmtn, numF, numT, parT, resT	0.7859	0.8464	0.8908
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, ownN, ownG, cut, refT, synT, advT, numF, numT, parT, resT	0.7859	0.8464	0.8908
nounT, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, ownN, ownG, refT, synT, advT, numF, numT, parT, resT	0.7859	0.8461	0.8905
Delematizirana varijanta			
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, def, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7555	0.8181	0.8723
nounT, gen, cs, anim, verbT, verbF, per, adjT, deg, def, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7532	0.8202	0.8686
nounT, gen, num, cs, anim, verbT, per, adjT, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7528	0.8170	0.8711
nounT, gen, num, cs, verbT, per, adjT, pronT, refT, synT, advT, conT, numT, parT	0.7528	0.8183	0.8672
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, deg, def, pronT, ownN, ownG, cut, refT, synT, advT, conT, frmtn, numF, numT, parT, resT	0.7527	0.8172	0.8699
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, deg, def, pronT, ownN, ownG, cut, refT, synT, advT, conT, numF, numT, parT, resT	0.7527	0.8172	0.8699
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7520	0.8170	0.8689
nounT, gen, num, cs, anim, verbT, verbF, neg, adjT, deg, def, pronT, ownN, ownG, cut, refT, synT, advT, conT, frmtn, numF, numT, parT, resT	0.7517	0.8170	0.8694
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, def, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT	0.7509	0.8151	0.8686
nounT, gen, num, cs, anim, verbT, verbF, per, neg, adjT, deg, def, pronT, ownG, cut, advT, conT, frmtn, numF, numT, parT, resT	0.7509	0.8159	0.8688

Tablica 5.2: Rezultati ispitivanja za prijelazničku varijantu parsera MATE

atributi	LAS	UAS
all	**0.7752**	**0.8378**
anim, adjT, refT, deg, numT, numF, pronT, nounT, verbF, advT, cs, per, verbT	0.7682	0.8314
anim, adjT, numF, nounT, verbF, cs, per	0.7695	0.833
adjT, nounT, cs, per	0.7680	0.8325
adjT, pronT, cs	0.7686	0.8341
pronT, cs	0.7541	0.8173
nounT, gen, num, cs, verbT, verbF, adjT, pronT, advT, conT	0.7672	0.8290
nounT, gen, num, cs, anim, verbT, verbF, per, adjT, def, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7672	0.8290
P::pronT, V::verbF, M::numF, P::refT, P::cs, P::per, S::cs, M::cs, M::numT, P::anim, A::cs, A::adjT, R::advT, N::cs, N::nounT, M::anim, A::anim, V::per, R::deg, A::deg	0.7631	0.252
P::cs, P::per, A::adjT, N::cs, N::nounT, V::per	0.7631	0.8252
V::verbF, M::numF, V::verbT, P::cs, S::cs, M::numT, A::cs, A::adjT, R::advT, N::cs, N::nounT, V::per, R::deg	0.7619	0.8258
S::cs, A::adjT, N::cs, N::nounT	0.7714	0.8351
none	0.7260	0.8022

5.1.1. Parser Malt

Tablica 5.3 prikazuje 10 najboljih setova atributa za klasičnu i delematiziranu varijantu pohlepnog pretraživanja korištenjem parsera Malt te dobivene rezultate za svaki od njih. I kod Malt parsera može se primijetiti da se idealni setovi ponašaju na sličan način kao i kod parsera MATE, ali ipak sam parser radi značajno lošije. I ovdje parser radi vrlo dobro za skupove svih veličina koristi li se leksikalizirana varijanta parsanja. Također se može primijetiti da je pad u kvaliteti rada parsera kod delematiziranog parsanja mnogo veći u odnosu na klasično parsanje kada se usporede MATE i Malt parser.

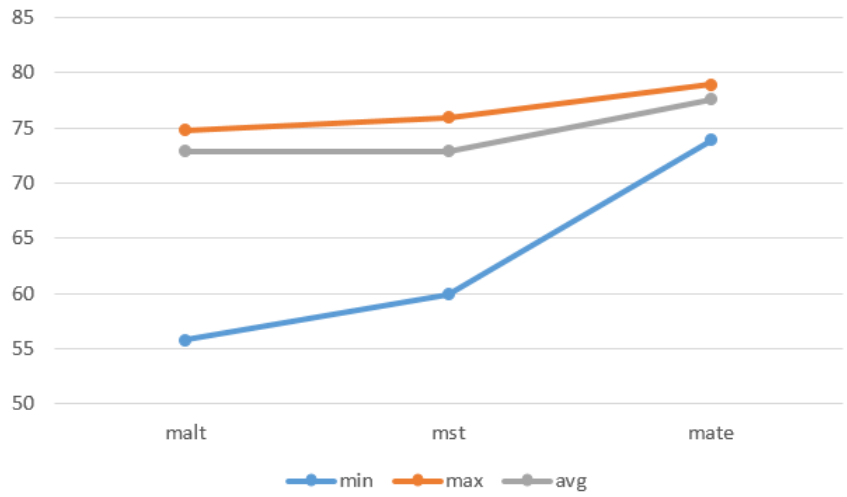
5.1.2. MST parser

Tablica 5.4 prikazuje rezultate rada optimizatora za MST parser. Parser MST pokazuje bolje rezultate parsanja od parsera Malt, što je u skladu s dosadašnjim ispitivanjima na morfološki složenim jezicima poput hrvatskog (Željko Agić, 2012). Također je primjetno da se parseri temeljeni na grafovima mnogo bolje nose s delematiziranim parsanjem (pad je u odnosu na klasično parsanje manji nego kod Malt parsera). Kod svih se parsera vidi da su najbolji rezultati dobiveni za skupove atributa srednje veličine (otprilike upola manji od veličine izvornog seta atributa), te da je redukcija u skupu značajki značajna. Parser MST možda ima i najviše koristi od smanjenja skupa značajki. Naime, ovaj parser za razliku od drugih ne promatra morfološke značajke neke riječi zasebno, već ih uzima kao cjelinu. Ovo znači da teže razaznaje kada riječi različite vrste dijele neku značajku i ima veći prostor mogućih značajki koje koristi. Ipak, baš kao i parser MATE i kod ovog je parsera za dobre rezultate u deleksikaliziranom parsanju potrebno što više morfoloških značajki.

Slika 5.1 prikazuje usporedbu rada parsera Malt, MATE (varijante temeljene na grafovima) i MST u ovisnosti o različitim odabranim skupovima morfoloških značajki. Narančastom je linijom označena maksimalna vrijednost koju mogu dostići parseri, sivom prosječna vrijednost rezultata parsanja nad svim ispitanim skupovima, dok je plavom bojom označena najmanja postignuta vrijednost parsanja. Vidljivo je da je parser MATE ima najbolje rezultate, te da se njegov najslabiji rezultat može usporediti s prosječnom performansom preostala dva parsera. Iz ovog su razloga daljnja ispitivanja rađena samo korištenjem graf varijante parsera MATE.

Tablica 5.3: Rezultati pohlepnog pretraživanja za Malt parser

Klasična varijanta			
Atributi	LAS	UAS	LA
synT, adjT, numT, numF, parT, nounT, advT, cs, num, verbT, frmnt	0.7477	0.8084	0.8694
synT, adjT, numT, numF, parT, nounT, advT, cs, num, verbT	0.7477	0.8084	0.8694
synT, adjT, parT, nounT, advT, cs, verbT	0.7472	0.8070	0.8683
synT, adjT, numF, parT, nounT, advT, cs, num, verbT, frmnt	0.7471	0.8081	0.8689
synT, adjT, numT, numF, resT, parT, nounT, advT, cs, num, verbT, frmnt	0.7471	0.8078	0.8692
synT, adjT, numF, parT, nounT, advT, cs, num, verbT	0.7471	0.8081	0.8689
synT, adjT, numT, numF, resT, parT, nounT, advT, cs, num, verbT	0.7471	0.8078	0.8692
synT, adjT, numF, resT, parT, nounT, advT, cs, num, verbT	0.7466	0.8078	0.8691
synT, adjT, refT, numF, nounT, advT, cs, num, verbT	0.7463	0.8073	0.8675
synT, adjT, numT, numF, resT, parT, nounT, advT, cs, num, verbT, cut, frmnt	0.7460	0.8078	0.8678
Delematizirana varijanta			
synT, adjT, refT, def, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, frmnt, conT	0.7024	0.7654	0.8392
synT, adjT, refT, def, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, conT	0.7024	0.7654	0.8392
synT, adjT, refT, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, conT	0.7016	0.7648	0.8379
synT, adjT, refT, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, frmnt, conT	0.7016	0.7648	0.8379
synT, adjT, refT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, per, verbT, frmnt, conT	0.7014	0.7645	0.8381
synT, adjT, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, conT	0.7013	0.7643	0.8386
adjT, refT, numF, pronT, resT, parT, nounT, verbF, advT, cs, per, verbT, conT	0.7009	0.7642	0.8371
synT, adjT, refT, def, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, per, verbT, frmnt, conT	0.7009	0.7631	0.8400
synT, adjT, refT, def, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, per, verbT, conT	0.7009	0.7631	0.8400
adjT, refT, def, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, per, verbT, frmnt, conT	0.7067	0.7632	0.8395



Slika 5.1: Usporedba rezultata rada parsera

5.2. Rezultati analize zanimljivih skupova atributa

Tablica 5.5 navodi odabrane skupove atributa te rezultate dobivene tijekom prve faze rada optimizatora. Svakom je skupu dodijeljena oznaka koja će se koristiti u nastavku rada iz koje se može pročitati kojom je pretragom dobiven: zajedničkim (C) ili individualnom (I) izbacivanjem atributa, te na klasičan (K) ili deleksikaliziran (D) način. Ukoliko je skup atributa dobiven individualnom manipulacijom skupa atributa, tada se njegova oznaka prikazuje na sljedeći način: „vrsta riječi::atribut“.

Tablica 5.6 prikazuje rezultate unakrsne provjere za svaki od odabranih skupova atributa. Napravljena su 4 ispitivanja: na skupu rečenica sa zlatnim morfološkim oznakama, zatim na deleksikaliziranom skupu sa zlatnim morfološkim oznakama te konačno na leksikaliziranom i deleksikaliziranom skupu rečenica, ali uz sistemske morfološke oznake. Vidljivo je da skupovi srednje i velike veličine daju vrlo slične rezultate na leksikaliziranom skupu rečenica, no da se na deleksikaliziranom skupu rečenica ipak bolje ponašaju oni skupovi morfoloških atributa koji su dobiveni deleksikaliziranim pretraživanjem, što donekle opravdava intuiciju da taj način pretraživanja daje bolju sliku o kvaliteti odabranih morfoloških oznaka. Naime tako se dobiveni skupovi ponašaju dobro u obje situacije. Ono što pomalo razočarava jest da se smanjenjem seta morfoloških oznaka nije značajno doprinijelo poboljšanju parsanja sa sistemskim značajkama, odnosno nije se smanjio utjecaj greške na ulazu u parser koji je posljedica automatizirane morfološke analize.

Tablica 5.7 prikazuje preciznost, odziv i F_1 mjeru za postojeće sintaktičke funkcije ovisno o odabranom skupu atributa. Mjerenje je dobiveno iz unakrsne provjere kori-

štenjem sistemskih MSD oznaka uz leksikaliziranu varijantu parsanja. Iz ove je tablice vidljivo da postoje neke sintaktičke funkcije koje uopoće ne ovise o odabranom skupu atributa i vjerojatno se mogu odrediti na temelju same pojavnice i vrste riječi (primjerice pomoćni glagol i interpunkcijski znakovi). Oscilacije se događaju kod složenijih sintaktičkih funkcija kao što su objekt, subjekt i priložna oznaka. Ipak ono što se može uočiti jest da iako ne postoji jedan skup atributa koji bi davao najbolje rezultate za sve sintaktičke funkcije, ipak najbolje ponašanje pokazuju skupovi atributa srednje i velike veličine. Dodatno se može primijetiti da bi se parsanje moglo poboljšati ukoliko bi se istovremeno koristilo više modela učenih na različitim kombinacijama skupova morfoloških atributa.

Konačna analiza napravljena je tako da se analizira ponašanje modela ovisno o preklopu. Tablica 5.8 prikazuje statističke značajke pojedinih preklopa dok tablica 5.9 prikazuje najbolje dobivene rezultate po preklopima za svaku od testiranih varijanti unakrsne provjere ovisno o tome jesu li korištene zlatne MSD oznake ili ne, te je li korištena leksikalizirana ili deleksikalizirana varijanta parsanja. Uz svaki je rezultat prikazan i skup atributa za koje su ti rezultati postignuti. Ono što je zanimljivo jest da rezultati parsanja ovisno o preklopu dosta variraju unutar jednog skupa atributa, no da bi se kombiniranjem tih skupova mogli postići još bolji rezultati parsanja. Ono što je zanimljivo jest da veličine skupova atributa za koje su postignuti maksimumi u kvaliteti parsanja variraju u veličini od najmanjih do najvećih kod leksikaliziranog parsanja, međutim kod deleksikaliziranog parsanja ipak se maksimumi postižu za skupove srednje i velike veličine. Mali skupovi atributa dobri su u situacijama u kojima ima veći broj pridjeva, manji broj glagola i imenica.

5.3. Primjena modela na druge jezike

Tablica 5.10 prikazuje rezultate ispitivanja najboljeg modela na češkom i slovenskom jeziku. S obzirom da je ispitivanje rađeno na relativno malom setu rečenica, češki jezik daje vrlo slabe rezultate parsanja što je posljedica činjenice da ima definirano čak 40 različitih mogućih sintaktičkih funkcija. Slovenski pak daje bolje rezultate parsanja, opet zbog činjenice da ima manje sintaktičkih funkcija nego hrvatski jezik. I dok kod češko postoji pad u kvaliteti parsanja kada se koristi reducirani skup podataka u svim slučajevima, kod slovenskog je za dobiveni model parsanje čak i kvalitetnije u leksikaliziranoj varijanti, iako se to poboljšanje ne bi moglo nazvati statistički značajnim. Primjetno je da je kod slovenskog varijanca u kvaliteti parsanja po preklopima dosta velika iz čega se može zaključiti se optimalni parametri parsanja ne mijenjaju samo od

jezika do jezika već ovise i o svojstvima danog skupa rečenica koje se parsiraju. Dodatno, očito je da reduciranje skupa atribute ne donosi pretjerano velik pad u kvaliteti parsanja te i kod ovih jezika postoji mogućnost da sa smanji skup atributa, a da se ne smanji značajno i kvaliteta parsanja.

Tablica 5.4: Rezultati pohlepnog pretraživanja za MST parser

Klasična varijanta			
Atributi	LAS	UAS	LA
adjT, numF, pronT, nounT, ownN, advT, cs, ownG, verbT, frmnt, conT	0.7587	0.8223	0.8753
adjT, refT, numF, pronT, nounT, ownN, advT, cs, ownG, verbT, frmnt, conT	0.7581	0.8199	0.8764
neg, adjT, refT, def, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, conT	0.7583	0.8213	0.8771
neg, adjT, refT, def, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, frmnt, conT	0.7583	0.8213	0.8770
neg, adjT, numF, pronT, nounT, ownN, advT, cs, ownG, verbT, frmnt, conT	0.7579	0.8181	0.8771
anim, adjT, refT, def, numF, pronT, nounT, ownN, verbF, advT, cs, ownG, verbT, frmnt, conT	0.7575	0.8196	0.8808
anim, adjT, refT, numF, pronT, nounT, ownN, advT, cs, ownG, verbT, frmnt, conT	0.7570	0.8178	0.8758
anim, adjT, numF, pronT, nounT, ownN, advT, cs, verbT, frmnt, conT	0.7570	0.8194	0.8761
neg, adjT, refT, def, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, conT	0.7570	0.8194	0.8777
neg, adjT, refT, def, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, frmnt, conT	0.7570	0.8194	0.8777
Delematizirana varijanta			
synT, adjT, numF, pronT, resT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7314	0.7971	0.8606
anim, synT, adjT, numF, pronT, resT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7314	0.7971	0.8632
neg, anim, synT, adjT, def, deg, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, conT	0.7308	0.7966	0.8628
neg, anim, synT, adjT, def, deg, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, frmnt, conT	0.7308	0.7966	0.8628
neg, anim, synT, adjT, refT, def, deg, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, frmnt, conT	0.7308	0.7966	0.8628
neg, anim, synT, adjT, refT, def, deg, numT, numF, pronT, resT, parT, nounT, ownN, verbF, advT, cs, ownG, verbT, cut, conT	0.7308	0.7966	0.8628
neg, anim, synT, adjT, def, deg, numT, numF, pronT, resT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7302	0.7947	0.8617
neg, anim, synT, adjT, def, numT, numF, pronT, resT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7300	0.7942	0.8619
neg, anim, synT, adjT, numT, numF, pronT, resT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7298	0.7950	0.8608
neg, anim, synT, adjT, def, deg, numT, numF, pronT, parT, nounT, verbF, advT, cs, verbT, cut, conT	0.7297	0.7947	0.8616

Tablica 5.5: Zanimljivi skupovi atributa

Oznaka	atributi	LAS	UAS	LA
All	potun skup atributa	0.7808	0.8407	0.8897
CD10	nounT, gen, num, cs, verbT, verbF, adjT, pronT, advT, conT	0.7508	0.8156	0.8691
CK7	anim, adjT, numF, nounT, verbF, cs, per	0.7870	0.8472	0.8914
CK2	pronT, cs	0.7738	0.8346	0.8807
CK4	adjT, nounT, cs, per	0.7837	0.8442	0.8906
CK3	adjT, pronT, cs	0.7731	0.8344	0.8836
CD14	nounT, gen, num, cs, verbT, per, adjT, pronT, refT, synT, advT, conT, numT, parT	0.7528	0.8183	0.8671
CK13	anim, adjT, refT, deg, numT, numF, pronT, nounT, verbF, advT, cs, per, verbT	0.7883	0.8477	0.8941
CD21	nounT, gen, num, cs, anim, verbT, verbF, per, adjT, def, pronT, ownN, ownG, refT, synT, advT, conT, numF, numT, parT, resT	0.7555	0.8181	0.8723
IK20	P::pronT, V::verbF, M::numF, P::refT, P::cs, P::per, S::cs, M::cs, M::numT, P::anim, A::cs, A::adjT, R::advT, N::cs, N::nounT, M::anim, A::anim, V::per, R::deg, A::deg	0.7893	0.8491	0.8887
IK6	P::cs, P::per, A::adjT, N::cs, N::nounT, V::per	0.7817	0.8418	0.8879
IK13	V::verbF, M::numF, V::verbT, P::cs, S::cs, M::numT, A::cs, A::adjT, R::advT, N::cs, N::nounT, V::per, R::deg	0.7884	0.8491	0.8924
IK4	S::cs, A::adjT, N::cs, N::nounT	0.7766	0.8414	0.8830

Tablica 5.6: Unakrsna provjera odabranih skupova

	Gold MSD + lex			Gold MSD + delex		
	LAS %	UAS %	LA %	LAS %	UAS %	LA%
All	81.17 ± 1.92	86.55 ± 1.06	90.40 ± 1.6	**78.81 ± 1.65**	**84.69 ± 1.33**	**89.16 ± 1.07**
CD10	**81.48 ± 1.71**	**86.85 ± 1.36**	90.51 ± 0.96	78.78 ± 2.05	84.68 ± 1.67	89.03 ± 1.24
CK7	81.13 ± 1.54	86.51 ± 1.25	90.35 ± 0.94	76.6 ± 2.03	82.78 ± 1.7	87.48 ± 1.27
CK2	79.77 ± 1.56	85.11 ± 1.30	89.57 ± 0.86	72.99 ± 1.74	79.16 ± 1.41	85.60 ± 1.23
CK4	81.14 ± 1.86	86.5 ± 1.48	90.31 ± 1.03	75.73 ± 2.03	81.9 ± 1.72	86.98 ± 1.34
CK3	80.94 ± 1.55	86.25 ± 1.24	90.30 ± 0.94	74.37 ± 2.17	80.85 ± 1.79	86.08 ± 1.43
CD14	81.43 ± 1.71	86.80 ± 1.42	90.49 ± 0.93	78.75 ± 1.95	84.69 ± 1.59	89.03 ± 1.21
CK13	81.45 ± 1.61	86.75 ± 1.24	90.62 ± 1.00	77.98 ± 1.79	83.97 ± 1.48	88.46 ± 1.27
CD21	81.37 ± 1.82	86.70 ± 1.51	**90.53 ± 1.01**	78.80 ± 1.83	84.73 ± 1.45	89.09 ± 1.17
IK20	81.14 ± 1.74	86.63 ± 1.33	90.34 ± 1.14	76.98 ± 1.84	83.14 ± 1.41	87.69 ± 1.3
IK6	80.66 ± 1.75	86.1 ± 1.37	90.11 ± 1.13	75.44 ± 2.16	81.62 ± 1.84	86.72 ± 1.43
IK13	81.03 ± 1.65	86.39 ± 1.33	90.31 ± 1.05	77.19 ± 2.2	83.43 ± 1.76	87.92 ± 1.44
IK4	80.71 ± 1.82	86.18 ± 1.46	90.01 ± 1.17	73.74 ± 2.15	80.3 ± 1.74	85.47 ± 1.58
	System MSD + lex			System MSD + delex		
All	79.08 ± 2.09	85.34 ± 1.76	88.58 ± 1.33	76.51 ± 2.18	83.31 ± 1.88	87.07 ± 1.59
CD10	79.34 ± 1.84	85.62 ± 1.43	88.58 ± 1.24	76.75 ± 2.05	83.54 ± 1.67	87.24 ± 1.37
CK7	79.18 ± 1.81	85.35 ± 1.47	88.65 ± 1.14	74.72 ± 2.21	81.59 ± 1.79	85.74 ± 1.47
CK2	77.56 ± 1.91	83.81 ± 1.51	87.57 ± 1.41	72.13 ± 3.31	78.96 ± 3.13	84.12 ± 1.85
CK4	79.11 ± 2.08	85.36 ± 1.6	88.61 ± 1.38	73.99 ± 2.25	80.92 ± 1.85	85.3 ± 1.49
CK3	78.97 ± 2.11	85.21 ± 1.65	88.48 ± 1.39	73.8 ± 3.29	80.91 ± 3.12	84.93 ± 1.81
CD14	79.27 ± 1.87	85.52 ± 1.56	88.69 ± 1.19	**76.76 ± 2**	**83.58 ± 1.69**	**87.26 ± 1.36**
CK13	**79.64 ± 2.14**	**85.77 ± 1.71**	**88.88 ± 1.46**	76.21 ± 1.93	82.88 ± 1.53	86.96 ± 1.41
CD21	79.21 ± 2.09	85.48 ± 1.72	88.63 ± 1.27	76.30 ± 2.46	83.13 ± 2.06	86.96 ± 1.75
IK20	79.46 ± 1.89	85.67 ± 1.45	88.84 ± 1.26	75.10 ± 2.13	81.91 ± 1.6	86.1 ± 1.51
IK6	78.95 ± 1.90	85.23 ± 1.51	88.44 ± 1.32	73.67 ± 2.3	80.58 ± 1.92	85 ± 1.53
IK13	79.4 ± 1.95	85.52 ± 1.56	88.8 ± 1.27	75.17 ± 2.75	82.06 ± 2.27	86.13 ± 1.91
IK4	78.89 ± 2.03	85.24 ± 1.63	88.36 ± 1.37	71.85 ± 2.28	79.17 ± 1.8	83.77 ± 1.77

Tablica 5.7: Preciznost, odziv i F1-mjera za postojeće sintaktičke funkcije ovisno o odabranom skupu atrubuta

	obj			prep			pred			adv		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All	73.23	69.77	71.46	97.27	97.86	97.57	93.93	94.19	94.06	68.62	**63.92**	66.19
CD10	71.88	69.48	70.66	97.42	97.86	97.64	94.55	94.73	94.64	69.26	63.39	**66.20**
CK7	73.44	68.93	71.11	97.17	97.96	97.56	94.83	94.88	94.85	68.72	63.21	65.85
CK2	73.13	66.92	69.89	97.40	97.85	97.62	92.28	92.88	92.58	68.43	62.67	65.43
CK4	73.47	68.38	70.83	97.33	98.07	97.70	94.07	94.88	94.48	68.90	62.64	65.62
CK3	72.70	68.09	70.32	97.26	97.97	97.62	93.75	94.56	94.15	69.57	62.42	65.80
CD14	72.64	**70.20**	71.40	97.36	97.94	97.65	94.42	94.69	94.55	69.30	63.10	66.06
CK13	73.76	69.67	**71.66**	**97.58**	97.88	97.73	94.94	95.07	95.01	69.45	63.14	66.14
CD21	72.35	69.67	70.99	97.44	97.97	97.71	94.3 5	94.73	94.54	68.45	63.28	65.77
IK20	**74.01**	69.69	71.78	97.49	97.88	97.69	94.47	94.98	94.82	68.95	62.92	65.80
IK6	72.24	69.62	71.01	97.40	97.85	97.62	94.01	94.69	94.35	69.26	62.64	65.78
IK13	73.83	69.67	71.69	97.48	**98.16**	**97.82**	**95.00**	**95.41**	**95.20**	69.36	63.10	66.08
IK4	71.96	68.71	70.30	97.29	97.83	97.56	93.84	94.46	94.15	**69.67**	62.57	65.93

	pnom			atr			oth			elp		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All	65.59	54.56	59.57	88.79	92.00	90.37	**73.54**	**65.08**	**69.05**	54.89	**40.49**	46.40
CD10	66.16	55.52	60.37	88.51	91.88	90.17	73.19	63.10	67.77	55.56	38.94	45.79
CK7	67.19	54.66	60.28	88.68	92.10	90.36	73.25	62.10	67.27	51.87	38.94	44.48
CK2	59.78	52.09	55.67	87.22	92.03	89.56	71.86	63.10	67.19	54.20	40.19	46.15
CK4	65.19	55.20	59.78	88.54	92.32	90.39	72.90	63.79	68.04	53.81	39.56	45.60
CK3	67.00	57.02	61.61	88.63	92.32	**90.44**	73.27	63.89	68.26	54.42	38.32	44.97
CD14	66.41	55.73	60.61	**88.69**	91.94	90.28	72.89	65.08	68.76	56.76	39.25	**46.41**
CK13	**67.38**	**57.13**	**61.83**	88.62	92.26	90.41	71.64	63.89	67.54	56.50	39.25	46.32
CD21	66,24	54.88	60.02	88.68	92.03	90.32	72.85	64.68	68.52	**57.08**	37.69	45.40
IK20	66.36	54.98	60.14	88.47	**92.43**	90.41	73.16	64.09	68.32	56.00	39.25	46.15
IK6	65.95	52.95	58.74	8831	91.93	90.08	72.81	63.49	67.83	51.95	37.38	43.48
IK13	66.97	55.63	60.77	88.43	63.59	67.72	72.43	63.59	67.72	54.02	37.69	44.40
IK4	66.53	53.70	59.43	88.37	92.09	90.19	72.98	64.58	68.53	52.23	40.19	45.42

	punc			co			aux			sup		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
All	98.34	99.34	98.84	93.70	91.63	92.65	94.30	96.72	95.50	95.05	95.35	95.22
CD10	98.38	**99.44**	98.91	94.24	92.26	93.24	94.92	96.53	95.72	95.29	96.35	**95.81**
CK7	98.36	99.38	98.87	93.49	91.47	92.47	94.62	**97.51**	96.05	94.75	95.23	94.99
CK2	98.38	99.42	98.90	92.29	91.52	92.40	93.69	95.30	94.49	95.52	95.23	95.38
CK4	98.41	99.32	98.86	93.68	92.10	92.88	94.38	97.30	95.82	**95.91**	95.08	95.49
CK3	98.34	99.32	98.83	93.70	91.57	92.62	94.41	96.80	95.59	95.06	95.54	95.30
CD14	98.36	99.29	98.82	93.53	91.31	92.41	94.77	96.94	95.84	95.51	**95.89**	95.70
CK13	98.46	99.28	98.87	93.27	91.89	92.57	94.95	97.13	96.01	95.27	94.93	95.10
CD21	98.34	99.36	98.85	93.99	91.42	92.69	94.75	96.67	95.70	95.69	95.69	95.69
IK20	**98.53**	99.34	**98.93**	93.66	**92.63**	93.14	94.75	97.02	95.87	95.59	95.54	95.56
IK6	98.32	99.41	98.86	94.04	91.36	92.68	94.54	97.05	95.78	95.35	95.54	95.44
IK13	98.29	99.30	98.80	93.73	91.31	92.50	**95.06**	97.19	**96.11**	95.57	95.13	95.35
IK4	98.41	99.38	98.89	**94.17**	91.79	**92.96**	94.54	96.94	95.72	95.28	95.13	95.20

	ap			sb			atv		
	P	R	F1	P	R	F1	P	R	F1
All	84.59	81.48	83.00	80.59	82.61	81.59	85.33	84.26	84.79
CD10	83.46	80.50	81.95	80.64	82.53	81.58	85.12	85.20	85.16
CK7	84.77	80.99	82.84	80.34	84.23	82.24	83.49	82.27	82.88
CK2	80.03	71.27	75.40	78.24	81.56	79.86	83.96	79.12	81.47
CK4	84.39	80.62	82.46	80.39	83.78	82.05	85.02	80.38	82.63
CK3	84.45	81.36	82.88	79.32	83.48	81.35	84.00	77.65	80.70
CD14	84.31	**81.78**	83.03	80.62	82.83	81.71	87.11	85.10	86.09
CK13	85.21	81.36	**83.24**	**80.88**	**84.33**	**82.57**	87.29	**86.46**	**86.87**
CD21	85.06	80.75	82.85	80.26	83.08	81.65	86.80	84.89	85.84
IK20	**85.13**	80.13	82.56	80.80	84.11	82.42	87.62	84.68	86.13
IK6	84.82	81.60	83.18	80.03	82.63	81.31	83.93	80.59	82.23
IK13	84.26	80.81	82.50	80.27	83.28	81.75	**88.45**	85.20	86.80
IK4	84.33	80.93	82.60	79.36	82.16	80.74	82.73	79.43	81.05

Tablica 5.8: Statističke značajke korištenih testova za ispitivanje prilikom unakrsne provjere.

značajka	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
prosj. duljina	20.51	24.63	22.27	22.36	22.13	22.63	24.37	23.01	22.35	22.45
riječi	5086	6109	5522	5546	5487	5611	6044	5706	5542	5567
pojavnice	2014	2556	2465	2566	2277	2313	2468	2420	2481	2453
lema	1369	1687	1757	1813	1564	1584	1587	1678	1706	1632
MSD oznaka	306	314	310	307	288	308	327	311	297	313
imenica	31.69	32.98	34.68	33.93	34.06	32.69	34.63	33.25	34.48	33.68
glagol	16.36	14.90	14.74	14.41	13.85	15.63	14.73	14.39	14.24	14.14
pridjev	10.07	11.59	11.21	11.43	11.50	9.91	11.47	11.86	12.25	13.36
zamjenica	6.15	5.04	4.71	5.19	4.61	5.56	5.89	4.91	4.11	4.87
broj	0	2.46	2.99	2.87	2.48	2.35	1.85	2.66	2.45	2.17
prilog	3.71	3.65	3.01	2.79	3.61	3.24	3.06	3.61	3.09	3.59
prijedlog	9.26	9.21	10.34	9.90	9.97	9.55	10.11	9.90	9.46	9.45
veznik	5.92	5.53	4.74	4.72	4.94	6.01	5.45	5.15	4.84	5.57
čestica	0.81	0.65	0.34	0.29	0.20	0.59	0.31	0.23	0.27	0.36
interpunkcija	13.98	13.67	13.11	13.97	14.67	14.26	12.38	13.65	14.02	12.57
ostale	0.53	0.31	0.13	0.49	0.11	0.21	0.12	0.39	0.61	0.23
Obj	7.65	7.55	6.99	7.34	6.67	7.50	7.94	7.96	7.25	7.36
Pred	10.91	9.27	9.16	9.00	8.89	9.70	9.30	9.24	8.53	9.29
Adv	5.09	4.81	4.71	5.25	5.27	4.99	4.75	5.33	4.75	4.67
Pnom	2.16	1.85	1.23	1.35	1.13	2.01	1.77	1.95	1.57	1.51
Atr	22.69	26.58	26.15	25.12	27.63	25.04	27.81	25.99	27.44	28.96
Oth	2.38	1.94	1.30	1.64	1.80	1.87	1.24	1.61	2.08	2.07
Elp	0.85	0.59	0.33	0.58	0.36	0.66	0.36	0.75	0.56	0.68
Punc	13.15	13.18	12.77	13.42	14.23	13.72	12.33	13.46	13.95	12.32
Co	3.85	3.65	2.84	3.19	3.55	3.76	3.06	3.03	3.32	3.36
Aux	6.63	6.68	7.21	7.07	6.60	6.27	6.17	6.17	6.04	6.11
Ap	2.65	2.11	4.07	3.98	2.68	2.66	2.51	2.89	3.30	2.25
Sub	3.95	3.68	3.55	3.30	2.82	3.62	4.00	3.61	3.03	3.36
Sb	7.88	6.89	7.64	7.18	6.89	6.95	6.87	6.75	7.09	6.90
Atv	1.10	1.96	1.77	1.75	1.49	1.85	1.77	1.47	1.73	1.85

Tablica 5.9: Najviši postignuti LAS za svaki preklop i skup atributa za koji je postignut

	Gold MSD + lex	gold MSD + delex	System MSD + lex	System MSD + delex
1	83.61, CD10	80.74, CD10	82.11, IK13	79.02, CD21
2	82.47, CK3	79.87, all	81.45, CK3	77.97, all
3	83.23, CK13	79.95, CD14	82.09, CK13	78.75, CK13
4	83.96, all	80.57, all	81.88, all	78.30, all
5	84.07, CD21	80.65, CD21	82.06, CD21	78.28, CD10
6	83.86, CD14	79.92, CD14	81.53, CD14	78.73, CD14
7	83.79, CD10	80.69, CD14	81.61, IK13	79.11, all
8	83.07, IK20	79.75, CD21	81.87, IK20	77.81, IK20
9	83.56, CK7	79.91, CD10	81.56, CK7	78.46, CD10
10	83.63, CK4	80.56, CD21	81.49, CK4	78.36, CD14

Tablica 5.10: Unakrsna provjera potpunog i reduciranog skupa atributa za češki i slovenski jezik

	Gold MSD + lex			Gold MSD + delex		
	LAS %	UAS %	LA %	LAS %	UAS %	LA%
AllCze	76.04 ± 1.81	83.11 ± 1.60	84.11 ± 1.31	72.49 ± 1.53	80.55 ± 1.67	80.75 ± 1.63
CD10Cze	75.59 ± 2.18	82.65 ± 1.99	83.77 ±1.56	71.28 ± 1.39	79.17 ± 1.61	80.00 ± 1.29
AllSlo	86.02 ± 3.47	89.51 ± 2.03	88.89 ± 3.14	81.62 ± 4.09	86.90 ± 2.33	84.28 ± 3.73
CD10Slo	86.05 ± 3.30	89.73 ± 1.84	88.90 ± 2.97	80.14 ± 4.47	85.87 ± 2.89	83.20 ± 4.01
	System MSD + lex			System MSD + delex		
AllCze	72.52 ± 2.20	81.20 ± 1.84	80.93 ± 1.69	68.54 ± 2.22	78.27 ± 2.03	77.17 ± 2.13
CD10Cze	72.17 ± 2.60	80.98 ± 2.20	80.61 ± 1.89	67.58 ± 1.65	77.21± 1.60	76.60 ± 1.54
AllSlo	83.31 ± 2.86	89.01 ± 1.35	88.89 ± 3.14	78.81 ± 2.44	85.81 ± 1.25	81.86 ± 2.36
CD10Slo	83.58 ± 2.73	89.28 ± 1.23	86.49 ± 2.60	78.32 ± 2.65	85.68 ± 1.41	81.39 ± 2.50

6. Zaključak

Tekstove pisane prirodnim jezikom moguće je analizirati na više razina – na razini riječi i njenih karakteristika te na razini rečenice. Obje ove analize međusobno su isprepletene te jedna poboljšava kvalitetu druge. U ovom je radu razmatrana prvenstveno analiza rečenice postupkom parsanja ovisnosnim gramatikama čija se primjena pokazala uspješnima u Preksi prvenstveno jer sadrži dovoljnu količinu korisne informacije o strukturi rečenice, a da je kao formalizam dovoljno jednostavna da se može kombinirati s metodama strojnog učenja. Temeljno je pitanju prilikom parsanja rečenice što je potrebno parseru da bi kvalitetno obavio svoj posao. Ovisno o implementaciji, većina parsera osim same rečenice u postupku parsanja koristi i dodatne informacije kao što su podaci o morfološkim značajkama riječi te osnovnom obliku riječi. Pitanje koje se nameće jest koje od tih dodatnih informacija doprinose kvaliteti parsanja, a koje ju potencijalno narušavaju.

Tijekom izrade ovog rada, naglasak je bio na istraživanju utjecaja morfoloških značajki riječi na kvalitetu parsanja. Da bi ovakvo istraživanje bilo moguće, implementiran je optimizator koji sustavnim pretraživanjem prostora stanja mogućih kombinacija značajki temeljem odabrane mjere kvalitete rada parsera olakšava pronalazak kvalitetnih skupova morfoloških značajki. Optimizator je moguće koristiti za optimizaciju rada većine trenutno javno dostupnih parsera temeljenih na podacima. Optimizator trenutno radi pretraživanje prostora stanja korištenjem potpuno točnih morfoloških značajki i mogao bi se nadograditi da optimizaciju provodi uzimajući u obzir kvalitetu ovisnosnog parsanja kada se provede cijeli lanac analize rečenice.

Optimizatior je upotrebljen da bi se pronašao optimalan skup parametara parsanja hrvatskog jezika. Tijekom istraživanja korišteni su sljedeći javno dostupni parseri: parser Malt, parser MATE i parser MST. Pokazalo se da najbolje rezultate u parsanju hrvatskog jezika daje parser MATE. Što se tiče optimalnog skupa morfoloških značajki koje su potrebne parseru za rad, pokazalo se da redukcija skupa značajki ne može poboljšati prosječnu kvalitetu parsanja, no taj je skup moguće značajno smanjiti bez da se ta kvaliteta parsanja naruši. Ovo može značiti jednostavniju, bržu i jeftiniju izgrad-

nju novih skupova ovisnosnih stabala s obzirom da je dio morfološke analize rečenica nepotreban. Dobiveni je model ispitan na hrvatskom srodnim jezicima slovenskom i češkom, koji pokazuju slične rezultate kao i hrvatski.

S druge pak strane, pokazalo se da ne postoji jedan skup atributa koji bi bio najbolji u svim situacijama, odnosno za sve skupove rečenica, iz čega se može zaključiti da bi se najbolji rezultati parsanja mogli postići kombinacijom modela parsanja treniranih uz korištenje različitih skupova atributa. Da bi ovo bilo moguće potrebno je detaljnije istražiti koji su to uvjeti koji utječu na kvalitetu rada određenog modela.

LITERATURA

Željko Agić i Danijela Merkler. Three syntactic formalisms for data-driven dependency parsing of croatian. U Ivan Habernal i Václav Matoušek, urednici, *Text, Speech, and Dialogue*, svezak 8082 od *Lecture Notes in Computer Science*, stranice 560–567. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40584-6. doi: 10.1007/978-3-642-40585-3_70. URL http://dx.doi.org/10.1007/978-3-642-40585-3_70.

Željko Agić, Nikola Ljubešić, i Danijela Merkler. Lemmatization and morphosyntactic tagging of croatian and serbian. U *Proceedings of ACL*, stranice 48–57, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://aclweb.org/anthology//W/W13/W13-2408.pdf.

Miguel Ballesteros i Joakim Nivre. Maltoptimizer: An optimization tool for maltparser. U *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, stranice 58–62, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2380921.2380933.

Alena Böhmová, Jan Hajič, Eva Hajičová, i Barbora Hladká. The prague dependency treebank. U *Treebanks*, stranice 103–127. Springer, 2003.

Bernd Bohnet. Efficient parsing of syntactic and semantic dependency structures. U *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, stranice 67–72, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-29-9. URL http://dl.acm.org/citation.cfm?id=1596409.1596421.

Bernd Bohnet. Very high accuracy and fast dependency parsing is not a contradiction. U *Proceedings of the 23th international Conference on Computational Linguistics*, stranice 89–97, Beijing, August 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1873792.

Bernd Bohnet i Jonas Kuhn. The best of bothworlds – a graph-based completion model for transition-based parsers. U *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, stranice 77–87, Avignon, France, April 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E12-1009.

Bernd Bohnet i Joakim Nivre. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. U *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, stranice 1455–1465, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D12-1133.

Sabine Buchholz i Erwin Marsi. Conll-x shared task on multilingual dependency parsing. U *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, stranice 149–164, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1596276.1596305.

Xavier Carreras. Experiments with a higher-order projective dependency parser. U *In Proc. EMNLP-CoNLL Shared Task*, 2007.

Chih-Chung Chang i Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, Svibanj 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL http://doi.acm.org/10.1145/1961189.1961199.

Yoeng-Jin Chu i Tseng-Hong Liu. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396, 1965.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. U *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, stranice 1–8. Association for Computational Linguistics, 2002.

Michael A. Covington. A fundamental algorithm for dependency parsing. U *Proceedings of the 39th Annual ACM Southeast Conference*.

Koby Crammer i Yoram Singer. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991, 2003.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, i Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

Jason M Eisner. Three new probabilistic models for dependency parsing: An exploration. U *Proceedings of the 16th conference on Computational linguistics-Volume 1*, stranice 340–345. Association for Computational Linguistics, 1996.

Željko Agić. Pristupi ovisnosnom parsanju hrvatskih tekstova, 2012.

Tomaz Erjavec, Darja Fiser, Simon Krek, i Nina Ledinek. The jos linguistically tagged corpus of slovene. U *LREC*, 2010.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, i Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, Lipanj 2008. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1390681.1442794`.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, i Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. U *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, stranice 1–18, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-29-9. URL `http://dl.acm.org/citation.cfm?id=1596409.1596411`.

Tadao Kasami i Koji Torii. A syntax-analysis procedure for unambiguous context-free grammars. *Journal of the ACM (JACM)*, 16(3):423–431, 1969.

Sandra Kubler, Ryan McDonald, Joakim Nivre, i Graeme Hirst. *Dependency Parsing*. Morgan and Claypool Publishers, 2009. ISBN 1598295969, 9781598295962.

Ryan Mcdonald i Fernando Pereira. Online learning of approximate dependency parsing algorithms. U *In Proc. of EACL*, stranice 81–88, 2006.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, i Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. U *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, stranice 523–530, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220641. URL http://dx.doi.org/10.3115/1220575.1220641.

Ryan McDonald, Kevin Lerman, i Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. U *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, stranice 216–220, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1596276.1596317.

Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553, Prosinac 2008. ISSN 0891-2017. doi: 10.1162/coli.07-056-R1-07-027. URL http://dx.doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre. Non-projective dependency parsing in expected linear time. U *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, stranice 351–359, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL http://dl.acm.org/citation.cfm?id=1687878.1687929.

Joakim Nivre i Jens Nilsson. Pseudo-projective dependency parsing. U *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, stranice 99–106, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219853. URL http://dx.doi.org/10.3115/1219840.1219853.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, i Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. U *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, stranice 221–225, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1596276.1596318.

ITIROO Sakai. Syntax in universal translation. U *Proc. of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis*, 1962.

Daniel H Younger. Recognition and parsing of context-free languages in time< i> n</i>< sup> 3</sup>. *Information and control*, 10(2):189–208, 1967.

Arnold M Zwicky. Heads. *Journal of linguistics*, 21(01):1–29, 1985.

Optimizacija parametara ovisnosnog parsera za hrvatski jezik

Sažetak

Strojna sintaktička analiza rečenice ili parsanje preduvjet je za više razine strojne analize teksta. Postavlja se pitanje kako uspješno provesti parsanje morfosintaktički složenih jezika poput hrvatskog jezika.

U sklopu diplomskog rada provedeno je istraživanje utjecaja morfološke analize riječi u rečenici na rad postojećih ovisnosnih parsera te je implementiran sustav koji omogućava optimizaciju morfološkog skupa značajki. Implementirani je sustav primijenjen na hrvatski jezik kod kojeg se pokazalo da je moguće značajno smanjiti veličinu skupa morfoloških značajki bez narušavanja kvalitete rada parsera.

Razvijeni model ispitan je na hrvatskom srodnim jezicima slovenskom i češkom koji pokazuju slično ponašanje.

Ključne riječi: ovisnosno parsanje, optimizacija parametara, ovisnosne gramatike, morfološke značajke

Optimizing Dependency Parsing Parameters for Croatian Language

Abstract

Parsing is the neccessary step in any kind of higher-level text analysis. However, it is still unclear how to improve parsing of morphologically rich languages such as Croatian.

This works presents the analysis of the impact different morphological features have on the success of parsing and describes the system implemented in order to optimize the set of morphological features which improves the quality of parsing. The implemented system was then used to optimize the morphological feature set for parsing of Croatian language. This works shows that it is possible to significantlly reduce the size of the morphological feature set while still maintaining the quality of parsing.

The model was then tested on Czech and Slovene language, and these test show similar results.

Keywords: dependency parsing, dependency grammar, featura optimization, morphological features