Laboratorij za analizu teksta i inženjerstvo znanja – TakeLab

Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikrolektroniku, računalne i inteligentne sustave
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 287

Uparivanje koreferentnih imenovanih entiteta metodama strojnog učenja

Ivan Kmetović

Zagreb, lipanj 2011.

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

Zagreb, 21. veljače 2011.

DIPLOMSKI ZADATAK br. 287

Pristupnik: **Ivan Kmetović**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Uparivanje koreferentnih imenovanih entiteta metodama strojnog učenja**

Opis zadatka:

Razrješavanje koreferencije postupak je kojim se utvrđuje koji se izrazi u tekstu dokumenta odnose na isti izvanjezični entitet. Koreferentni izrazi mogu biti vlastita imena, imeničke fraze ili zamjenice. Razrješavanje koreferencije važan je zadatak u okviru obrade pridodnog jezika te nužan preduvjet za mnoge zadatke ekstrakcije informacija.
U okviru diplomskog rada potrebno je proučiti postupke za uparivanje koreferentnih imenovanih entiteta, s naglaskom na vlastita imena i imeničke fraze. Razraditi postupak za uparivanje koreferentnih vlastitih imena i imeničkih fraza u tekstovima na hrvatskome jeziku temeljen na nadziranom strojnom učenju. Razviti programsku implementaciju postupka te provesti vrednovanje na odgovarajućem ispitnom uzorku. Radu priložiti izvorni programski kod i ispitne uzorke.

Zadatak uručen pristupniku: 25. veljače 2011.
Rok za predaju rada: 10. lipnja 2011.

Mentor:

Prof.dr. sc. Bojana Dalbelo-Bašić

Djelovođa:

Doc.dr.sc. Domagoj Jakobović

Predsjednik odbora za diplomski rad profila:

Prof.dr.sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Svakim danom raspolažemo sve većom računalnom moći i pokušavamo rješavati različite izazove koje prije nismo mogli rješavati. Jedan od tih izazova je i obrada prirodnog jezika (engl. *Natural Language Processing*). U informacijskom dobu količina dostupnih informacija je ogromna i potrebna nam je pomoć računala da obradimo sve te dostupne informacije. Obrada prirodnog jezika predstavlja jedan od najtežih područja umjetne inteligencije.

Neki primjeri problema kod obrade prirodnog jezika su: strojno prevođenje, parsiranje prirodnog jezika, automatska izrada sažetka teksta, prepoznavanje imenovanih entiteta, označavanje vrsta riječi, generiranje prirodnog jezika, razrješavanje koreferencija, izrada sustava za odgovaranje na pitanja, prepoznavanje govora itd.

U ovom radu bavimo se problemom razrješavanja koreferencije (engl. *Coreference Resolution*). Taj problem možemo smjestiti u područje ekstrakcije informacija iz teksta (engl. *information extraction*). To je postupak kojim se utvrđuje koji se izrazi u tekstu dokumenta odnose na isti izvanjezični entitet s obzirom na to da se u tekstovima na iste izvanjezične entitete najčešće ne referiramo istim nazivima već zamjenicama, skraćenicama, dijelovima naziva i sl. Primjerice, ako neka imamo dokument u čijem tekstu se spominje *predsjednik RH Ivo Josipović*, to bio primjer korištenja imenovanog izraza, na taj entitet se može referirati zamjenicom *on* ili nominalnim izrazom *predsjednik*. Koreferentni izrazi mogu biti zamjenice, imeničke fraze ili vlastita imena. Ovaj rad koncentrira se na uparivanje imenovanih entiteta.

Rješavanje problema razrješavanja koreferencije preduvjet je za rješavanje složenijih problema obrade prirodnog jezika, prvenstveno iz područja ekstrakcije informacija. Primjeri problema čije se rješavanje olakšava razrješavanjem koreferencije su: sustavi koji odgovaraju na pitanja, problem ekstrakcije citata iz teksta i sl. Tako bi se u problemu odgovaranja na pitanja mogle pratiti informacije o izvanjezičnim entitetima na koje se različito referenciramo u datom tekstu. Također, potrebno je prepoznati na koje se izvanjezične entitete referencira korisnik u svojim upitima.

Ova rad se koncentrira na uparivanje koreferentnih imenovanih entiteta. Za nave-

deni zadatak koristit će se pristupi nadziranog strojnog učenja. Sustavi koji razrješavaju zamjenice imaju visoku točnost. Kada proširimo problem i na imeničke fraze ili vlastita imena, točnost sustava pada. Zbog toga se pokušava poboljšati uparivanje koreferentnih imenovanih entiteta.

U radu su analizirane značajke i utvrdilo se koje značajke treba koristiti da bi što efikasnije riješili zadani problem. Definirani su različiti skupovi značajki odnosno konfiguracije. Cilj ovog diplomskog rada proučiti je kako se različite konfiguracije ponašaju u sustavima koji će raditi s tekstovima na hrvatskom jeziku. Implementiran je postupak nadziranog strojnog učenja temeljen na pravilima. Osim toga, u procesu evaluacije napravljeni su eksperimenti i s nekim drugim tehnikama nadziranog strojnog učenja.

U sljedećem poglavlju dan je pregled područje razrješavanja koreferencije s važnijim srodnim radovima. Treće poglavlje sadrži opis metoda strojnog učenja pomoću kojih će se pokušati riješiti problem. U četvrtom poglavlju opisuje se priprema skupa za učenje, dok se u petom poglavlju analiziraju značajke koje se koriste u razrješavanju koreferencije. Potom slijedi opis programske implementacija i evaluacija rezultata.

2. Pregled područja

Razrješavanje koreferenci postupak je u kojem se uparuju različiti izrazi koji se odnose na isti izvanjezični entitet (Clark i González-brenes, 2008). Takvi parovi sastoje se od antecedenta i anafore. Anaforom nazivamo koreferencu na antecedent koji se odnosi na određeni izvanjezični entitet. Tako u rečenici: *Ivo Ivić kasni, ali on će doći*, antecedent je *Ivo Ivić*, a anafora je zamjenica *on*. Slaganjem koreferentnih parova koji se odnose na isti izvanjezični entitet dobivamo koreferetni lanac. Koreferentne izraze možemo podijeliti u tri kategorije (Clark i González-brenes, 2008):

- zamjenice,
- nominalni izrazi,
- imena.

U tablici 2.1 dani su primjeri izraza u tekstu za izvanjezični entitet Pero Perić.

Tablica 2.1: Različite vrste izraza koje se odnosi na isti izvanjezični entitet

Vrsta izraza	Primjer
Zamjenice	On, njega
Nominalni izraz	Čovjek u crnom odijelu
Imenovani izraz	Pero Perić, gospodin Perić

U postupku razrješavanju koreferencije želimo razrješavati izraze unutar iste rečenice i među različitim rečenicama. Takvi postupci imaju ključnu ulogu u razumijevanju teme teksta.

Zadatak razrješavanja koreferencije definiran je na šestoj konferenciji MUC[1] (engl. *Message Understanding Conference*), održanoj 1995. godine. Na istoj konferenciji definiran je i zadatak prepoznavanja imenovanih entiteta (engl. *Named Entity Recognition*)

[1]Niz konferencija koje su poticale razvoj novih i boljih metoda u području ekstrakcije informacija (engl. *information extraction*). Konferenciju organizira agencija DARPA pri američkom ministarstvu obrane.

u tekstu čije rješavanje obično prethodi problemu razrješavanja koreferenci. Na konferenciji se definirao i korpus tekstova na kojem su se rješavali zadani problemi. Dvije godine kasnije održana je sedma konferencija MUC, na kojoj je također predstavljeno nekoliko radova iz područja razrješavanja koreferencije, kao i novi korpus. Spomenute korpuse znanstvenici i danas intenzivno koriste za ispitivanje raznih metoda razrješavanja koreferencije na engleskom jeziku.

Robustan i točan alat za razrješavanje koreferencije predstavlja bitan element u mnogim zadacima obrade prirodnog jezika. Ipak, ne mogu se sve vrste anafora jednako uspješno razrješavati. Tipičan sustav za razrješavanje zamjenica ima uspješnost između 85 i 90% (Mitkov i Sb, 1999), dok je općenita uspješnost razrješavanja koreferenci mnogo lošija. Tako, primjerice, sustav (Soon et al., 2001) ima F-mjeru 60,4% na ispitnom skupu MUC-7.

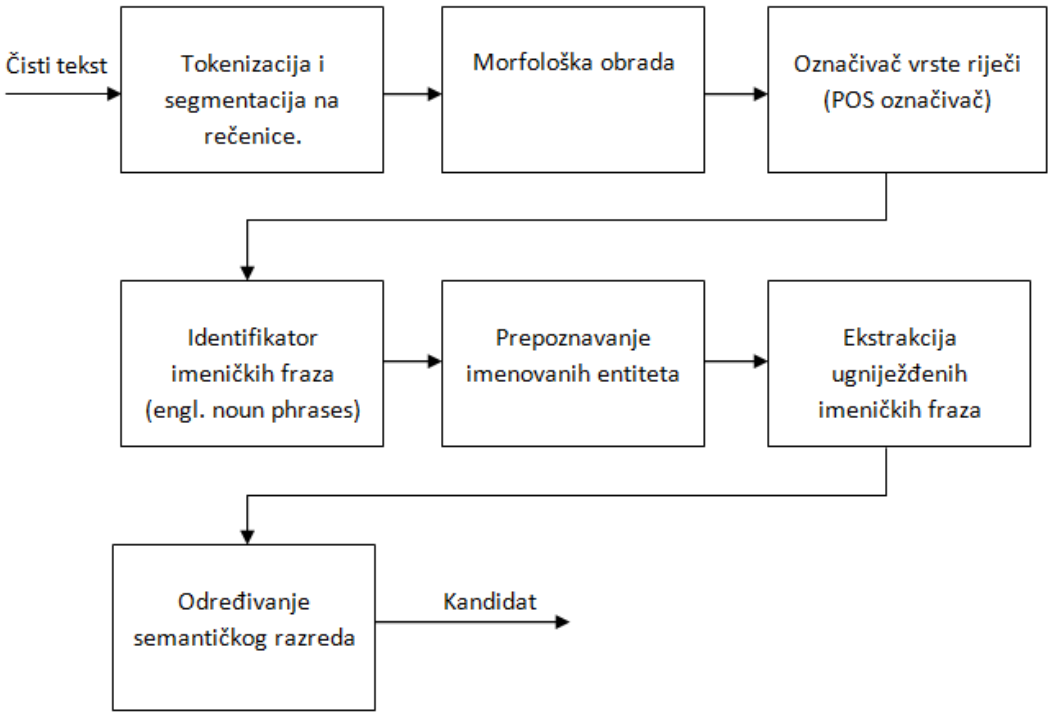
Navedeni rezultati upućuju da je potrebno poboljšati razrješavanje koreferencije među vlastitim imenima i imeničkim frazama. U tablici 2.2. predstavljeni su neki tipični problemi koreferencije kod vlastitih imena.

Tablica 2.2: Primjeri koreferentnih vlastitih imena

Antecedent	Anafora
Hrvatski zavod za mirovinski osiguranje	HZMO
F - 14	f14
Predsjednik Republike Hrvatske Ivo Josipović	Predsjednik Josipović

U (Soon et al., 2001) razvijen je postupak zasnovan na strojnom učenju, koristeći pripremljen korpus, za razrješavanja koreferencije. Postupak je zahtijevao relativno malen korpus za učenje koji je sadržavao unaprijed označene koreferentne lance. Cijeli niz alata (engl. *pipeline*) korišten je da bi se prepoznale imeničke fraze, zamjenice i ostali tekstovni elementi koji se referenciraju na izvanjezični entitete. Iz označenog korpusa generirani su parovi elemenata koji se odnose na izvanjezične entitete (engl. *markables*), za svaki par definirano je odnosi li se na isti izvanjezični entitet odnosno je li koreferentan. Potom je skup za učenje predan algoritmu strojnog učenja koji ima zadatak izgraditi klasifikator. Autori su koristili klasifikator izgrađen algoritmom C5, naprednijom verzijom algoritma C4.5 (Quinlan, 1993). Za razrješavanje koreferencije u novim neviđenim tekstovima koristi se spomenuti niz alata kojim se prepoznaju parovi kandidati za koreferenciju, a potom koristimo izgrađeni klasifikator da odredi radi li se stvarno o koreferentnim izrazima. Rad (Soon et al., 2001) kao i

brojni drugi radovi pokazuju da je vrlo korisno imati određene jezične alate na raspolaganju (vidi sliku 2.1), prvenstveno zbog automatskog prepoznavanja imeničkih fraza i zbog automatskog generiranja brojnih značajki parova kandidata koje se koriste u klasifikatoru na odgovor na pitanje radi li se o koreferentnim izrazima. U nedostatku određenih jezičnih alata prisiljeni smo ručno izvršavati određene zadatke, što treba uzeti u obzir. Kritični dio postupka je odabir podobnih kandidata za koreferenciju, što bitno utječe na odziv cjelokupnog sustava za razrješavanje koreferencije. U radu (McEnery et al., 1997) proučava se utjecaj udaljenost među izrazima kandidatima za koreferenciju. Pokazano je da utjecaj značajke udaljenosti igra važnu ulogu kod razrješavanja koreferencije među zamjenicama, ali kad govorimo o vlastitim imenima, na koje se stavlja naglasak u ovom radu, značajka udaljenosti se nije pokazala važnom.



Slika 2.1: Primjer sustava za razrješavanje koreferencije Soon et al. (2001)

Zahvaljujući konferencijama MUC, pogotovo MUC-6 i MUC-7 gdje je precizno definiran zadatak razrješavanja koreferenci, dostupni su korpusi s unaprijed označenim kandidatima za koreferenciju. Na žalost, vrlo malo je korpusa na drugim jezicima koji su prilagođeni datom zadatku. Sustav (Soon et al., 2001) je na skupu MUC-6 imao odziv 58,6%, a preciznost 67,3%. Odgovarajuća F-mjera je iznosila 62,6%. Za korpus MUC-7 odziv je iznosio 56,1%, a preciznost 65,5%. F-mjera bila je 60,4%. Rezultati su bili na razini rezultata sustava koji su koristili nenadzirano strojno učenje

(postupke temeljene na grupiranju). Tri značajke su se pokazale najznačajnijim: STRMATCH, ALIAS, APPOSITIVE. Značajka STRMATCH jednostavno je uspoređivanje niza znakova, prethodi joj postupak normalizacije u kojem se eliminiraju zamjenice i članovi (primjer engleskog jezika, *a*, *an*, *the*). Značajka ALIAS je prilično složena. Ako su oba izraza u paru imenovani entiteti (osobe, organizacije, dan, i tako dalje) ova značajka definira odnose li se izrazi na iste izvanjezične entitete. Ovisno o tome o kojoj se vrsti entiteta radi, primjerice radi li se o osobi ili organizaciji, primjenjuju se različiti postupci. Značajka APPOSITIVE definira je li prvi izraz u paru apozicija drugog izraza u paru.

U radu (Uryupina, 2004) poseban je naglasak stavljen na vlastita imena. Postupak određivanja koreferencije između dva izraza pretežno sastavljena od vlastitih imenica rastavljen je na tri dijela: normalizaciju, izbor znakovnog niza i samu usporedbu. U postupku normalizacije eliminiraju se određeni znakovi poput članova, interpunkcijskih znakova. U postupku izbora znakovnog niza odabire se najinformativniji dio izraza. To može biti prva ili posljednja riječ u izrazi, posljednja imenica u izrazu, najčešće korištena imenica. Posljednji dio se odnosi na samu usporedbu kandidata nad kojim su provedeni postupci normalizacije i izbora znakovnog niza. Primijenjene su razne metode, od najelementarnije provjere jesu li nizovi identični do različitih mjera sličnosti. U evaluaciji je korišten MUC program (Vilain et al., 1995) u kojem se ne uspoređuju individualni parovi nego cijeli koreferentni lanci. Za svaku anaforu, gledaju se svi kandidati antecedenti (prethodna vlastita imena u ovom slučaju). Svi dobiveni parovi se predaju klasifikatoru Ripper (Cohen, 1995). Ako klasifikator odredi da su kandidati koreferentni, oni se dodavaju u pripadajući koreferentni lanac. Potom nastavljamo sa sljedećom anaforom. U radu se eksperimentiralo s mnogo različitih konfiguracija (kombinacija značajki). Rezultati izraženi kao F-mjera kretali su se između 71,9% do 80,5%.

U radu (Strube et al., 2002) predstavljena je značajka MED (engl. *Minimum Edit Distance*). Autori su prethodno izradili sustav za razrješavanje koreferencije i postigli odlične rezultate sa zamjenicama, dobre rezultate s vlastitim imenima i nezadovoljavajuće rezultate s imeničkim frazama koje definiraju neki entitet (primjer u tablici 2.1). Želeći unaprijediti razrješavanje anaforičkih veza kada se radi o vlastitim imenima i definirajućim imeničkim frazama iskoristili su mjeru minimalne udaljenosti (Wagner i Fischer, 1974). Postigli su značajno bolji odziv sustava, izgubivši malo preciznosti. To predstavlja iskorak jer značajka nije zahtjevna za izračunavanje i nezavisna se o jeziku i domeni teksta. F-mjera se poboljšala za 11% kada se u obzir uzmu vlastita imena i 18% za definirajuće imenske fraze. Uzimajući sve vrste anafora u obzir F-mjera se

poboljšala od 59,97% do 67,98%, što predstavlja napredak od 8%.

Uparivanjem koreferentnih vlastitih imena u pravnoj domeni bavio se (Branting, 2002). Cilj je bio razviti sustav koji će omogućiti detekciju sukoba interesa u pravnim slučajevima. Prepoznato je devet vrsta varijacija koje otežavaju uparivanje vlastitih imena poput interpunkcijskih znakova, razmaka, skraćenica, pravopisnih pogrešaka, permutacija ili nedostajanje riječi u nazivu. Autor postupak rastavlja u tri dijela: normalizacija, indeksiranje i procjena sličnosti. Normalizacija je postupak u kojem se znakovni nizovi čiste u svrhu boljeg iskorištavanja algoritama usporedbe. Postupci normalizacije mogu uključivati eliminiranje interpunkcijskih znakova (primjerice crtica, pa se F-16 pretvara u F16), pretvaranje svih slova u mala slova, filtriranje zaustavnih riječi. Indeksiranje je postupak u kojem se smanjuje broj potrebnih uspoređivanja znakovnih nizova da bi se procijenila njihova sličnost. Korišteni su i fonetski algoritmi koji indeksiranje rade u skladu s engleskim izgovorom imena. Prvenstveno se odnosi na organizaciju znakovnih nizova u strukture koje su pogodne da se reducira broj uspoređivanja poput tablica raspršenog adresiranja. U radu je predstavljeno 16 algoritama za usporedbu vlastitih imena. Najbolji algoritmi RED_WS_Approx i RED_WS_EQ imali su F-mjeru oko 85%. Oba algoritma koriste fonetski algortiam *Redudant*, uspoređuju se riječ po riječ u nazivu, a razlike su da za konačnu usporedbu prvi algoritam koristi Levenshteinovu udaljenost, a drugi algoritam provjerava jesu li traženi nizovi identični.

U ovom radu uparivaju se koreferentni imenovani entiteti na hrvatskom jeziku. Svi prethodno navedeni sustavi napravljeni su za druge jezike, prvenstveno engleski jezik. Preuzet će se neke značajke i postupci koji su korišteni u navedenim radovima i ispitati njihova korisnost u rješvanju problema koreferencije imenovanih entiteta na hrvatskom jeziku.

3. Učenje temeljeno na pravilima

3.1. Uvod

Uparivanje koreferentnih imenovanih entiteta svodi se na problem binarne klasifikacije. Uparujemo sva moguća spominjanja imenovanih entiteta. Jedan primjer skupa odnosno par spominjanja može pripadati klasi koreferentnih parova ili klasi parova koji nisu koreferentni.

Tipična primjena učenja binarne klasifikacije uključuje (Suh, 2011):

1. unaprijed određene dvije klase,
2. diskretnu domenu podataka,
3. dovoljan broj primjeraka za učenje,
4. vrijednosti atributa mogu biti ili diskretne ili numeričke.

Metoda koja je odabrana za rješavanje problema uparivanja koreferentnih imenovanih entiteta temeljena je na automatskoj izgradnji pravila. Navedeni postupak spada u metode nadziranog strojnog učenja i pokazao se vrlo uspješnim u rješavanju problema uparivanja koreferentih imenovanih entiteta na engleskom jeziku (Uryupina, 2004). Sustavi temeljeni na pravilima omogućuju brzu i uspješnu izgradnju modela koji ćemo koristiti.

Imamo poznatu ciljnu varijablu i ona je u obliku dvije klase. Zanimaju nas modeli (pravila) koji razdvajaju te klase. Sustavi koji uče skup pravila po kojima će obavljati klasifikaciju primjera imaju niz poželjnih svojstava. Pravila su relativno jednostavna i ljudi ih mogu lako razumjeti. Vrlo često sustavi temeljeni na pravilima postižu bolje rezultate nego sustavi temeljeni na stablima odluke. Jedan nedostatak sustava temeljenih na pravilima relativno je loše skaliranje u odnosu na veličinu skupa za učenje i osjetljivost na šum[1] u podacima. S obzirom na to da su upravo takvi skupovi po-

[1] Šum u podacima uključuje sve neispravnosti koje nisu sistemskog karaktera, poput subjektivne procjene vrijednosti atributa, nepreciznih mjerenja i sl.

dataka vrlo česti u realnim primjenama, uložen je značajan trud da bi se ublažio ovaj nedostatak.

Postoji mnogo algoritama koji se koriste za učenje klasifikacije, a temelje se na ekstrakciji pravila. Možemo ih svrstati u dva glavna pristupa:

1. Tehnika "podijeli, pa vladaj": ovaj pristup uključuje rješavanje problema odozdo prema dolje. Ideja je da se stvore pravila za klasifikaciju organizirajući sve primjere skupa za učenje u stablo odluke, a ta organizacija se temelji na nizu testova koji se provode nad svakim atributom skupa za učenje. U ovom pristupu biramo jedan atribut skupa za učenje i provodimo test i ovisno o rezultatima dijelimo početni skup za učenje na manje podskupove, zatim rekurzivno provodimo testove nad svakim od podskupova. Postupak ponavljamo sve dok ne dobijemo čiste podskupove, odnosno sve dok ne dobijemo podskupove u kojima sve instance pripadaju istoj klasi. Ovaj proces je zapravo proces heurističkog traženja svih mogućih pravila klasifikacije. Pravila dobivamo tako da prođemo sve puteve od korijena dobivenog stabla do listova stabla. Primjeri algoritama koji koriste ovaj pristup su: Cart, ID3, C4.5, C5.0;

2. Drugi pristup uključuje generiranje pravila na način da se pokrije što više instanci određene klase bez da se uspostavlja stablo odluke. Nakon što stvorimo jedno pravilo, sve instance pokrivene tim pravilom se izdvajaju i nastavljamo stvarati pravila na preostalom skupu instanci. Ovakav pristup je nešto neprecizniji, ali dosta brži jer ne gradimo cijelo stablo. Primjeri algoritama koji koriste navedeni pristup su: Prism, Induct, IREP, RIPPER.

Ciljevi navedenih pristupa su da se precizno i efikasno izvuče znanje o klasifikaciji iz zadanog skupa za učenje. Navedeno znanje se može prikazati na dva način što se vidi iz prethodno definiranih pristupa. Možemo koristiti klasifikacijska pravila ili stabla odluke.

Klasifikacijska pravila su najjednostavnija za prikazivanje znanja. Pravila uključuju jedan ili više testova nad atributa koji se koriste u skupu za učenje i predstavljaju povezanost instance s određenom klasom. Pravila se navode u sljedećoj formi:

```
Ako je atribut X=x i atribut Y=y, onda
instanca Z pripada klasi z.
```

Ciljni atribut može imati samo dvije klase. Kad primjenjujemo pravila odlučujemo na osnovni je li određeni uvjet postavljen nad nekim atributom ispunjen ili ne. Prikaz pravilima je koncizniji.

Stabla odluke predstavljaju integrirano znanje o klasifikaciji. Svaki čvor stabla predstavlja jedan test nad jednim ili više atributa (odgovara uvjetu u pravilu). Listovi stabla predstavljaju klase. Instancu provedemo kroz stablo, izvršavajući testove na svakom čvoru i vrsta lista do koje dođemo nam definira klasu instance. Stabla sadrže više znanja, potpun pogled na klasifikacijske informacije u domeni, ali prostorno i vremenski su zahtjevniji.

Čest problem koji se javlja i kod stabla odluka i kod pravila je prenaučenost hipoteza (Mitchell, 1997). Neka je skup pravila S. Pravilo p je prenaučeno ako postoji pravilo $p' \in S$ takvo da pravilo p ima manju pogrešku nego p' na primjerima za učenje, ali p' ima manju pogrešku nego p na cijelom prostoru primjera za učenje. U (Quinlan, 1987) istražena je mogućnost podrezivanja stabala u svrhu dobivanja kvalitetnijih hipoteza. Slične metode iskorištene su iza poboljšavanje postupaka temeljenih na učenju pravila.

3.2. REP

REP (engl. *Reduced Error Pruning*), IREP (engl. *Incremental Reduced Error Pruning*) (Fürnkranz i Widmer, 1994) i RIPPER (engl. *Repeated Incremental Pruning to Produce Error Reduction*) (Cohen, 1995) algoritmi su koji su pokušavali riješiti problem prenaučenosti pravila. Postupci koji su se koristili ranije koristili su isti skup za stvaranje i evaluiranje pravila što je dovodilo do prenaučenosti.

REP je algoritam koji se koristi u sustavu za učenje pravila u svrhu povećavanja preciznosti stvorenog skupa pravila. Algoritam REP je iz skupine algoritama koji stvore skup pravila na temelju skupa za učenje koji često bude prenaučen. Nakon što se stvori prenaučeni skup pravila, ta pravila se podrezuju dok se ne postigne zadovoljavajuća razina točnosti. Općenito postupak podrezivanja smanjuje učestalost pogrešaka na novim neviđenim podacima, pogotovo kad je prisutan značajan šum u podacima. Cilj podrezivanja pravila je odstraniti dijelove pravila koji previše generaliziraju nad skupom za učenje. U praksi, to najčešće dovodi to poboljšavanja točnosti pravila.

U algoritmu REP, skup za učenje dijeli se u dva disjunktna podskupa:

1. skup za rast (engl. *growing set*),
2. skup za rezanje (engl. *pruning set*).

U početnoj fazi stvara se skup pravila koristeći neku pohlepnu heurističku metodu nad skupom za rast. Svako pravilo postupno se gradi dodavanjem uvjeta. Najčešći

kriterij koji se koristi prilikom stvaranja pravila jest kriterij informacijske dobiti. Informacijskom dobiti mjerimo količinu informacije za klasificiranje koja se dobije dodavanjem uvjeta u pravilo koje gradimo. Vrijednost informacijske dobiti nakon što pravilu R dodamo uvjet L računamo na sljedeći način:

$$Info\_Gain(L, R) = t(log_2 \frac{p_1}{p_1 + n_1} - log_2 \frac{p_0}{p_0 + n_0}) \tag{3.1}$$

Označimo s R' pravilo koje je nastalo nakon što smo pravilu R dodali uvjet L. U formuli 3.1 p_0 je broj pozitivnih primjera koje pokriva pravilo R, n_0 je broj negativnih primjera koje pokriva pravilo R, p_1 je broj pozitivnih primjera koje pokriva pravilo R', n_1 je broj negativnih primjera koje pokriva pravilo R', a t predstavlja broj pozitivnih primjera koji su još uvijek pokriveni nakon što smo dodali uvjet L pravilu R.

Ovaj početni skup pravila obično je prenaučen, odnosno početni skup pravila je puno veći i složeniji nego konačni optimizirani skup pravila. Nakon što se stvori početni skup pravila, algoritam REP koristi tehnike za podrezivanje. Početni se skup pravila primjenom operatora za podrezivanje postupno smanjuje. U svakom koraku postupka podrezivanja odabire se onaj operator koji najviše smanjuje pogrešku na skupu za podrezivanje. Drugim riječima, svako pravilo ispitujemo na skupu za podrezivanje, a potom ga modificiramo tako da na bude što točnije na skupu za podrezivanje. Uobičajen operator podrezivanja jest odstranjivanje uvjeta iz pravila ili čak cijelog pravila. Postupak podrezivanja staje kada dodatna primjena operatora podrezivanja smanjuje točnost pravila na skupu za podrezivanje.

Algoritam REP poboljšava točnost stvorenih pravila, čak i kada se koriste podaci koji sadrže šum. Najveći nedostatak algoritma jest složenost. Ako radi na velikom skupu za učenje koji sadrži dosta šuma, složenost algoritma REP iznosi $O(n^4)$ (gdje je n broj primjera u skupu za učenje). Algoritmom IREP se pokušalo riješiti navedeni nedostatak.

3.3. IREP

Algoritam IREP koristi temeljne koncepte iz algoritma REP uz modifikaciju načina učenja pravila i novu tehniku podrezivanja. Skup za učenje dijeli se na skupa za rast i skup za podrezivanje, kao i kod algoritma REP. Dvije trećine skupa za učenje čini skup za rast, a jedna trećina se koristi za podrezivanje (Fürnkranz i Widmer, 1994). U algoritmu IREP objedinjujemo postupke ranog podrezivanja (engl. *prepruning*) i kasnog podrezivanja (engl. *post pruning*). Ova integracija eliminira potrebu za počet-

nim stvaranjem prenaučenog skupa pravila. Dakle, kod algoritma IREP odmah dobivamo konačni podrezani skup pravila, a taj skup gradimo pravilo po pravilo. Nakon što smo određeno pravilo stvorili koristeći skup za rast, odmah ga podrezujemo koristeći skup za podrezivanje. Nakon što smo završili s postupkom podrezivanja, iz skupa za rast i iz skupa za podrezivanje brišemo sve primjere koje smo koristili da dobijemo konačno pravilo. Za razliku od algoritma REP, preostali skup ponovno se raspodjeljuje da bi se izbjegli mogući problemi zbog loše podjele skupa (engl. *bad split*). Proces stvaranja pravila traje sve dok ima pozitivnih primjera u skupu za učenje ili dok ne stvorimo pravilo koje ima visoku učestalost pogreške. Algoritam IREP je na istoj razini kad govorimo o točnosti u odnosu na algoritam REP, a značajno je brži. Složenost algoritma IREP je $O(n \log^2 n)$ (Cohen, 1995), što je značajno poboljšanje u odnosu na složenost $O(n^4)$ koju je imao algoritam REP.

U nekoliko koraka bit će pokazano da je složenost algoritma IREP $O(n \log^2 n)$:

1. Svako pravilo će u prosjeku imati $\log n$ uvjeta zato jer će svaki uvjet pokriti otprilike pola primjera za učenje;

2. Svaki od tih uvjeta mora biti testiran jedanput na n primjeraka u skupu za stvarajne pravila. Dakle, za svako pravilo trošak dodavanja uvjeta je n;

3. Imamo $\log n$ uvjeta po pravilu, svako testirano n puta, što znači da za izgradnju jednog pravila vrijedni $n \log n$;

4. Tijekom postupka podrezivanja, svaki od $\log n$ uvjeta mora biti testiran na n primjeraka u skupu za podrezivanje, dok ne dobijemo konačno pravilo odnosno najviše $\log n$ puta. Tako dobivamo da je ukupan trošak podrezivanja pravila $n \log^2 n$;

5. Vrijedi: $O(n \log n) + O(n \log^2 n) = O(n \log^2 n)$;

6. Ako je veličina konačnog skupa pravila konstanta (ne ovisi o n), sveukupna složenost IREP algoritma iznosi $O(n \log^2 n)$.

Cohenova implementacija postupka stvaranja jednog pravila po algoritmu IREP počinje s praznim pravilom i razmatra dodavanje jednog od sljedećih oblika uvjeta:

- $A_n = v$,
- $A_c <= \psi$,
- $A_c >= \psi$,

gdje je A_n nominalni atribut, A_c kontinuirana varijbla i ψ neka vrijednost koja se javlja u skupu za učenje.

Navedena implementacija dodaje uvjete po kriteriju maksimiziranja informacijske dobiti sve dok pravilo ne pokriva negativne primjere iz skupa za rast. Pravila se prestaju stvarati kad je učestalost pogreške posljednjeg stvorenog pravila veća od 50% (u originalnoj implementaciji algoritma IREP dodavanje pravila prestaje kad je točnost posljednjeg stvorenog pravila manja od točnosti praznog pravila). Slijedi pseudokod algoritma IREP:

Algoritam 1 IREP(Pos, Neg)

Require: Pos: Pozitivni primjeri, Neg: Negativni primjeri
  Skup pravila $= \{\}$
  while Pos $\neq \{\}$ **do**
    {stvaranje i podrezivanje novog pravila}
    podijeli(Pos, Neg) na (rastPos, rastNeg) i (podrezivanjePos, podrezivanjeNeg)
    pravilo = stvoriPravilo(rastPos, rastNeg)
    pravilo = podreziPravilo(podrezivanjePos, podrezivanjeNeg)
    if pogreška pravila na skupu (podrezivanjePos, podrezivanjeNeg) veća od 50% **then**
      return Skup pravila
    else
      dodaj pravilo u Skup pravila
      odstrani sve primjere pokrivene pravilom iz skupa rastPos, rastNeg
    end if
  end while
  return Skup pravila

3.4. Preinake algoritma IREP

Cohen je uveo neke promjene u podrezivanju s ciljem da poboljša točnost algoritma IREP. Razmatrao je brisanje bilo kojeg niza uvjeta iz pravila. Odluku je temeljio na funkcijama definiranim u tablici 3.1.

Cohenova početna implementacija algoritma IREP zaustavljala je stvaranje pravila ako je posljednje stvoreno pravilo imalo očekivanu pogrešku veću od 50%. Taj uvjet je preuzet i u implementaciji algoritma IREP2. Eksperimentiranjem je ustanovljeno da je

Tablica 3.1: Vrednovanje pravila u algoritmu IREP

$v*(Pravilo, PodrezivanjePos, PodrezivanjeNeg) = \frac{p-n}{p+n}$	IREP*
$v(Pravilo, PodrezivanjePos, PodrezivanjeNeg) = \frac{p+(N-n)}{P+N}$	IREP
$v'(Pravilo, PodrezivanjePos, PodrezivanjeNeg) = \frac{p}{p+n}$	IREP2

P je broj pozitivnih primjera,
N je broj negativnih primjera,
n je broj negativnih primjera pokrivenih promatranim pravilom,
p je broj pozitivnih primjera pokrivenih promatranim pravilom.

navedeno pravilo osjetljivo na male i srednje velike skupove za učenje, pogotovo kad ciljni koncept pokriva mnogo manjih pravila (manji u smislu da jedno pravilo pokriva malen broj primjeraka).

3.5. RIPPER

Cohen je predložio novi uvjet za zaustavljanje stvaranja pravila. Uveo je princip minimalne duljine opisa (engl. *Minimum Description Length, MDL*) (Mitchell, 1997). Predloženi princip (Quinlan, 1995) je pokušao iskoristiti u svom algoritmu za izgradnju stabala odluke (Quinlan, 1993). Tu mjeru koristio je za određivanje složenosti stvorenog IREP pravila.

Princip minimalne duljine opisa je formalizacija Occamove oštrice (Mitchell, 1997) u kojoj se smatra da je najbolja hipoteza za zadani skup podataka ona koji dovodi do najbolje kompresije datog skupa. Svaki skup podataka može biti opisan nizom simbola neke konačne abecede (npr. binarne). Ideja u pozadini principa MDL je u tome da svaku pravilnost u podacima možemo iskoristiti da bi skratili niz znakova kojim opisujemo podatke.

Brisanjem jednog uvjeta u pravilu ili cijelog pravila može u nekim slučajevima povećati učestalost pogreške, ali istodobno se smanjuje veličina skupa pravila. Možemo opisati ukupnu duljina opisa u dvije komponente:

1. duljina pravila koji klasificiraju primjere,

2. duljina onih podataka koje skup pravila pogrešno klasificira.

Skup pravila za određivanje pripadnosti određenoj klasi nazivamo teorijom. Za određivanje klasifikacije potrebna nam je teorija i iznimke (skup primjera koje zadani

skup pravila pogrešno klasificira). Cilj principa MDL je pronaći najbolji podksup pravila za određenu klasu, a to je onaj koji minimizira duljinu opisa teorije i iznimaka.

Prvo kodiramo svako pravilo računajući informacijske bitove, a to radimo na temelju svih uvjeta od kojih je pravilo sastavljeno. Potom oduzimamo vrijednost informacije o redoslijedu uvjeta u pravilu, a to iznosi $\log_2(k!)$, gdje je k broj uvjeta u pravilu. Tu vrijednost oduzimamo jer je redoslijed uvjeta nebitan (radi se o konjukciji uvjeta). Duljinu teorije dobijemo kada zbrojimo duljinu svih pravila i oduzmemo vrijednost informacije o redoslijedi pojedinih pravila. Ta vrijednost iznosi $\log_2(R!)$, gdje je R broj pravila u podskupu. Te bitove oduzimamo jer redoslijed pravila nije bitan jer svi klasificiraju primjere u istu klasu.

Iznimke se kodiraju tako da zbrajamo informacijske bitove lažno pozitivnih primjera i lažno negativnih primjera. Ako pravilo pokriva r od n primjera za učenje, ako oznakom fn označimo lažno negativne primjer, a oznakom fp lažno pozitivne primjere, iznimke kodiramo s:

$$\log_2 \binom{r}{fp} + \log_2 \binom{n-r}{fn} \tag{3.2}$$

Informacija o cijelom podskupu pravila je zbroj teorijskih bitova i bitova koji opisuju iznimke:

$$bitovi\_iznimki + W \cdot teorijskibitovi \tag{3.3}$$

Faktor W iz izraza 3.3 je manji od 1. Njegova vrijednost ovisi o tome koliko su korištene značajke redudantne (Quinlan, 1993).

Zadatak je pronaći podskup pravila S za klasu C koji minimizira ukupno kodiranje.

Ideja se temelji na tome da brišemo uvjete tako dugo dok možemo smanjiti ukupnu duljinu opisa. Opis svake komponente je najsažetija moguća forma, u slučaju pravila to bi bio broj literala. Ovim postupkom izabiremo optimalnu razinu generalizacije nad skupom za učenje.

Nakon što se stvori pravilo izračuna se njegova duljina (broj bitova) prema skupu za rast i pozitivnim i negativnim primjerima iz skupa za podrezivanje. Kad je duljina za više od d bitova dulja od najmanje do sada dobivene duljine ili kad nema više pozitivnih primjera u skupu za rast, algoritam IREP* zaustavlja stvaranje novih pravila. Duljina pravila pokazuje njegovu složenost. Postavljanjem vrijednosti d definirali smo najveću složenost koju stvoreno pravilo može imati u odnosu na najmanje složeno pravilo u cijelom skupu.

Upotreba metode MDL kao uvjeta zaustavljanja stvaranja pravila u kombinaciji s

novom metodom vrednovanja pravila u postupku podrezivanja pokazalo se iznimno uspješno.

Metode koje su implementirane u algoritmu IREP predstavljale su značajna poboljšanja u odnosu na početni predloženi algoritam REP. Međutim, još su predložena dodatna poboljšanja. Nastala je ideja da pravila stvorena algoritmom IREP* prođu još jednu obradu. Ovaj korak je blizak prvobitnom algoritmu REP jer i tada je postojala naknadna obrada pravila.

Cohenovi eksperimenti doveli su do sljedećeg postupka optimiziranja skupa pravila $R_1 \ldots R_k$:

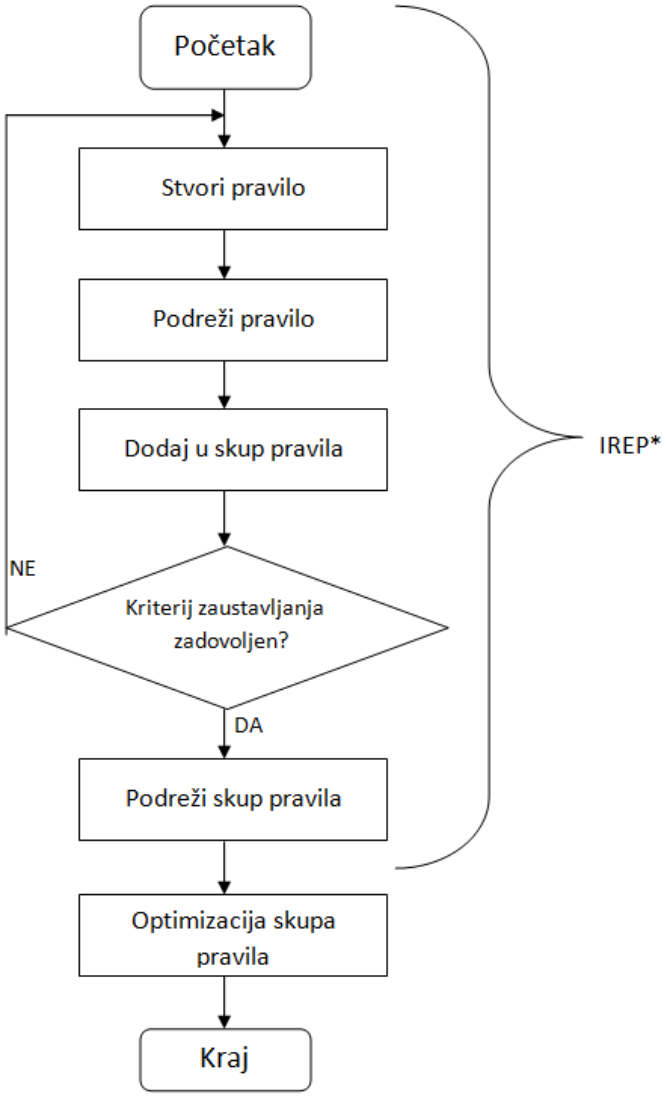
1. Promatraj pravila redom kakvim su stvarana, dakle R_1, R_2, itd.,
2. Za svako pravilo R_i stvori dva alternativna pravila, zamjensko (engl. *replacement*) R_i' i prerađeno (engl. *revision*) R_i'':
   (a) pravilo R_i' stvoreno je rastom iz praznog skupa pravila, a potom podrezano tako da se minimizira učestalost pogreške cijelog skupa pravila $(R_1, \ldots, R_i', \ldots, R_k)$ na skupu za podrezivanje;
   (b) pravilo R_i'' stvoreno je analogno pravilu R_i', osim što se prerada pravila stvara pohlepnim dodavanjem dodatnih uvjeta pravilu R_i, a ne praznom pravilu.
3. Koristeći MDL princip odaberi najbolje pravilo iz skupa R_i, R_i', R_i''.

Svaka varijanta pravila R_i evaluira se tako da se umetne u izgrađeni skup pravila, a potom se brišu pravila koja uvećavaju ukupnu opisnu duljinu pravila i primjera. Ukupna opisna duljina primjera i pojednostavljenog skupa pravila koristi se za usporedbu varijanti pravila R_i.

Cohen je novi algoritam nazvao RIPPER (engl. *Repeated Incremental Pruning to Produce Eror Reduction*). Ako se rezultat algoritma RIPPER ponovno optimizira na isti način, onda govorimo o algoritmu RIPPER2. U općem slučaju, ako ponavljamo optimizacijski postupak k puta, algoritam nazivamo RIPPERk. Slika 3.1 prikazuje cijeli postupak stvaranja pravila za učenje.

Dakle, algoritam RIPPER možemo najsažetije opisati u pet koraka:

1. iskoristi algoritam IREP* za određivanje početnog skupa pravila za klasifikaciju,
2. optimiziraj skup pravila koristeći algoritam za optimizaciju,
3. dodaj pravila da pokriješ možebitne pozitivne primjere koristeći IREP*,



Slika 3.1: Niz koraka koji tvori algoritam RIPPER

4. uzmi rezultate iz trećeg koraka i ponovi postupak optimiziranja ponavljajući korake 2 i 3,

5. zaustavi se nakon k optimizacija.

4. Priprema skupa za učenje

Prije nego krenemo na problem uparivanja ručno imenovanih entiteta u tekstu dokumenta, potrebno je izvršiti obradu teksta i to na način da se izdvoje sva spominjanja entiteta (engl. *mention*). Za svako spominjanje potrebno je definirati neke značajke.

4.1. Označavanje spominjanja entiteta

Razlikujemo tri vrste spominjanja:

- spominjanje imenovanog entiteta (engl. name mention),
- spominjanje nominalnog entiteta (engl. nominal mention),
- spominjanje entiteta korištenjem zamjenice (engl. pronoun mention).

Svaka od navedenih vrsta ima karakteristične značajke koje moramo odrediti prilikom ručnog označavanja.

4.1.1. Imenovano spominjanje entiteta

Kada označimo spominjanje imenovanog entiteta, moramo odrediti njegovu vrstu. Razlikujemo četiri različite vrste imenovanog entiteta:

1. Ime osobe: u ovu skupinu spadaju oni entiteti koji se odnose na imena i prezimena osoba, primjeri takvih entiteta su *Pero Perić*, *predsjednik Josipović*, *premijerka Kosor*, *bivši predsjednik Republike Hrvatske Stjepan Mesić*. Samo spominjanje ne sadrži samo vlastite imenice nego i dodatne riječi koje opisuju entitet i sve to označavamo;

2. Ime organizacije: u ovu skupinu spadaju oni entiteti koji se odnose na organizacije što uključuje korporacije, agencije, udruge i druge grupe ljudi s uspostavljenom organizacijskom strukturom. Primjeri su *Haški sud*, *Vlada Republike Hrvatske*, *UN* i sl.;

3. Imena lokacija: u ovu skupinu spadaju geografski entiteti, mjesta. Primjeri su *Markov trg*, *Libija*, *Dubrovnik*, *Srđ*, *Ulica Ive Vojnovića* i sl.;

4. Geopolitički entiteti: u ovu skupinu spadaju politički definirani entiteti poput nacija, regija, ljudi. Primjeri su *Hrvati*, *Istrani*, *Dalmatinci*, *Balkan* (kad se koristi kao politički pojam, ako se koristi kao zemljopisni pojam odnosno misli se na planinu Balkan ili na poluotok onda se označava kao lokacija).

Osim vrste imenovanog entiteta, potrebno je označiti njegov rod. Rod (muški, ženski, srednji) proizlazi iz oblika same riječi i deklinacijske skupine, a za živa bića se najčešće poklapa sa stvarnim spolom.

Konačno, potrebno je označiti i broj odnosno radi li se o jednini ili množini. Ako se imenica odnosi na jedan objekt onda je u jednini, a ako se odnosi na više njih onda je u množini. Na kraju svaka oznaka spominjanja imenovanog entiteta ima definiranu vrstu imenovanog entiteta, rod i broj.

4.1.2. Nominalno spominjanje entiteta

Ako spominjemo određeni entitet i pritom ne koristimo njegovo ime (ili ga možda i nema) nego ga opisujemo, onda takvo spominjanje nazivamo nominalnim. Primjeri takvih spominjanje su: *prosvjedi u Varšavskoj*, *predsjednik udruge*, *voditelj projekta*, *poznati trgovac nekretninama*. Za takva spominjanja označavamo samo rod i broj. Ta se svojstva određuju prema imenici koja je glava (engl. head) imeničke fraze. Primjeri kako se određuje glava nalaze se u tablici 4.1.

Tablica 4.1: Primjeri glava kod nominalnih spominjanja

Spominjanje	Jezgra	Rod	Broj
prosvjedi u Varšavskoj	prosvjedi	muški	množina
predsjednik udruge	predsjednik	muški	jednina
voditelj projekta	voditelj	muški	jednina
poznati trgovac nekretninama	trgovac	muški	jednina

4.1.3. Spominjanje entiteta korištenjem zamjenica

Posljednja vrsta spominjanja entiteta je korištenjem zamjenica. U obzir smo uzimali dvije vrste: osobne zamjenice i posvojne zamjenice. Za svaku se određivao rod i broj.

4.2. Ostale značajke entiteta

Pored navedenih značajki od koji neke ovise o vrsti spominjanja, koristimo i čitav niz značajki (Uryupina, 2004) koje se automatski izračunaju prilikom označavanja. Te značajke i njihovo značenje navedene su u tablici 4.2.

Tablica 4.2: Značajke koje se izračunavaju prilikom označavanja spominjanja

Značajka	Opis značajke
CHSTART	Redni broj znaka u članku gdje započinje označeno spominjanje entiteta.
CHEND	Redni broj znaka u članku gdje završava označeno spominjanje entiteta.
PARID	Redni broj odlomka u članku gdje se nalazi označeno spominjanje entiteta
SENTID	Redni broj rečenice u članku gdje započinje označeno spominjanje entiteta
WORDNUM	Broj riječi koje sadrži označeno spominjanje
WORDSTART	Redni broj riječi u članku gdje započinje označeno spominjanje entiteta
WORDEND	Redni broj riječi u članku gdje završava označeno spominjanje entiteta
WORDPARSTART	Redni broj riječi u odlomku (gdje se nalazi spominjanje) gdje započinje spominjanje.
WORDPAREND	Redni broj riječi u odlomku (gdje se nalazi spominjanje) gdje završava spominjanje.
WORDSENTSTART	Redni broj riječi u rečenici (gdje se nalazi spominjanje) gdje započinje spominjanje.
WORDSENTEND	Redni broj riječi u rečenici (gdje se nalazi spominjanje) gdje završava spominjanje.

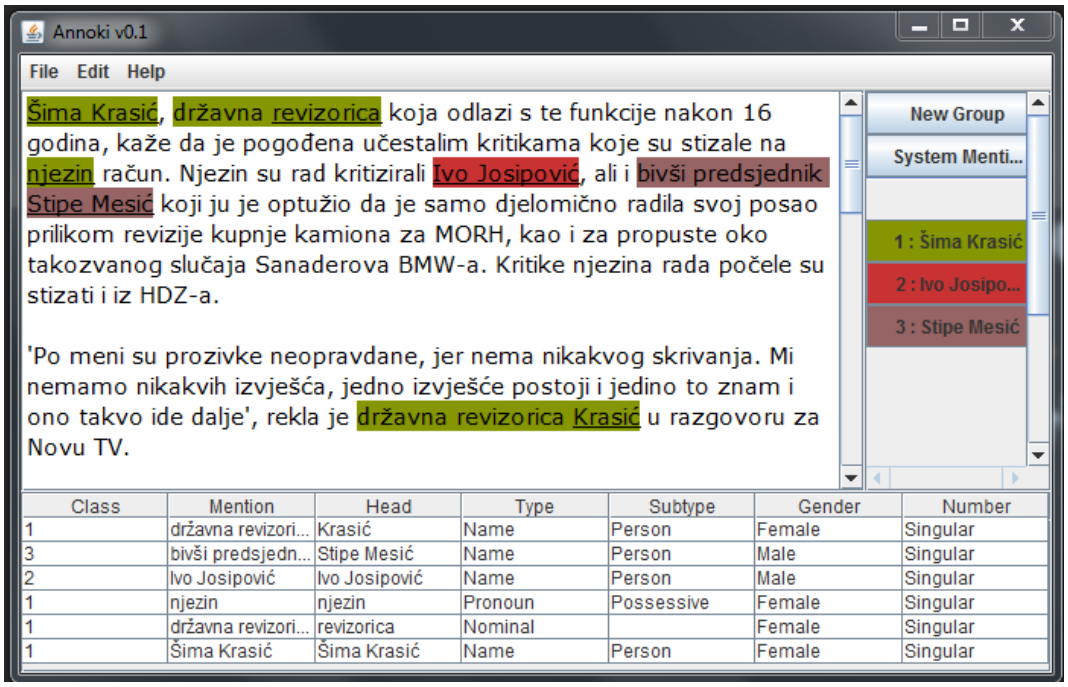
Iz informacija navedenih u tablici 4.2 možemo jednostavno izračunati koliko su spominjanja međusobno udaljena koristeći kao mjeru broj znakova, broj rečenica ili broj odlomka.

4.2.1. Opis alata za označavanje

Trenutno za hrvatski jezik ne postoji korpus s označenim spominjanjima koji bi se koristio za razrješavanje koreferencije. Zbog toga je napravljen alat Anoki (Hranj, 2011) za ručno označavanja spominjanja u dokumentu.

Alat omogućuje označavanje spominjanja koje se potom može staviti u neku od postojećih grupa ili može biti stvorena nova grupa. U istoj grupi nalaze se sva spominjanja koja se odnose na isti izvanjezični entitet. Glavni prozor alata Anoki prikazan je na slici 4.1. Nakon što označivač označi spominjanje mora odrediti dodatna svojstva spominjanja koja su definirana ranije.

Prilikom označavanja spominjanja izračunavaju se vrijednosti značajki navedenih u tablici 4.2.



Slika 4.1: Ručno označavanje u alatu Anoki

Osim već spomenutih vrsta spominjanja u tekstu, označivač može označiti proizvoljne imeničke fraze kao sustavska spominjanja. Takve fraze ne pripadaju niti jednoj grupi, a njihova zadaća je simulacija automatskog označavanja spominjanja odnosno uvođenje pogreške.

5. Korištene značajke u uparivanju koreferentnih imenovanih entiteta

Za izgradnju sustava za uparivanje koreferentnih imenovanih entiteta potrebno je izdvojiti značajke koje nam mogu pomoći. Izdvojene značajke koristi algoritam za izgradnju pravila koji određuju je li određeni par koreferentan ili nije. Značajke su vezane uz određene podzadatke:

- normalizacija,
- biranje najznačajnijeg podniza odnosno riječi,
- uspoređivanje znakovnih nizova (engl. *matching*).

5.1. Normalizacija

Normalizacija je postupak u kojem znakovne nizove istog značenja svodimo na isti oblik. Na primjer, sustav ne može praviti razliku između nizova BIH i BiH, McDonald's i MCDonald's ili F – 16 i f–16. Za postupak normalizacije predložene su dvije metode navedene u tablici 5.1 (Uryupina, 2004).

Tablica 5.1: Postupci normalizacije

Naziv postupka	Opis postupka	Primjeri
No_case	pretvaranje svih znakova u mala slova	BIH $\rightarrow$ bih
no_punctuation	izbacivanje svih interpunkcijskh znakova iz zadanog znakovnog niza	F-16 $\rightarrow$ F16

Ove postupke možemo grupirati slijedno. Primjerice, ako imamo znakovni niz F–16, primjenom oba postupka dobit ćemo niz f16. Moguće je također koristiti ra-

zličite kombinacije normalizacijskih postupaka kod izračunavanja različitih značajki za parove spominjanja entiteta. Normalizacijske postupke potrebno je prilagoditi jeziku na kojem se rješavaju problemi. Tako je u engleskom jeziku jedan od uobičajenih normalizacijski postupaka izbacivanje članova (*a*, *an*, *the*) iz znakovnog niza koji se razmatra. U drugim jezicima mogu se normalizirati pojedini znakovi ($\hat{o} \rightarrow o$).

5.2. Biranje najznačajnijeg podniza

Veliki broj spominjanja imenovanih entiteta čine znakovni nizovi koji se sastoje od više riječi. Nisu sve riječi u znakovnom nizu jednako važne. Neke nose više informacija, neke manje. Zbog toga je predloženo kod uspoređivanja znakovnih nizova, zapravo uspoređivati samo one najreprezentativnije riječi.

Iskorištene su dvije najjednostavnije metode (Uryupina, 2004):

- prvi: uzimanje prve riječi iz znakovnog niza,
- zadnji: uzimanje posljednje riječi iz znakovnog niza.

Oba postupka su veoma brza i još omogućuje puno brže uspoređivanje u sljedećoj fazi postupka, jer zapravo uspoređujemo znakovne nizove sastavljene od samo jedne riječi.

5.3. Postupci za određivanje udaljenosti između znakovnih nizova

Uspoređivanje znakovnih nizova predstavlja najvažniji podzadatak u procesu uparivanja koreferentnih imenovanih entiteta. Predložen je čitav niz postupaka, a navedeni postupci su detaljno opisani u sljedećem odjeljku.

Ovdje će biti opisani algoritmi za izračunavanje udaljenosti između znakovnih nizova koji su korišteni kod rješavanja problema uparivanja koreferentnih imenovanih entiteta. Općenito, postupci za uspoređivanje nizova imaju široku primjenu:

- ispravljanje pravopisnih pogrešaka (engl. spelling correction),
- sustavi za davanje sugestija dok unosite upit,
- sustavi za ispravljanje pogrešaka kod optičkog prepoznavanja znakova (engl. optical character recognition),
- bioinformatika, primjerice, uspoređivanje lanaca DNK ili RNK,

– analiza slike.

Udaljenost među znakovnim nizovima definira se kao broj operacija koji je potrebno napraviti da bi jedan znakovni niz pretvorili u drugi znakovni niz. Razlike među konkretnim algoritmima su zapravo u skupovima dopuštenih operacija. Operacije mogu biti: umetanje znaka, brisanje znaka, zamjena znaka, prebacivanje znaka i sl.

5.3.1. Levenshteinova udaljenost

Levenshteinova udaljenost je mjera koja služi za mjerenje razlike između dva niza (Levenshtein, 1966). Levenshteinova udaljenost između dva znakovna niza definira se kao broj potrebnih promjena da bi se jedan znakovni niz transformirao u drugi znakovni niz. Dozvoljene promjene su:

– umetanje znaka,

– brisanje znaka,

– zamjena znaka.

Iz definicije se vidi da se Levenshteinova udaljenost može računati između dva znakovna niza proizvoljne duljine. Postoje mjere koje zahtijevaju da dva znakovna niza koja se uspoređuju budu iste duljine. Primjer jedne takve mjere je Hammingova udaljenost koja se definira kao broj pozicija u nizovima na kojima se nalaze različiti znakovi. Drugim riječima, Hammingova udaljenost je minimalni broj zamjena znakova potreban da se prvi niz znakova transformira u drugi niz.

Primjeri izračunavanja Levenshteinove udaljenosti navedeni su u tablici 5.4.

Algoritam za računanje Levenshteinove udaljenosti je primjer dinamičkog programiranja (Bellman i Dreyfus, 1962). To je metoda rješavanja složenih problema razlaganjem tih problema na jednostavnije potprobleme. Navedeni problemi moraju imati sljedeća svojstva: optimalna podstruktura:

– optimalno rješenje podproblema može se iskoristiti za pronalaženje optimalnog rješenja cijelog problema,

– preklapajući problemi (engl. *overlapping subproblems*): rješenje jednog podproblema koristi se za rješavanje više većih podproblema.

Algoritam za računanje Levenshteinove udaljenosti predstavlja primjer implementacije dinamičkog rješenja koje koristi pristup odozdo prema gore (engl. bottom-up). U tom pristupu rješavamo i pamtimo rješenja manjih podproblema koja će nam trebati kod računanja većih. Ideja je da koristimo matricu d na način da vrijednost elementa $d[i, j]$ predstavlja Levenshteinovu udaljenost između prvih i znakova prvog niza i j znakova

Tablica 5.2: Primjeri određivanja Levenshteinove udaljenosti

Prvi znakovni niz	Drugi znakovni niz	Levenshteinova udaljenost
Ladena	Laden	Brisanje jednog znaka. Udaljenost je 1.
Knina	Kninu	Zamjena jednog znaka. Udaljenost je 1.
Hrvatska	Hrvatskoj	Zamjena i dodavanje jednog znaka. Udaljenost je 2.
SAD	Bay	Zamjena dva znaka. Udaljenost je 2.
SAD	Laden	Zamjena jednog znaka i dodavanje dva znaka. Udaljenost je 3.
Split	Mesić	Zamjena četiri znaka. Udaljenost je 4.
Dračevcu	Josipović	Zamjena 7 znakova i dodavanje 1 znaka. Udaljenost je 8.

drugog niza. Matricu postupno popunjavamo do kraja. Posljednji donji desni element matrice predstavlja konačno rješenje. Složenost navedenog algoritma je $O(mn)$ gdje je m duljina prvog znakovnog niza, a n duljina drugog znakovnog niza. Levenshteinova udaljenost je nula ako i samo ako su dva znakovna niza identična. Ako su dva znakovna niza identične duljine, gornja granica za Levenshteinovu udaljenost je Hammingova udaljenost. Najveća moguća Levenshteinova udaljenost je duljina duljeg znakovnog niza. Levenshteinova udaljenost je uvijek barem jednaka ili veća od razlike u udaljenosti dvaju znakovnih nizova.

Koristeći Levenshteinovu udaljenost definiramo atribute navedene u tablici 5.3. Levensheteinovu udaljenost možemo koristiti i na razini riječi tako da smatramo dopuštenim operacijama umetanje, brisanje ili izmjenu riječi.

Tablica 5.3: Značajke koje se temelje na Levenshteinovoj udaljenosti

Značajka	Opis
LS	Levenshteinova udaljenost između dva znakovna niza na razini znaka
LS_ante	Značajka LS podijeljena brojem znakova u antecedentu
LS_anafora	Značajka LS podijeljena brojem znakova u anafori
LW	Levenshteinova udaljenost između dva znakovna niza na razini riječi
LW_ante	Značajka LW podijeljena brojem riječi u antecedentu
LW_anafora	Značajka LW podijeljena brojem riječi u anafori

Algoritam 2 daje pseudokod za izračunavanje Levenshteinove udaljenosti.

Algoritam 2 LevenshteinovaUdaljenost(s, t)

Require: s: prvi znakovni niz, t: drugi znakovni niz

```
d[0..m, 0..n] = {}
for i = 0 → m do
    d[i, 0] = i
end for
for j = 0 → m do
    d[0, j] = j
end for
for j = 1 → n do
    for i = 1 → m do
        if s[i] == t[j] then
            d[i, j] = d[i-1, j-1]
        else
            d[i, j] := minimum(d[i-1, j] + 1, d[i, j-1] + 1, znak d[i-1, j-1] + 1)
        end if
    end for
end for
return d[m,n]
```

5.3.2. Najdulji zajednički podniz znakova

Neka imamo znakovni niz s i znakovni niz t. Tražimo duljinu najdužeg znakovnog niza koji je podniz i od niza s i od niza t.

Primjerice, neka je znakovni niz s="Josipović", a znakovni niz t="Kontromanović". Najdulji zajednički podniz znakova je "ović" i njegova duljina je četiri znaka.

Ovaj problem se može riješiti dinamičkim programiranjem. Prvo je potrebno naći najdulje zajedničke sufikse za sve parove prefiksa zadanih znakovnih nizova. Najdulji zajednički sufiks je:

$$NZSUF(s_{1..p}, t_{1..q}) = \begin{cases} NZSUF(s_{1..p-1}, t_{1..q-1}) + 1 & \text{ako je } s[p] = t[q] \\ 0 & \text{inače} \end{cases}$$

Najdulji element u skupu najduljih zajedničkih sufiksa mora biti najdulji zajednički podniz nizova s i t:

Tablica 5.4: Primjer matrice za računanje Levenshteinove udaljenosti između znakovnog niza *Saturday* i znakovnog niza *Sunday*

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

$$NZPN(s,t) = max_{1 \leq i \leq m, 1 \leq j \leq n} NZSUF(s_{1..i}, t_{1..j})$$

Koristeći algoritam za izračunavanje najduljeg zajedničkog podniza znakova definiramo značajke navedene u tablici 5.5. Navedeni algoritam možemo koristiti i na razini riječi.

Tablica 5.5: Značajke koje se temelje na najduljem zajedničkom podnizu znakova

Značajka	Opis
NZPZS	Najdulji zajednički podniz znakova
NZPZS_ante	Značajka NZPZS podijeljena brojem znakova u antecedentu
NZPZS_anafora	Značajka NZPZS podijeljena brojem znakova u anafori
NZPZW	Broj zajedničkih riječi
NZPZW_ante	Značajka NZPZW podijeljena brojem riječi u antecedentu
NZPZW_anafora	Značajka NZPZW podijeljena brojem riječi u anafori

Pseudokod algoritma koji nalazi najdulji zajednički podniz dan je u tablici algoritam 3.

Algoritam 3 NajduljiZajednickiPodniz(s, t)

Require: s: prvi znakovni niz, t: drugi znakovni niz

```
L[1..m, 1..n] = {}
z = 0
ret = {}
for i = 1 → m do
  for j = 1 → n do
    if s[i] == t[j] then
      if i = 1 || j = 1 then
        L[i,j] = 1
      else
        L[i,j] = L[i-1,j-1] + 1
      end if
      if L[i, j] > z then
        z = L[i,j]
        ret = {}
      end if
      if L[i, j] = z then
        ret = ret ⋃ S[i − z + 1..z]
      end if
    end if
  end for
end for
return ret
```

5.4. Ostale značajke korištene u uparivanju koreferentnih imenovanih entiteta

U tablici 4.2 navedene su značajke koje se izračunavaju prilikom označavanja. Te vrijednosti možemo iskoristiti za definiranje značajki koje se koriste u uparivanju koreferentnih imenovanih entiteta navedenih u tablici 5.6.

Tablica 5.6: Ostale značajke korištene u uparivanju koreferentnih imenovanih entiteta

Značajka	Opis
broj	slažu li se spominjanja u broju
broj_znakova	razlika u broju znakova između spominjanja
broj_riječi	razlika u broju riječi između spominjanja
d_rečenice	broj rečenica između dva spominjanja (ako su mentioni u istoj rečenici vrijednost značajke je nula)
d_spominjanje	broj ostalih spominjanja između dva promatrana spominjanja
d_odlomak	broj odlomaka između spominjanja (ako su mentioni u istom odlomku vrijednost značajke je nula)

6. Evaluacija

6.1. Evaluacijske mjere

Uparivanje koreferentnih imenovanih entiteta možemo promatrati kao problem binarne klasifikacije. Promatrani par imenovanih entiteta može pripadati jednoj od dvije moguće klase, klasi pozitivnih primjera odnosno koreferentnih imenovanih entiteta ili klasi negativnih primjera odnosno parova imenovanih entiteta koji nisu koreferentni.

Rezultat rada algoritma za uparivanje koreferentnih imenovanih entiteta nad korpusom dokumenata možemo prikazati pomoću konfuzijske matrice. U tablici 6.1 se nalazi primjer konfuzijske matrice. Varijabla a predstavlja broj puta kada se određeni par imenovanih entiteta označio koreferentnim i ručnim i automatskim putem. Varijabla d predstavlja broj puta kada se određeni par imenovanih entiteta označio nekoreferentnim i ručnim i automatskim putem. Varijable b i c određuju broj puta kada su se parovi različito označili , odnosno kada se rezultat ručnog i automatskog označavanja koreferentnosti nije poklapao.

Tablica 6.1: Konfuzijska matrica

	Ručno klasificirano kao koreferentan par	Ručno klasificirano kao nekoreferentan par
Automatski klasificirano kao koreferentan par	a	b
Automatski klasificirano kao nekoreferentan par	c	d

U procesu evaluacije pristupa kojim rješavamo problem uparivanja koreferentnih entiteta koriste se mjere preciznosti (engl. *precision*) i odziva (engl. *recall*) kako bi se ocijenile performanse sustava. Preciznost je procjena vjerojatnosti, ako je sustav označio par imenovanih entiteta kao koreferentan, da je on zbilja koreferentan:

$$preciznost = \frac{a}{a+b} \tag{6.1}$$

Odziv je procjena vjerojatnosti, ako je par imenovanih entiteta stvarno koreferentan, da će ga sutav takvim i označiti:

$$odziv = \frac{a}{a+c} \tag{6.2}$$

Postoji dobro poznati odnos između preciznosti i odziva. Moguće je povećati jedno, a na štetu drugoga. Pogledajmo trivijalne slučajeve. Ako sustav svaki par označava kao koreferentan, odziv je zapravo 100% jer će svaki par koji jest koreferentan biti označen kao takav. U tom slučaju preciznost će biti veoma niska jer će se i nekoreferentni parovi označavati kao koreferentni.

S druge strane, preciznost od 100% postižemo kada označimo jedan par kao koreferentan, i on stvarno jest koreferentan. Sve ostale parove označimo kao nekoreferentne i tako dobijemo odziv koji je blizu 0% jer čitav niz koreferentnih parova, označavamo kao nekoreferentne parove (osim jednog spomenutog).

Konačno, mjera koja se najčešće koristi za ocjenu performansi algoritama u strojnom učenju naziva se F-mjera (engl. *F-measure*) i temelji se na mjerama preciznosti i odziva. F-mjera se izračunava kao:

$$F_1 = \frac{2 \cdot preciznost \cdot odziv}{preciznost + odziv} = \frac{2a}{2a+b+c} \tag{6.3}$$

Kada su preciznost i odziv otprilike jednaki, F-mjera je skoro jednaka prosjeku preciznosti i odziva. Ako su preciznost i odziv različiti, F-mjera je manja od njihovog prosjeka.

6.1.1. Unakrsna validacija

U svim eksperimentima prilikom evaluacije korištena je unakrsna validacija, točnije k-struka unakrsna validacija. Prednosti unakrsne validacije kao metode za evaluaciju klasifikacijskih modela su u tome da su svi dostupni primjeri iskorišteni za ispitivanje, te se konstrukcija modela u svakoj iteraciji koristi velikom većinom dostupnih primjera. Treba uočiti da su računski zahtjevi primjetno veći u odnosu na klasičnu evaluaciju na ispitnom skupu. Jedan od problema klasične evaluacije u kojoj se klasifikacijski model nauči na skupu za učenje, a evaluira na ispitnom skupu je u tome što se možemo zapravo oslanjati na jedan, moguće nekarakterističan podskup primjera, bilo

da ga koristimo za učenje ili ispitivanje. Unakrsna validacija izbjegava navedeni problem jer osigurava višestruko ponavljanje procesa evaluacije na ispitnom skupu koristeći različite podskupove za učenje i ispitivanje.

U eksperimentima je korištena k-struka unakrsna validacija. Skup primjera se podijelio u k podskupova približno iste veličine. U svakoj od k iteracija mijenjao se podskup za ispitivanje, a svih ostalih $k-1$ podskupova bi se koristilo za izgradnju klasifikacijskog modela. Prilikom podjele inicijalnog skupa primjera u k podskupova postupak slučajnog odabira primjera se izmjeni na način da osigura približno jednaku zastupljenost klasa u svakom od dobivenih podskupova. Navedeni postupak se naziva stratifikacija i njime poboljšavamo reprezentativnost svakog poskupa.

6.2. Opis skupa za učenje

Za potrebe evaluacije koristit će se ručno sakupljeni novinski članci s različitih internetskih portala (npr. Vjesnik, TPortal, Index, Jutarnji list, Večernji list). Za svaki članak ručno su označena spominjanja korištenjem alata za označavanje Anoki. Dobiveni korpus sadrži 100 novinskih članaka. Statistički podaci o dobivenom korpusu prikazani su u tablici 6.2.

Tablica 6.2: Statistički podaci o korpusu dokumenata

Opis	Vrijednost
prosječna duljina članka	3359 znakova
prosječan broj koreferentnih parova	35
prosječan broj nekoreferentnih parova	383

6.3. Određivanje mjere slaganja označivača

Kada se koristi ručno označavanje teksta, potrebno je odrediti mjeru slaganja označivača. Slaganje označivača predstavlja gornju granicu uspješnosti metode razrješavanja koreferencije.

Možemo definirati dvije vrste slaganja označivača. Jedna vrsta slaganja je oko označivanja spominjanja ili proporcija potpunog slaganja (engl. *proportion of true agreement*) opisanog u (Krenn et al., 2004) koji je primjenjiv kada imamo dva označivača

koji donose odluke na binarnim varijablama. Druga vrsta slaganja odnosi se na slaganja označivača o grupiranju spominjanja u entitet, odnosno slažu li se označivači da je određeni par spominjanja koreferentan ili ne.

Navedene mjere izračunate su na temelju 20 članaka koje su označili oba označivača.

6.3.1. Mjera potpunog slaganja

Promatramo tekst članka na razini riječi. Označivač može odabrati za svaku riječ hoće li biti dio nekog spominjanja ili ne. Podaci o slaganju označivača mogu se prikazati konfuzijskom matricom (vidi tablicu 6.3).

Tablica 6.3: Općenita konfuzijska matrica za poklapanje označavanja

	B+	B-
A+	n_{++}	n_{+-}
A-	n_{-+}	n_{--}

Vrijednost n_{++} predstavlja broj riječi u članku koje su oba označivača označili kao dio nekog spominjanja, n_{+-} je broj riječi koje je označivač A označio, a označivač B nije. n_{-+} je broj riječi koje je označivač A nije označio, a označivač B jest i n_{--} je broj riječi koje oba označivača nisu označila kao dio nekog spominjanja.

Proglašavajući jednog od označivača referentnim, možemo odrediti F-mjeru. Matrica zabune za određivanje mjere potpunog slaganja nalazi se u tablici 6.4.

Tablica 6.4: Konfuzijska matrica za poklapanje označavanja

	B+	B-
A+	1240	356
A-	168	6432

Budući da nas zanima potpuno slaganje, gdje oba označivača nezavisno donose iste odluke temeljene na pravilima označavanja koja su prethodno definirana, koristit ćemo izraz:

$$p_v = p_{++} + p_{--} \tag{6.4}$$

, gdje p_v označava vršno slaganje, p_{++} jednako je n_{++}/n, a p_{--} iznosi n_{--}/n. Vršno ili opservacijsko slaganje razlikuje se od potpunog slaganja u tome što uključuje i slaganje nastalo slučajno, pogreškom jednog od označivača. Zbog specifične problematike označavanja spominjanja možemo pretpostaviti da je potpuno slaganje približno jednako vršnom slaganju.

Pripadna mjera vršnog slaganja iznosi $p_v = 0,936$.

6.3.2. Slaganje grupiranja

Problemu određivanja mjere slaganja grupiranja možemo pristupiti na sličan način kao i mjeri potpunog slaganja. Promatramo pripadnost svake riječi koja je označena kod oba označivača. Iz skupa riječi koje su oba označivača označila promatramo sva moguća rukovanja među riječima i promatramo pripadnost grupama. Ovdje se također radi o binarnoj varijabli, tj. svaka riječ para može biti svrstana u istu grupu (označavamo znakom '=') ili svaka riječ para može biti svrstana u neku drugu grupu (označavamo znakom '$\neq$'). Sustavska spominjanja nisu uzimana u obzir jer mogu biti proizvoljno odabrana od strane označivača. Matrica zabune za slaganje grupiranja prikazana je u tablici 6.5. F-mjera slaganja grupiranja iznosi $F_1 = 0,883$.

Tablica 6.5: Konfuzijska matrica za slaganje grupiranja

	$B_=$	$B_\neq$
$A_=$	5236	1250
$A_\neq$	137	39653

6.4. Eksperimenti s različitim pristupima nadziranog strojnog učenja

6.4.1. Stabla odlučivanja

Stablo odlučivanja može se promatrati kao klasifikacijski algoritam izražen u formi povezanog grafa sa strukturom stabla. Unutarnji čvorovi u stablu odlučivanja označeni su nazivima atributa i predstavljaju test na vrijednost atributa, a grane koje izlaze označene su mogućim vrijednostima atributa. Listovi stabla odlučivanja označeni su nazivima klasa u koje se primjeri razvrstavaju. Stabla odluke robusna su na šum i

mogu učiti disjunktivne koncepte (Mitchell, 1997). Klasifikacija primjera obavlja se odozgo, od korijena prema listovima.

Temeljni algoritam za izgradnju stabla odluke je algoritam ID3 (Quinlan, 1986). Algoritam C4.5 (Quinlan, 1993) omogućio je korištenje numeričkih atributa pri izgradnji stabala odluke, određivanje težine za dane atribute, podršku za nedostajuće vrijednosti. Konačno, algoritam C4.5 definirao je postupak podrezivanja stabala. U tom postupku korišten je princip minimalne duljine opisa, koji je kasnije korišten i u algoritmu RIPPER za podrezivanje pravila.

6.4.2. Naivan Bayesov klasifikator

Naivan Bayesov klasifikator jedna je od praktično primjenjivih metoda Bayesovog učenja. Ako su D podaci, a h hipoteza (model), onda Bayesovo pravilo definiramo na sljedeći način:

$$p(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{6.5}$$

Bayesov naivan klasifikator temelji se na pretpostavci da su atributi unutar podataka međusobno nezavisni (zato ga nazivao naivnim) i time se značajno pojednostavljuje problem (Mitchell, 1997). Navedena pretpostavka u stvarnosti može biti narušena, ali u praksi pokazuje često zadovoljavajuće rezultate. Ne pretražuje se eksplicitno čitav prostor hipoteza. Umjesto toga, ispravna hipoteza se gradi određivanjem frekvencije pojavljivanja klasa i vrijednosti atributa.

Klasifikator računa MAP-hipotezu (engl. *Maximum A Posteriori*). To je ona hipoteza za koju je $P(h|D)$ najveći za predočene podatke D. Na temelju Bayesovog pravila dobivamo:

$$h_{MAP} = \arg\max_{h_i \in H} P(h_i|D) h_{MAP} = \arg\max_{h_i \in H} frac P(D|h_i) P(h_i) P(D) \tag{6.6}$$

Vrijednost $P(D)$ možemo izostaviti jer je konstanta:

$$h_{MAP} = \arg\max_{h_i \in H} P(D|h_i) P(h_i) \tag{6.7}$$

6.4.3. Stroj s potpornim vektorima

Stroj s potpornim vektorima (engl. *Support Vector Machine*) je jedan od algoritama nadziranog strojnog učenja. Ovom metodom primjeri se predstavljaju kao točke u

višedimenzionalnom prostoru. Cilj je pronaći takve granice između primjera različite klase da budu maksimalno udaljene od primjera. Takav pristup smanjuje pogrešku prilikom generalizacije. Iako je inicijalno zamišljen kao linearni klasifikator, primjenom jezgrenih funkcija može se koristiti i kao nelinearni klasifikator. Često korištena jezgrena funkcija je radijalna bazna funkcija odnosno Gaussova jezgra:

$$k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{6.8}$$

Učenjem SVM-a traže se parametri pretpostavljenog oblika decizijske funkcije, koja će ispravno klasificirati sve uzorke u skupu za učenje. Zbog šuma u podacima do ponekad nije moguće. Rješenje tog problema nalazi se u korištenju klasifikatora s mekim granicama definiranima parametrom koje dozvoljavaju odstupanja od ispravne klasifikacije svih podataka u skupu za učenje s ciljem bolje generalizacije nad još neviđenim podacima.

6.4.4. Rezultati

U programskom sustavu Weka (Bouckaert et al., 2010) proveden je eksperiment u kojem se ispitalo koliko uspješno prethodno navedeni algoritmi rješavaju problem uparivanja koreferentnih imenovanih entiteta. Eksperiment se sastojao od deseterostruke unakrsne validacije (engl. *10 fold cross validation*). Rezultati su predstavljeni u tablici 6.6.

Tablica 6.6: F-mjera navedenih algoritama nadziranog strojnog učenja dobivena desetorostrukom unakrsnom validacijom u programskom sustavu Weka

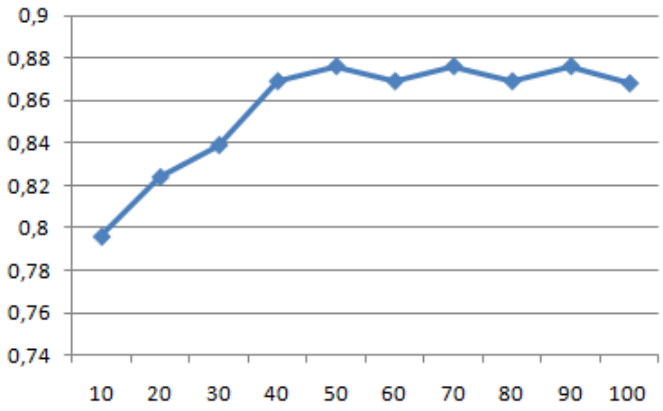
Algoritam	F-mjera
Naivan Bayesov klasifikator	0,857
Stabla odluke (C4.5)	0,871
SVM	0,856

Stabla odluke su se pokazala najuspješnijim pristupom.

6.5. Ovisnost o veličini skupa za učenje

Eksperiment je napravljen korištenjem algoritma RIPPER. Cilj je bio utvrditi ovisnost F_1-mjere i veličine skupa za učenje. Napravljeno je 10 skupova članaka, takvih da

je najmanji imao 10 članaka, a najveći je sadržavao svih 100 članaka. Razlika među skupovima je bila 10 članaka. Na slici 6.1 prikazana je ovisnost F_1-mjere i veličine skupa za učenje. U svakom eksperimentu napravljena je 10-struka unakrsna validacija.



Slika 6.1: Ovisnost F_1-mjere i veličine skupa za učenje

Rezultati pokazuju da je dovoljno 50 članaka u skupu za učenje da postignemo najbolju F_1-mjeru. Uspješnost sustava se nakon toga više ne poboljšava.

6.6. Ispitivanje raznih konfiguracija značajki

Eksperimenti su provedeni nad skupom od 50 članaka. Ispitujemo koliki utjecaj na uspješnost imaju određeni atributi. Definirane su neki skupovi značajki i prikazani u tablici 6.7.

Prvo se promatralo samo potpuno poklapanje promatranih spominjanja. Ako su se promatrana spominjanja potpuno poklapala smatrali smo ih koreferentnim. F_1-mjera je iznosila $0,52$ ako smo uspoređivali prve riječi spominjanja, a $0,63$ ako smo uspoređivali posljednju riječ u spominjanjima.

Prvi eksperimenti su napravljeni uzimajući u obzir posljednju riječ u spominjanjima. Rezultati su navedeni u tablici 6.8.

Najlošiji rezultat su postigle značajke iz skupa OSTALO, gdje je ključna značajka koja pokazuje da li se potpuno preklapaju dva spominjanja (točnije dvije posljednje riječi u spominjanju). Ostale značajke iz navedenog skupa imale su neznatan utjecaj na rezultat.

Korištenjem značajki koje uzimaju u obzir udaljenost između dva znakovna niza ili najdulji zajednički podniz, rezultati postaju značajno bolji. Također je vidljivo da

Tablica 6.7: Određeni imenovani podskupovi značajki

Ime podskupa	Značajke
SVE	LS, LS_ante, LS_anafora, LW, LW_ante, LW_anafora, NZPZS, NZPZS_ante, NZPZS_anafora, NZPZW, NZPZW_ante, NZPZW_anafora, poklapanje, broj, broj_znakova, broj_riječi, d_rečenice, d_spominjanje, d_odlomak
LS_SVE	LS, LS_ante, LS_anafora
LW_SVE	LW, LW_ante, LW_anafora
NZPZS_SVE	NZPZS, NZPZS_ante, NZPZS_anafora
NZPZW_SVE	NZPZW, NZPZW_ante, NZPZW_anafora
OSTALO	poklapanje, broj, broj_znakova, broj_riječi, d_rečenice, d_spominjanje, d_odlomak

Tablica 6.8: Rezultati ispitivanja različitih podskupova značajki

Podskup	Rezultat
SVE	0,876
LS_SVE+NZPZS_SVE	0,87
LS_SVE	0,871
NZPZS_SVE	0,869
LS	0,83
NZPZS	0.778
OSTALO	0,63

je nužno takve značajke normalizirati odnosno podijeliti s brojem znakova jednog od spominjanja.

Drugi eksperiment je proveden uzimajući u obzir sve riječi u spominjanju. Najveća postignuta F_1-mjera iznosi 0,875 i slična je kao i u prethodnom eksperimentu. U ovom slučaju još veći značaj ima normaliziranje udaljenosti među znakovnim nizovima.

Što se tiče metoda normalizacije (korištenje malih slova i uklanjanja znakova interpunkcije), pokazalo se da nekorištenje normalizacije neznatno degradira (F_1-mjera padne za 1%).

Tablica 6.9: Rezultati ispitivanja različitih podskupova značajki uzimajući u obzir sve riječi u promatranim spominjanjima

Podskup	Rezultat
SVE	0.875
LS_SVE	0.847
LS	0.694
OSTALO	0.48

6.6.1. Eksperiment sa svim vrstama spominjanja

Proveden je eksperiment u kojem su se u obzir uzimala sve vrste spominjanja. Korištene su sve značajke. Prvo su se uzimale u obzir sve riječi u spominjanju i postignuta je F_1-mjera 0,54. Potom se koristila samo posljednja riječ u spominjanjima i postignuta je F_1-mjera 0,535. Rezultati su očekivani, potrebno je posebnu pažnju posvetiti nominalnim spominjanjima koji su dosta zahtjevni za razrješavanje. Kod razrješavanja zamjenica značajke iz skupa OSTALO su korisne jer često se zamjenice odnose na spominjanja koja su u istoj ili susjednoj rečenici.

7. Zaključak

U ovom radu predstavljen je problem uparivanja koreferentnih imenovanih entiteta u dokumentima na hrvatskom jeziku. To je problem koji spada u područje razrješavanja koreferencije. To je postupak kojim se utvrđuje koji se izrazi u tekstu dokumenta odnose na isti izvanjezični entitet s obzirom na to da se u tekstovima na iste izvanjezične entitete najčešće ne referiramo istim nazivima već zamjenicama, dijelovima naziva, drugim imenskim frazama i sl. Ovaj rad se koncentrira na imenovane entitete i cilj mu je pronaći postupke koji bi poboljšali razrješavanje koreferencije imenovanih entiteta.

Za rješavanje ovog problema implementirana je metoda nadziranog strojnog učenja temeljena na pravilima. Implementiran je algoritam RIPPER koji se pokazao uspješnim u ovom području na drugim jezicima. Ručno su označena spominjanja u 100 novinski članaka i to je korišteno kao skup za učenje. Korišteni su postupci koji omogućuju uspoređivanje znakovnih nizova poput Levenshteinove udaljenosti i određivanja najduljeg zajedničkog podniza dvaju nizova. Najbolja postignuta F_1-mjera iznosila je 0,876 što predstavlja dobar rezultat.

Budući rad bi uključivao posvećivanje i nominalnim spominjanjima koji su najteži problem kod razrješavanja koreferencije. Također, treba iskoristiti sve jezične alate koji će biti razvijeni za hrvatski jezik da bi se poboljšala uspješnost razrješavanja koreferencije.

LITERATURA

Richard Bellman i Stuart E Dreyfus. *Applied dynamic programming, by Richard E. Bellman and Stuart E. Dreyfus*. Princeton University Press, Princeton, N.J.,, 1962.

Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, i Ian H. Witten. WEKA–experiences with a java open-source project. *Journal of Machine Learning Research*, 11:2533–2541, 2010.

Karl Branting. Name-matching algorithms for legal case-management systems. U *The Journal of Information, Law and Technology*, 2002.

Jonathan H. Clark i José P. González-brenes. Coreference resolution: Current trends and future directions, 2008.

William W. Cohen. Fast effective rule induction. U *International Conference on Machine Learning*, stranice 115–123, 1995.

Johannes Fürnkranz i Gerhard Widmer. Incremental reduced error pruning, 1994.

Zoran Hranj. Razrješavanje koreferencije metodom nenadziranog strojnog učenja, 2011.

Brigitte Krenn, Stefan Evert, i Heike Zinsmeister. Determining intercoder agreement for a collocation identification task. U *In: Proceedings of KONVENS 2004*, 2004.

VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

A. McEnery, I. Tanaka, i S. Botley. Corpus annotation and reference resolution. U *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, ANARESOLUTION '97, stranice 67–74, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. URL `http://portal.acm.org/citation.cfm?id=1598819.1598829`.

Tom Michael Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. WCB/McGraw-Hill, Boston, MA, 1997.

Ruslan Mitkov i Wolverhampton Wv Sb. Anaphora resolution: The state of the art. Technical report, School of Languages and European Studies, 1999.

J. R. Quinlan. Simplifying decision trees. *Int. J. Man-Mach. Stud.*, 27:221–234, September 1987. ISSN 0020-7373. doi: 10.1016/S0020-7373(87)80053-6. URL `http://portal.acm.org/citation.cfm?id=50007.50008`.

J. R. Quinlan. Mdl and categorical theories (continued). U *In Machine Learning: Proceedings of the Twelfth International Conference, Lake Taho*, stranice 464–470. Morgan Kaufmann, 1995.

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022643204877.

Wee Meng Soon, Hwee Tou Ng, i Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27:521–544, December 2001. ISSN 0891-2017. URL `http://portal.acm.org/citation.cfm?id=972597.972602`.

Michael Strube, Stefan Rapp, i Christoph Müller. The influence of minimum edit distance on reference resolution. U *In proceedings of Empirical methods in Natural Language Processing Conference*, stranice 312–319, 2002.

C. Sang Suh. *Practical Applications of Data Mining*. Jones and Bartlett Learning, 2011.

Olga Uryupina. Evaluating name-matching for coreference resolution. U *In Proceedings of LREC'04*, 2004.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, i Lynette Hirschman. A model-theoretic coreference scoring scheme. U *Proceedings of the 6th conference on Message understanding*, MUC6 '95, stranice 45–52, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. ISBN 1-55860-402-2. doi: http://dx.doi.org/10.3115/1072399.1072405. URL `http://dx.doi.org/10.3115/1072399.1072405`.

Robert A. Wagner i Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21:168–173, January 1974. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/321796.321811. URL http://doi.acm.org/10.1145/321796.321811.

Uparivanje koreferentnih imenovanih entiteta metodama strojnog učenja

Sažetak

Razrješavanje koreferencije postupak je kojim se utvrđuje koji se izrazi u tekstu dokumenta odnose na isti izvanjezični entitet. Koreferentni izrazi mogu biti vlastita imena, imeničke fraze ili zamjenice. Razrješavanje koreferencije važan je zadatak u okviru obrade prirodnog jezika te nužan preduvjet za mnoge zadatke ekstrakcije informacija.

U okviru ovog diplomskog rada proučeni su postupci za uparivanje koreferentnih imenovanih entiteta. Razrađen je postupak za uparivanje koreferentnih imenovanih entiteta u tekstovima na hrvatskom jeziku temeljen na nadziranom strojnom učenju. Razvijena je programska implementacija postupka i vrednovana na odgovarajućem ispitnom uzorku.

Ključne riječi: razrješavanje koreferencije, uparivanje koreferentnih imenovanih entiteta, učenje temeljeno na pravilima.

Matching Coreferent Named Entitites Using Machine Learning

Abstract

Coreference resolution is the process in which we identify the mentions that are referring to a same real-world entity. Coreferent mentions can be either named, nominal or pronominal. Coreference resolution is a important task in the field of natural language processing and necessary prerequisite for many information extraction tasks.

In this diploma thesis we study techniques for matching coreferent named entities. Technique is developed for matching coreferent named entities in texts on Croatian language based on supervised machine learning. Programming implementation of procedure is made and evaluated on proper testing corpus.

Keywords: coreference resolution, matching coreferent named entities, rule based learning.