SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1680

PRIMJENA EVOLUCIJSKOG PROGRAMIRANJA NA NALAŽENJE OPTIMALNIH MJERA ZA EKSTRAKCIJU KOLOKACIJA IZ TEKSTA

Ivan Sikirić

Zagreb, rujan 2007.

Zahvaljujem mentorici prof. dr. sc. Dalbelo Bašić na podršci i stručnom vodstvu, te mr. sc. Janu Šnajderu i kolegi Saši Petroviću na brojnim savjetima kojima su popratili nastajanje ovog diplomskog rada

Sadržaj

Popis tablica

Popis slika

1. Uvod

Porastom količine dokumenata dostupnih u digitalnom obliku (posebno na Internetu) raste potreba za sustavima za organizaciju dokumenata pisanih prirodnim jezikom. Takvi dokumenti su općenito nestrukturirani, te su uz tehnike strojnog učenja potrebni složeni statistički i heuristički algoritmi da bi se olakšala njihova strojna obrada. Ekstrakcija kolokacija jedan je od takvih algoritama koji je od velike važnosti u sustavima za automatsko sadržajno označavanje dokumenata (indeksiranje), te u sustavima za dubinsku analizu teksta (eng. *text mining*). Postoji niz različitih mjera za ekstrakciju kolokacija iz teksta. U okviru ovog rada evolucijskim tehnikama će se pretražiti velik dio prostora svih mogućih mjera, u svrhu pronalaska mjere koja daje najbolje rezultate na standardnim zbirkama dokumenata na hrvatskom jeziku. Također će se ispitati kvaliteta i opravdanost najčešće korištenih mjera za ekstrakciju kolokacija, s posebnim naglaskom na rad Petrović et al. [25], koji se ubraja u važnije radove na području ekstrakcije kolokacija hrvatskog jezika.

U drugom poglavlju opisuju se bitni pojmovi iz područja dubinske analize teksta, te se navode najčešće korištene mjere za ekstrakciju kolokacija i metoda kojom se uspoređuju. U trećem poglavlju objašnjavaju se bitni pojmovi iz evolucijskog računanja, te princip rada i vrste evolucijskog algoritma. U četvrtom poglavlju objašnjava se način na koji su kombinirane paradigme iz oba područja u algoritam koji traži najbolje mjere za ekstrakciju kolokacija. U petom poglavlju dani su dobiveni rezultati i njihova analiza.

2. Kolokacija

2.1. *n*-gram riječi

Uz pojam kolokacije veže se pojam *n*-grama riječi, koji označava bilo koji niz od *n* riječi. Naziv za *n*-gram od dvije riječi je digram, dok se *n*-grami od tri riječi zovu trigramima, a *n*-grami od četiri riječi tetragramima. U kontekstu dubinske obrade teksta postoji i pojam *n*-grama slova. U daljnjem tekstu podrazumijeva se da termin *n*-gram označava *n*-gram riječi, a ne *n*-gram slova.

2.2. Definicija kolokacije

Mnogo je definicija kolokacije, neki izvori poistovjećuju je s idiomom, dok drugi smatraju kolokacijama bilo koji niz riječi (*n*-gram) koje se u tekstu zajedno pojavljuju više puta nego što je statistički vjerojatno.

Većina kolokacija zadovoljava sljedeća tri uvjeta [21]:

- značenje cijele kolokacije je veće od sume značenja riječi koje ju tvore
- nije moguće zamijeniti riječ kolokacije sa drugom riječi i očuvati značenje
- nije moguće slobodno modificirati kolokaciju dodatnim leksičkim materijalom ili vršiti nad njom gramatičke transformacije

Treći kriterij je naizraženiji kod idioma.

Jedan od ciljeva ovog rada je pronaći skup mjera koje će se koristiti u sustavu za automatsko indeksiranje dokumenata CADIS [12], stoga se prihvaća definicija kolokacije prikladna za indeksiranje, kako je opisano u Petrović et al. [25]. Četiri tipa kolokacija se prepoznaju:

- prvi tip kolokacije podudara se s definicijom imeničke složenice (*open compound*) [26]. To je neprekinut niz riječi koje funkcioniraju kao cjelina u rečenici. Primjeri: sanitarna inspekcija, uporabna dozvola

- drugi tip kolokacija su vlastite imenice i vlastita imena (*proper nouns, proper names*). Primjer: Pero Perić
- treći tip kolokacija su terminološki izrazi (*terminological expressions*). Ovo se uglavnom odnosi na koncepte i objekte iz tehničkih domena, npr. logička vrata
- četvrti tip kolokacija obuhvaća nizove riječi koji se često pojavljuju zajedno spojeni prijedlozima ili veznicima i opisuju slične koncepte (sport i rekreacija, lov i ribolov)

Prva tri tipa kolokacija nose korisne informacije o sadržaju dokumenta, dok četvrti tip pomaže stručnjacima pri neautomatiziranom indeksiranju. Treba imati na umu da su ove kolokacije kratke i neprekinute, za razliku od dugih kolokacija opisanih u [33].

2.3. Kolekcija dokumenata

Automatizirana ekstrakcija kolokacija temelji se na obradi statističkih i lingvističkih podataka o velikim kolekcijama dokumenata (broj dokumenata mjeri se u tisućama). Statistički podaci tipično uključuju broj pojavljivanja svake pojedine riječi i *n*-grama u čitavoj kolekciji. Prije pribavljanja svih potrebnih statistika, potrebno je pretprocesirati kolekciju. Svi dokumenti u kolekciji mogu se prikazati kao niz riječi i interpunkcijskih simbola. Interpunkcijski simboli se koriste kako bi se izdvojile pojedine rečenice teksta, a nakon toga se ignoriraju. U hrvatskom jeziku velik broj riječi podložan je *fleksiji*, tj. promjeni oblika. Primjerice, imenice se deklíniraju, glagoli konjugiraju. Postupak svođenja različitih oblika jedne riječi u osnovni oblik zove se *lematizacija*, a dobiveni osnovni oblik *lema*. Ovaj postupak razlikuje se od vađenja korijena riječi (osnovni oblici riječi neće nužno biti jednaki korijenima tih riječi), a obavlja se korištenjem specijaliziranih rječnika. Neke riječi se ne nalaze u tim rječnicima, pa ih nije moguće lematizirati. One ostaju u izvornom obliku. Postupak lematizacije osim transformacije riječi u leme također određuje i skup mogućih *vrsta riječi* kojima neka riječ može pripadati.

Vrste riječi dijele se u pet kategorija:

- imenice (oznaka: N)
- pridjevi (oznaka: A)
- glagoli (oznaka: V)
- stop riječi (oznaka: X) – sve ostale vrste riječi
- nepoznate (oznaka: F) – za riječi koje nisu u rječniku

Potrebno je napomenuti da termin *stop riječi* označuje riječi koje se pojavljuju u prevelikom broju (tipično veznici i prijedlozi), pa se ignoriraju pri nekim postupcima dubinske analize teksta. Sustav koji je pretprocesirao kolekcije dokumenata korištene u ovom radu bio je implementiran u skladu s gornjom podjelom, osim što mnoge priloge tumači kao pridjeve (jer se većina priloga hrvatskog jezika može protumačiti kao pridjev u srednjem rodu jednine). Svakoj riječi pridružuje se skup oznaka jer jednoznačno određivanje vrste riječi ovisi o kontekstu u kojem se one nalaze, a još uvijek ne postoji sustav koji bi kontekst uzeo u obzir. Primjer: riječ *kraj* može biti prijedlog i imenica, pa joj se pridružuje skup oznaka { N, X }.

Nakon takvog pretprocesiranja moguće je ekstrahirati sve različite riječi, digrame, trigrame, općenito *n*-grame bilo koje duljine u lematiziranom obliku. Pri tome se pazi da *n*-grami ne prelaze granice rečenica. Za svaku riječ i *n*-gram poznat je ukupan broj pojavljivanja u kolekciji dokumenata, a za svaku riječ dodatno je poznat skup oznaka kategorija vrsta riječi.

2.4. Brojanje i filtriranje po vrsti riječi

Postupci koji olakšavaju ekstrakciju kolokacija su:

- filtriranje *n*-grama po frekvenciji
- filtriranja *n*-grama po vrsti riječi

2.4.1. Filtriranje *n*-grama po frekvenciji

Neki *n*-grami se u kolekciji dokumenata pojavljuju samo nekoliko puta, pa se smatra da se na osnovu tih nekoliko pojavljivanja o tim *n*-gramima ne može ništa

zaključiti. U radu Petrović et al. [25] svi digrami i trigrami koji se pojavljuju manje od tri puta ne uzimaju se kao kandidati za kolokacije. Ponekad je cilj ekstrakcije pronaći vrlo rijetke kolokacije, te se tada frekvencijski filtar ne koristi.

2.4.2. Filtriranje *n*-grama po vrsti riječi

U različitim jezicima postoje različita ograničenja na vrste riječi pojedinih sastavnica kolokacije. Na primjer, kolokacije od dvije riječi u hrvatskom jeziku u pravilu su oblika (*pridjev, imenica*) ili (*imenica,imenica*), dok kolokacije od tri riječi poprimaju oblike (*pridjev, imenica, imenica*), (*pridjev, pridjev, imenica*), (*imenica, pridjev, imenica*), (*imenica, imenica, imenica*), (*imenica, veznik, imenica*) te (*imenica, prijedlog, imenica*). Kao mogući kandidati za kolokacije mogu se uzeti samo oni *n*-grami koji su odgovarajućeg oblika.

2.5. Ekstrakcija kolokacija

Mjere za ekstrakciju kolokacija tipično na temelju dostupnih statističkih podataka o *n*-gramima u kolekciji dokumenata svakom *n*-gramu pridružuju realan broj. Time se dobija rang lista *n*-grama, a najbolji *n*-grami liste proglašavaju se kolokacijama. Postavlja se pitanje koliki dio liste proglasiti kolokacijama. Ako se odabere fiksan broj, ne uzima se u obzir veličina kolekcije dokumenata. Ako neka druga kolekcija sadrži više dokumenata ili dokumente sa većim brojem riječi, vjerojatno će sadržavati više kolokacija. Bolja opcija je određen udio liste proglasiti kolokacijama. No, relativan udio kolokacija u kolekciji dokumenata nije konstantan, pa i ova metoda može donijeti loše rezultate. Najbolja je opcija shvatiti mjeru za ekstrakciju kolokacija kao neizraziti skup - *n*-grami s pridruženim velikim brojevima su vrlo vjerojatno kolokacije, dok *n*-grami s pridruženim malim brojevima vrlo vjerojatno nisu kolokacije. Broj pridružen *n*-gramu određuje *u kolikoj* mjeri se taj *n*-gram može smatrati kolokacijom. Naravno, ne pridružuju sve mjere *n*-gramima brojeve iz intervala [0, 1], kao što je uobičajeno za neizrazite skupove, te se utoliko razlikuju od njih, no bitan zaključak je – za svaku mjeru potrebno je odabrati neki *prag* $p \in \Re$. Oni *n*-grami kojima je pridružen broj veći od praga smatraju se kolokacijama, dok se ostali ne smatraju kolokacijama. Time se varijacije u veličini i sastavu kolekcije dokumenata uzimaju u obzir. Ovaj postupak korišten je u [25].

Formalno, ako je S_K skup dostupnih statističkih i lingvističkih podataka o kolekciji dokumenata K, a S_N skup poznatih statističkih i lingvističkih podataka o nekom *n*-gramu *N* iz te kolekcije, te ako je *S* skup svih uređenih parova (S_N, S_K), za sve moguće kolekcije dokumenata, tada je mjera za ekstrakciju kolokacija bilo koja funkcija $g : S \to \Re$. Odluku o proglašavanju nekog *n*-grama iz neke kolekcije dokumenata kolokacijom donosi binarna funkcija $f : S \to \{DA, NE\}$, definirana sljedećim izrazom:

$$f(S_N; S_K) = \begin{cases} DA, & g(S_N; S_K) \geq p \\ NE, & g(S_N; S_K) < p \end{cases}$$

Ako i samo ako je vrijednost funkcije *f* za neki *n*-gram u nekoj kolekciji dokumenata „DA“, taj *n*-gram se smatra kolokacijom. Parametar *p* je gore spomenuti prag, koji se određuje (eksperimentalno) za svaku mjeru *g* posebno. Važno je uočiti da neće nužno isti *n*-gram u različitim kolekcijama biti jednako klasificiran.

2.6. Standardne mjere za ekstrakciju kolokacija

2.6.1. Notacija

Varijable *a, b* i *c* označavat će proizvoljne riječi. Digrami će se umjesto ispravnog zapisa (*a*, *b*) označavati kraće kao *ab*. Funkcija *f* u daljnjem tekstu je *frekvencija* (broj pojavljivanja) neke riječi ili *n*-grama, *P* je funkcija *vjerojatnosti pojavljivanja* neke riječi ili *n*-grama u kolekciji dokumenata, a *vr* je funkcija koja nekoj riječi pridružuje skup oznaka kategorija vrsta riječi (neki element skupa $\wp(\{N, A, V, X, F\})$). Tako je *f*(*ab*) frekvencija digrama (*a*, *b*), *P*(*abc*) vjerojatnost pojavljivanja trigrama (*a*, *b*, *c*), a *vr*(*a*) skup kategorija vrsta riječi koji je pridružen riječi *a*.

2.6.2. Frekvencijska mjera

Ovo je najjednostavnija od standardnih mjera za ekstrakciju kolokacija. Svakom *n*-gramu pridružuje se vrijednost jednaka njegovoj frekvenciji. Rezultati koje postiže ova mjera su jako loši [25], čak i ako se uz ovu mjeru koristi filtriranje *n*-grama po vrsti riječi (čime se eliminiraju jako česti *n*-grami kao što su „čak i“, „uz ovo“ i slični, koji očito nisu kolokacije).

2.6.3. PMI (pointwise mutual information) mjera

Ova mjera dolazi iz područja teorije informacija i mjeri količinu informacija koju imamo o pojavljivanju jedne riječi, ako imamo informacije o pojavljivanju druge riječi [5]:

$$I(a,b) = \log_2 \frac{P(ab)}{P(a)P(b)}$$

Kako gornja mjera favorizira rijetke događaje, ponekad se umjesto nje koristi sljedeća [21]:

$$I'(a,b) = \log_2 \frac{P(ab)f(ab)}{P(a)P(b)}$$

2.6.4. DICE koeficijent

Ova mjera često se koristi za mjerenje sličnosti u tehnikama pretraživanja informacija [21].

$$DICE(a,b) = \frac{2f(ab)}{f(a)+f(b)}$$

2.6.5. Mjere temeljene na testiranju hipoteza

Dvije mjere iz područja statistike se koriste za ekstrakciju kolokacija. Obje se temelje na testiranju hipoteza, tj. sa prihvaćanjem ili odbijanjem hipoteze „riječi *a* i *b* se slučajno pojavljuju zajedno“.

Radi se o mjerama hi-kvadrata [21]:

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

i log-likelihood [24]:

$$G^2 = \sum_{i,j} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

E_{ij} su *očekivane (expected),* dok su O_{ij} *stvarne (observed)* vrijednosti elemenata kontigencijske tablice. Primjer kontigencijske tablice dan je za digrame, a za veće *n*-grame tablica je analogna:

Tablica 1: kontigencijska tablica za digrame

	prva riječ	¬prva riječ	ukupno
druga riječ	n_{11}	n_{12}	n_{1p}
¬druga riječ	n_{21}	n_{22}	n_{21}
ukupno	n_{p1}	n_{p2}	n_{pp}

Vrijednost n_{11} je broj pojavljivanja digrama u kolekciji dokumenata, n_{21} je broj pojavljivanja prve riječi nakon koje ne slijedi druga riječ, n_{p1} je ukupan broj pojavljivanja prve riječi, n_{22} je ukupan broj digrama kojima prva riječ nije na prvom mjestu, te druga riječ nije na drugom mjestu. Na kraju, n_{pp} je ukupan broj digrama u kolekciji dokumenata, a značenje ostalih elemenata tablice je analogno. U ovom radu koriste se sljedeće aproksimacije (navedeni su izrazi za digrame, za ostale *n*-grame vrijede analogni izrazi). Broj riječi neke kolekcije dokumenata označen je sa *N*:

$$O_{11} = f(ab) \qquad O_{12} = f(a) - f(ab)$$

$$O_{21} = f(b) - f(ab) \qquad O_{22} = N - f(a) - f(b) - f(ab)$$

$$E_{11} = \frac{f(a) \cdot f(b)}{N} \qquad E_{12} = \frac{f(a) \cdot (N - f(b))}{N}$$

$$E_{21} = \frac{(N - f(a)) \cdot f(b)}{N} \qquad E_{22} = \frac{(N - f(a)) \cdot (N - f(b))}{N}$$

2.7. Proširenja standardnih mjera

PMI i DICE mjere primjenjive su samo na ekstrakciju digrama. Za ekstrakciju trigrama predlažu se sljedeće modifikacije PMI mjere:

$$I_1(a,b,c) = \log_2 \frac{P(abc)}{P(a)P(b)P(c)}$$

$$I_1'(a,b,c) = \log_2 \frac{P(abc)f(abc)}{P(a)P(b)P(c)}$$

$$I_2(a,b,c) = \frac{I(xy,z) + I(x,yz)}{2}$$

$$I_3(a,b,c) = \frac{I(x,y) + I(y,z) + I(x,z)}{3}$$

Mjere I_1 i I_1' su prirodna proširenja PMI mjera za digrame I i I'. Mjera I_2 predložena je u Boulis [3], dok je mjera I_3 predložena u Tadić, Šojat [31].

DICE mjera modificira se na sljedeći način [22]:

$$DICE(a,b,c) = \frac{3f(abc)}{f(a) + f(b) + f(c)}$$

Slična proširenja koriste se za ekstrakciju kolokacija iz tetragrama.

U radu Petrović et al. [25] navodi se heuristika za ekstrakciju kolokacija iz trigrama koja se bazira na pretpostavci da je *za različite vrste kolokacija potrebno koristiti različite mjere za ekstrakciju*. Ta heuristika dana je izrazom:

$$H(a,b,c) = \begin{cases} 2\log_2 \frac{P(abc)}{P(a)P(c)}, & X \in vr(b) \\ I_1(a,b,c), & X \notin vr(b) \end{cases}$$

Ova mjera je dala vrlo dobre rezultate, a razlog tome je sljedeći: samo jedan tip kolokacija od tri riječi sadrži stop riječi, i to kao srednju riječ trigrama. Vjerojatnost pojavljivanja stop riječi relativno je visoka, pa se ona ne uzima u obzir u gornjem izrazu. Faktor 2 koristi se da bi gornji i donji izraz doveli na istu skalu, tj. da bi se ujednačio optimalni prag oba izraza.

2.8. Usporedba mjera za ekstrakciju kolokacija

Uz ovakvo mnoštvo različitih mjera za ekstrakciju kolokacija postavljaju se pitanja:

- koja od postojećih mjera je *najbolja*?
- da li je moguće konstuirati bolje mjere?

Odgovor na drugo pitanje daje se u četvrtom poglavlju, dok se na prvo pitanje odgovor daje odmah. Što točno znači da je neka mjera za ekstrakciju kolokacija *bolja* od neke druge? To uvelike ovisi o kolekcijama dokumenata na koje se one primjenjuju. Osim toga, postoji više vrsta kolokacija, kao i samih definicija kolokacija. Ako se pretpostavi da je odabrana definicija iz poglavlja 2.2., te da je odabrana kolekcija dokumenata za koju se navedene mjere žele primjenjivati, i dalje ostaje otvoreno pitanje objektivnog mjerenja performansi. Nekoliko je postojećih rješenja, a jedna od najraširenijih mjera performansi koja uzima u obzir *preciznost* i *odziv* je mjera F_1.

2.8.1. Mjera F_1

Preciznost u terminima pretraživanja informacija je omjer broja prikupljenih relevantnih dokumenata i broja svih prikupljenih dokumenata.

$$preciznost = \frac{\left|\{prikupljeni\ dokumenti\} \cap \{relevantni\ dokumenti\}\right|}{\left|\{prikupljeni\ dokumenti\}\right|}$$

Odziv u terminima pretraživanja informacija je omjer broja prikupljenih relevantnih dokumenata i ukupnog broja relevantnih dokumenata.

$$odziv = \frac{\left|\{prikupljeni\ dokumenti\} \cap \{relevantni\ dokumenti\}\right|}{\left|\{relevantni\ dokumenti\}\right|}$$

F_1 mjera je harmonijska sredina preciznosti i odziva.

$$F_1 = \frac{2 \cdot preciznost \cdot odziv}{preciznost + odziv}$$

2.8.2. Primjena

Prethodno navedene formule treba prilagoditi evaluaciji mjere za ekstrakciju kolokacija. Preciznost postaje omjer broja ispravno prepoznatih kolokacija i ukupnog broja prijavljenih kolokacija. Odziv postaje omjer broja ispravno prepoznatih kolokacija i ukupnog broja kolokacija u kolekciji dokumenata. Ukupan broj kolokacija u kolekciji dokumenata je nepoznat (to je jedan od razloga nastajanja ovog rada), a može biti i do nekoliko stotina tisuća u velikim kolekcijama. Zbog toga se pribjegava aproksimacijama - stručna osoba iz kolekcije izdvaja određen broj *n*-grama koji sigurno jesu kolokacije (uzorak pozitivnih *n*-grama), te određen broj *n*-grama koji sigurno nisu kolokacije (uzorak negativnih *n*-grama). Svi *n*-grami uzorka moraju sadržavati jednak broj riječi, ovisno o mjeri koja se testira (očito mora odgovarati broju *n*-grama za koji je mjera dizajnirana). Na tom uzorku se procjenjuje uspješnost mjere, po sljedećim formulama [7]:

$$preciznost = \frac{\left|\{prijavljeni\ pozitivni\ n\text{-}grami\}\right|}{\left|\{prijavljeni\ pozitivni\ n\text{-}grami\}\right| + \left|\{prijavljeni\ negativni\ n\text{-}grami\}\right|}$$

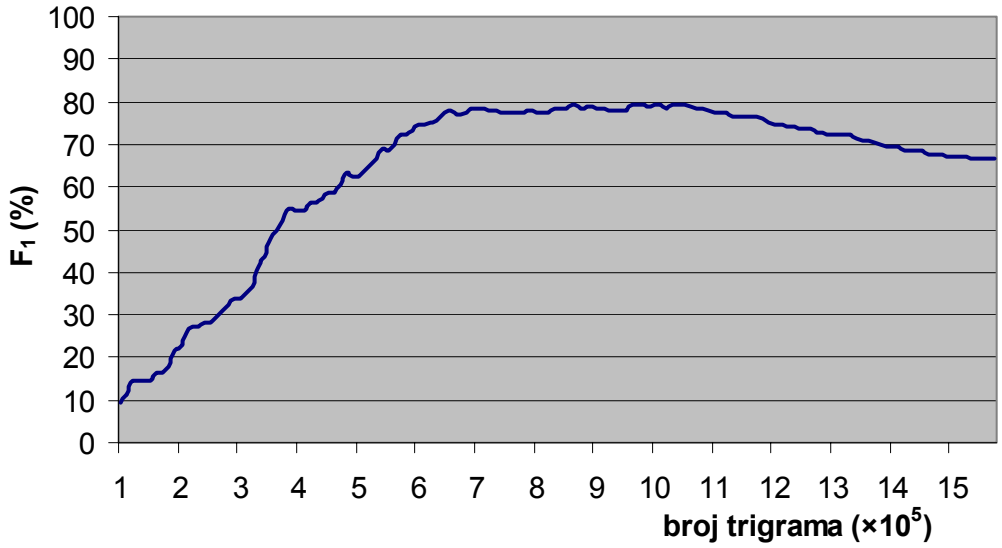
$$odziv = \frac{\left|\{prijavljeni\ pozitivni\ n\text{-}grami\}\right|}{\left|\{pozitivni\ n\text{-}grami\}\right|}$$

$$F_1 = \frac{2 \cdot preciznost \cdot odziv}{preciznost + odziv}$$

Za svaku od nabrojanih mjera ostaje otvoreno pitanje praga koji bi se trebao koristiti. Ako se koristi prenizak ili previsok prag, mjera će proglasiti sve *n*-grame kolokacijama, ili pak niti jedan. U oba slučaja mjera će imati nisku F_1 vrijednost. Variranjem praga mjere varira se broj *n*-grama koji se proglašavaju kolokacijama, a time i F_1 vrijednost koju mjera postiže. Kako se optimalan prag mjere utvrđuje eksperimentalno, kao F_1 vrijednost neke mjere za ekstrakciju kolokacija uzimat će se ona F_1 vrijednost koja se postiže *pri optimalnom pragu*. Ova metoda evaluacije korištena je u mnogim radovima, a jedan od razloga odabira ove mjere je mogućnost usporedbe rezultata sa [25].

2.8.3. Grafički prikaz performansi

Za prikaz performansi različitih mjera koristit će se grafovi poput sljedećeg:

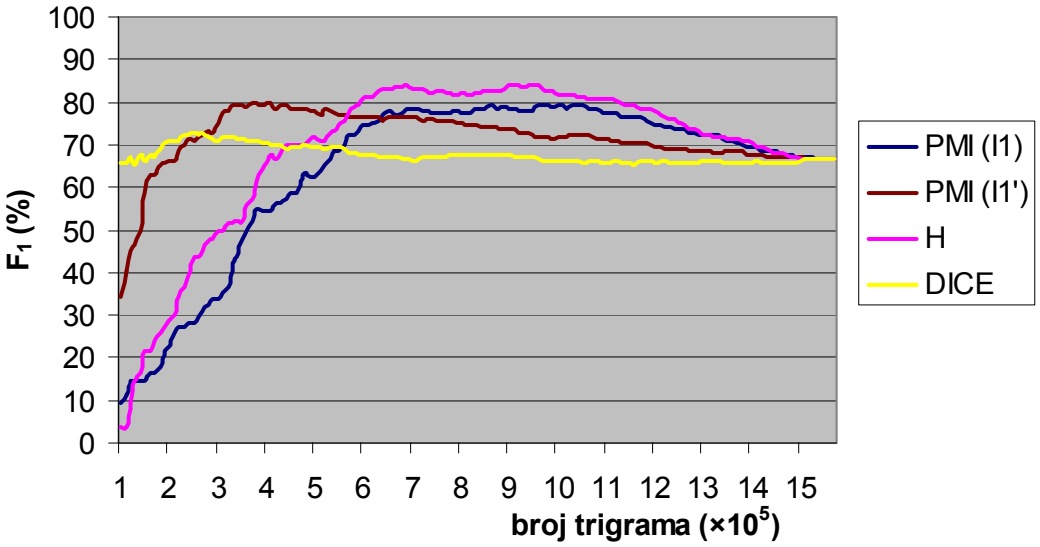


Slika 1: mjerenje performansi PMI mjere za trigrame

Na grafu je prikazana F_1 vrijednost mjere I_1 za ekstrakciju kolokacija iz trigrama, i to za svaki mogući prag te mjere. Umjesto iznosa praga, na *x*-osi nalazi se *broj n-grama koji su proglašeni kolokacijama pri nekoj vrijednosti praga mjere*. Time se omogućava usporedna analiza nekoliko različitih mjera na istom grafu. Na grafu se vidi da testirana mjera postiže maksimalnu vrijednost F_1 od oko 80% pri onom pragu za koji se kolokacijama proglašava između 850 tisuća i 1,1 milijun trigrama. Ako je poznat točan broj kolokacija u kolekciji dokumenata, ova informacija se može iskoristiti za dodatnu ocjenu kvalitete mjere.

Vrlo je važno napomenuti da ovaj graf pokazuje perfomanse mjere samo na jednoj kolekciji dokumenata, te samo na jednom uzorku pozitivnih i negativnih *n*-grama. Performanse ove mjere mogu biti potpuno drugačije na nekim drugim kolekcijama dokumenata, te to uvijek valja uzeti u obzir pri interpretaciji rezultata. U ovom slučaju radi se o istoj kolekciji dokumenata koja je korištena u radu Petrović et al. [25].

Slijedi grafički prikaz kojim se usporedno analizira više različitih mjera na istoj kolekciji dokumenata.



Slika 2: usporedba često korištenih mjera za trigrame

Na ovom se grafu vidi da mjera I_1 postiže maksimum od oko 80%, mjera I_1' postiže istu maksimalnu vrijednost, ali mnogo ranije, mjera DICE postiže jako loše rezultate (vrijednost F_1 ne mijenja se bitno s variranjem praga, stalno je ispod 75%), te na kraju heuristika H postiže najbolje rezultate, maksimum od preko 80% i to nakon maksimuma mjere I_1', a prije maksimuma mjere I_1. Točni iznosi optimalnih pragova mjera nisu bitni, te se ne mogu očitati direktno iz grafa.

Još jednom se naglašava da ovi grafovi (i svi ostali grafovi ovog tipa dalje u tekstu) ne odražavaju univerzalnu kvalitetu mjera, već se odnose samo na onu kolekciju dokumenata na kojoj su dobiveni (u ovom slučaju radi se o istoj kolekciji koja je korištena za generiranje prvog grafa). Osim toga, bitan je odabir uzorka pozitivnih i negativnih *n*-grama te kolekcije, jer se njime dobija samo aproksimacija točne F_1 vrijednosti, te bi neki drugi uzorak možda doveo do drukčijih rezultata (primjer za ovo nalazi se u 5. poglavlju).

3. Evolucijski algoritam

3.1. Povijest

U pedesetim godinama 20. stoljeća javlja se ideja korištenja Darwinovih načela evolucije za automatizirano rješavanje optimizacijskih problema. Ako je biološka evolucija stvorila živa bića koja svakodnevno rješavaju mnoge kompleksne probleme (najveći od kojih je sâmo preživljavanje), možda će simuliranje evolucije dovesti do pronalaska efikasnih rješenja raznih optimizacijskih problema. Mehanizmi biološke evolucije koji se koriste pri optimizaciji su razmnožavanje, mutacija, rekombinacija, prirodna selekcija i preživljavanje najboljih. Kroz stoljeće je rasla procesna moć računala, a time i primjenjivost i popularnost ovakvih optimizacijskih tehnika. Već u šezdesetima se raspoznaju tri grane: *evolucijsko programiranje (evolutionary programming), genetički algoritam (genetic algorithm)* i *evolucijske strategije (evolution strategies)*. U kasnim osamdesetim i ranim devedesetim godinama pojavljuju se još dvije paradigme - *genetičko programiranje (genetic programming)* i *klasifikatorski sustavi (classifier systems,* kasnije *learning classifier systems)*. Termini i načini klasificiranja su se mijenjali kako se područje razvijalo, zbog čega se jako teško snaći u pregršti pojmova koji se mogu naći u relevantnoj literaturi. Danas je ovih pet paradigmi poznato pod zajedničim nazivom *evolucijski algoritam (evolutionary algorithm)*, te zajedno s brojnim drugim optimizacijskim metodama (npr. *optimizacija mravljih kolonija - ant colony optimization)* pripadaju *evolucijskom računanju (evolutionary computation)*, važnom području *umjetne inteligencije (artificial intelligence)*.

3.2. Princip rada evolucijskog algoritma

Evolucijski algoritam najčešće se koristi za rješavanje kombinatorijalnih problema za koje nije poznat algoritam koji pronalazi egzaktno rješenje. Određivanjem ograničenja problema koji se pokušava riješiti definira se *prostor kandidata rješenja*, koji se zatim stohastički pretražuje. Najčešće se problem može formulirati tako da se traži aproksimacija parametara funkcije za koje ona postiže globalni minimum ili maksimum (funkcija može imati velik broj lokalnih ekstrema

koji predstavljaju zapreku tradicionalnim tehnikama optimizacije kao što je gradijentni spust). Analogija optimizacijskih problema s traženjem globalnog minimuma (ili maksimuma, vrsta ekstrema nije bitna) u daljnjem tekstu će se često koristiti za ilustriranje ključnih elemenata evolucijskog algoritma. Pretraga započinje iz više točaka - moguća rješenja optimizacijskog problema predstavljaju *jedinke* u *populaciji*. Time je smanjena vjerojatnost da će pretraga završiti u lokalnom ekstremu. Svaka jedinka ima svoj genetski materijal (genetski zapis, u nekim tehnikama koristi se izraz kromosom) koji je predstavljen nekom strukturom podataka (npr. binarnim brojevima). Nove jedinke u populaciji nastaju *križanjem* ili *mutacijom*. Križanje je rekombinacija genetskog materijala nekoliko jedinki (tradicionalno dvije) čime nastaju nove jedinke (tradicionalno jedna ili dvije), a mutacija je mijenjanje genetskog materijala jedne jedinke. Dobre jedinke imaju veće izglede za preživljavanje i/ili za prijenos svog genetskog materijala od loših jedinki. *Dobrotu* jedinke ocjenjuje se *funkcijom dobrote (fitness function)*. U svakoj iteraciji algoritma primjenom nekih od spomenutih operacija mijenja se sadržaj populacije, nastaje nova *generacija* populacije. Algoritam završava kad je ispunjen *uvjet zaustavljanja*. U tom trenutku populacija bi trebala sadržavati jedinke s većom prosječnom dobrotom od jedinki početne populacije, te se jedinke s najvećim dobrotama prijavljuju kao rezultat stohastičke pretrage.

U nastavku se nalazi pregled osnovnih elemenata evolucije.

3.2.1. Križanje

Pri križanju se uzimaju neki dijelovi genetskog materijala jedinki-roditelja i kombiniraju se u genetski materijal novih jedinki-djece. Struktura genetskog materijala mora biti takva da njeni dijelovi predstavljaju *osobine* jedinke [13, 23]. Bit križanja je kombiniranje osobina jedinki, kako bi se mogla stvoriti jedinka-dijete s boljim osobinama od svojih roditelja. Primjerice, ako genetski zapis jedinke ni na koji način ne uzima u obzir strukturu funkcije čiji minimum se traži, tada nije moguće iskoristiti eventualne pravilnosti u toj strukturi i postupak nema smisla - svodi se na nasumično pretraživanje prostora kandidata rješenja.

3.2.2. Mutacija

Neki evolucijski algoritmi ne koriste križanje, već samo mutaciju za stvaranje novih jedinki. U tom slučaju vrijedi komentar iz prve točke - genetski zapis jedinke mora odražavati njena svojstva, te postaje jako važno da mutacija dovodi do relativno malenih promjena u genetskom zapisu jedinke, kako bi nova jedinka imala svojstva slična jedinci od koje je nastala. Ako se križanje koristi u algoritmu, tada je primarna uloga mutacije uvođenje novog genetskog materijala u populaciju, kako bi se izbjeglo zaustavljanje u lokalnom ekstremu [13, 23].

3.2.3. Selekcija

Selekcija je mehanizam koji uzrokuje napredak populacije, tj. povećanje prosječne dobrote jedinke. Njome se određuje koje jedinke populacije prelaze u sljedeću iteraciju. Metoda selekcije jedinki koje preživljavaju mora biti pažljivo odabrana. Ako je selekcija prejaka (npr. loše jedinke uopće ne preživljavaju), postoji šansa da će u populaciji preostati samo jedinke vrlo sličnih karakteristika, čime se efektivno populacija kandidata rješenja zamjenjuje varijacijama jednog kandidata rješenja (koji bi mogao predstavljati lokalni ekstrem funkcije). Tada se kaže da je genetski materijal populacije osiromašen. Ako je pak selekcija preslaba (npr. loše jedinke u gotovo jednakoj mjeri preživljavaju i razmnožavaju se kao i dobre jedinke), tada postupak pretrage vrlo dugo traje i teško dolazi do dobrih jedinki.

Odabir jedinki koje sudjeluju u razmnožavanju (rekombinaciji genetskog materijala operatorom križanja) također je bitan aspekt selekcije. Neki od mehanizama su:

- Proporcionalno razmnožavanje (proportionate reproduction) - broj razmnožavanja neke jedinke proporcionalan je njenoj dobroti
- Odabir po rangu (ranking selection) - jedinke populacije sortiraju se silazno po dobroti, da bi se dobila rang-lista. Broj razmnožavanja neke jedinke obrnuto je proporcionalan njenom rangu u populaciji (što je rang niži, to je veći broj razmnožavanja neke jedinke).

- Turnirski odabir (tournament selection) - bira se slučajan broj jedinki iz populacije, a najbolje od njih postaju roditelji nove populacije. Ovaj postupak se ponavlja dok se ne stvori nova populacija.

Proporcionalno razmnožavanje i odabir po rangu često se izvode tako da se više kopija boljih jedinki ubaci u sljedeću generaciju populacije. Nakon toga nasumično se odabiru jedinke koje sudjeluju u rekombinaciji, ali bolje jedinke imaju veću vjerojatnost da budu odabrane, jer se pojavljuju više puta.

3.2.4. Uvjet zaustavljanja

Postupak generiranja nove generacije populacije ponavlja se dok se ne ispuni uvjet zaustavljanja. Najčešće korišteni uvjeti zaustavljanja su:

- Pronađeno je dovoljno dobro rješenje. Kriteriji „dovoljno dobrog“ rješenja moraju biti definirani.
- Potrošeni su resursi. Primjerice, procesorsko vrijeme možda se naplaćuje.
- Broj generiranja novih generacija je premašio unaprijed definiranu granicu.
- Istekao je unaprijed definiran interval vremena.
- Zaključeno je da daljnje iteracije neće dati poboljšanje rezultata. Kriteriji za to mogu biti procjena ljudskog stručnjaka ili npr. dosegnut određen broj iteracija nakon posljednjeg pronalaska jedinke s najvećom dobrotom.

3.3. Usporedba osnovnih tipova evolucijskog algoritma

3.3.1. Genetički algoritam

U ovoj metodi jedinke se tradicionalno prikazuju nizom bitova (takav zapis naziva se kromosom), no mogući su i nizovi znakova. Kromosomi mogu biti iste ili različitih duljina. Ako su duljine kromosoma različite, križanje je kompliciranije. Početna populacija bira se nasumično. U križanju sudjeluju dva roditelja i nastaju dva djeteta. Više je varijanti križanja – od polu-uniformne razmjene bitova (za

različiti bit vrši se razmjena vrijednosti s vjerojatnošću od 50%) preko križanja rezanjem i spajanjem (za svakog roditelja bira se jedan kontinuirani dio kromosoma, te njihovom zamjenom nastaju djeca) pa do križanja u jednoj ili više točaka (za svakog roditelja odabire se dio kromosoma jednake duljine, čijom zamjenom nastaju djeca duljine jednake duljini roditelja). Mutacija se obično izvodi kao izmjena jednog bita ili vrlo malog broja elemenata kromosoma, i to uz relativno malu vjerojatnost. Slijedi pseudokod ove metode.

```
generiraj početnu populaciju P
evaluiraj(P)
dok nije ispunjen uvjet završetka
    P := selektiraj(P)
    P := rekombiniraj(P)
    P := mutiraj(P)
    evaluiraj(P)
ispiši najbolje jedinke populacije P kao rješenje
```

Varijabilni dijelovi algoritma su veličina populacije, uvjet zaustavljanja, način selekcije, križanja i mutiranja, učestalost križanja i mutiranja, te broj jedinki koje se smatraju rješenjem. Evaluacija populacije koristi funkciju dobrote, pa je odabir funkcije dobrote jedan od najvažnijih aspekata ovog algoritma.

3.3.2. Evolucijsko programiranje

Ova metoda je slična genetičkom algoritmu, ali ne koristi rekombinaciju, već se oslanja samo na mutacije. Druga bitna razlika je – ne postoji standardna reprezentacija jedinki kao kod genetičkog algoritma. Mutacija najčešće dovodi do vrlo malih promjena, iako su moguće promjene proizvoljne veličine, pri čemu vjerojatnost za promjene opada proporcionalno relativnoj veličini promjene. Učestalost mutacija se smanjuje s vremenom, kako se jedinke približavaju globalnom minimumu. Kako se često ne zna koliko su jedinke daleko od globalnog minimuma, predložene su razne tehnike za rješavanje tog problema. Jedna od njih je *meta-evolucijska tehnika (meta-evolutionary technique)*, u kojoj odstupanje od mutacijske raspodjele evoluira zajedno s jedinkama. Slijedi pseudokod.

```
generiraj početnu populaciju P
evaluiraj(P)
dok nije ispunjen uvjet završetka
    P' := mutiraj(P)
    evaluiraj(P')
    P := selektiraj(P, P')
ispiši najbolje jedinke populacije P kao rješenje
```

U danom pseudokodu ne vidi se kako se mijenja način mutacije. Varijabilni dijelovi algoritma su veličina početne populacije, način generiranja početne populacije, način selekcije, odabir načina i učestalosti mutacije, načina promjene mutacije i učestalosti mutacije, uvjet završetka i broj jedinki koje se smatraju rješenjem. Odabir funkcije dobrote je važan, kao i odabir reprezentacije jedinke.

3.3.3. Evolucijska strategija

U ovoj metodi jedinke se prikazuju kao realni vektori, na koje se primjenjuje selekcija i mutacija (također postoje malobrojni primjeri u kojima se upotrebljava i rekombinacija). Mutacija se provodi dodavanjem slučajne vrijednosti (odabrane normalnom razdiobom) svakoj komponenti vektora. Selekcija je deterministička i ne ovisi o samom iznosu dobrote, već samo o rangu jedinke u populaciji. Jedinka nastala mutacijom (mutant) prelazi u novu generaciju populacije samo ako je veće dobrote od jedinke od koje je nastala (roditelja), inače roditelj prelazi u novu generaciju populacije. Ta strategija zove se *(1+1)ES*. Alternativno, iz roditelja nizom mutacija nastaje veći broj mutanata, a samo najbolja od svih jedinki (uključujući i roditelja) prelazi u novu generaciju populacije. Ova strategija naziva se *(m+1)ES*. Ako se roditelj ne smije prenijeti u novu populaciju, radi se o strategiji *(m, 1)ES*. Ovakav način zapisivanja generalizira se na *(m+n)ES* (tzv. *plus strategije*), odnosno *(m, n)ES* (tzv. *zarez strategije*), gdje je *m* broj nastalih mutanata, a *n* veličina nove populacije. Slijedi pseudokod.

```
generiraj početnu populaciju P
evaluiraj(P)
dok nije ispunjen uvjet završetka
    izaberi najboljih n jedinki iz P i stavi ih u P'
    iz P' reproduciraj m potomaka i stavi ih u P''
    P'' := mutiraj(P'')
    evaluiraj(P'')
    ako je odabrana plus strategija tada
          P := P' ∪ P''
    inače
          P := P''
ispiši najbolje jedinke populacije P kao rješenje
```

Varijabilni dijelovi algoritma su parametri *n* i *m*, odabir plus ili zarez strategije, način generiranja početne populacije. Odabir funkcije dobrote je važan.

3.3.4. Genetičko programiranje

Ova metoda vrlo je slična genetičkom algoritmu. Pseudokod tih metoda je identičan. Bitna razlika je u strukturi jedinke, te u konačnom cilju. Genetičkim programiranjem evoluiraju se *programi* koji rješavaju neki problem. Prirodna reprezentacija programa je *stablo* koje u unutarnjim čvorovima ima *operatore* dok u vanjskim čvorovima (listovima) ima *operande*. Pojam *program* u ovom kontekstu obuhvaća bilo kakav matematički izraz (funkciju) koji se može prikazati stablom na opisani način. Rekombinacija se izvodi odabirom po jednog podstabla svakog stabla, te njihovom zamjenom. Time nastaju dva djeteta koja se mogu drastično razlikovati od svojih roditelja. Mutacija odabire jedan čvor i mijenja ga, ili samo informaciju koju on sadrži, a moguće ju je i izostaviti.

3.3.5. Klasifikatorski sustavi

To su sustavi koji imaju mogućnost spoznaje događaja iz svoje okoline i reakcije na njih. Oni koriste genetičke algoritme da bi prilagodili svoje ponašanje prema promjenjivoj okolini.

Izgradnja klasifikatorskog sustava zahtijeva:

- Okolinu
- Receptore (*receptors*) koji sustavu daju informacije o okolini
- Djelovatelje (*effectors)* kojima sustav djeluje na okolinu
- Sam sustav – crnu kutiju koja živi u okolini, a sadrži receptore i djelovatelje

Reakcije na okolinu modeliraju se skupom *ako-onda* pravila. Jedno takvo pravilo naziva se *klasifikator*. Skup klasifikatora može se predstaviti kao početna populacija koja se može evoluirati korištenjem *učenja s podrškom*, te rekombinacijom i mutiranjem. Detaljniji opis algoritma učenja ovih sustava prelazi okvire ovog rada.

4. Korištene metode

U ovom poglavlju opisat će način na koji se evolucijski algoritam koristi za stohastičko pretraživanje velikog dijela prostora mogućih mjera za ekstrakciju kolokacija. Opisana metoda (u skladu sa 3. poglavljem) pripada kategoriji *genetičkog programiranja*. U poglavlju 4.1. navode se radovi sa sličnim temama. U poglavljima 4.2. - 4.6., te 4.8. opisuju se redom implementacije različitih elemenata korištenog algoritma genetičkog programiranja. U poglavlju 4.7. daje se pseudokod korištenog algoritma, detaljniji od pseudokoda danog u 3. poglavlju. Konačno, u poglavlju 4.9. dan je pregled parametara korištenog algoritma.

4.1. Srodni radovi

Područja pretraživanja informacija i evolucijskog računanja zastupljena su mnoštvom radova, no mnogo je manji broj radova koji kombiniraju genetičko programiranje s pretraživanjem informacija, a posebno dubinskom analizom teksta. Liu et al. [18] daju pregled raznih heurističkih metoda iz područja evolucijskog računanja i umjetne inteligencije općenito za ekstrakciju značajki iz teksta. Smith i Bull [27] opisuju primjenu genetičkog algoritma i genetičkog programiranja na konstrukciju i selekciju značajki pri pretprocesiranju dokumenata koje prethodi klasifikaciji. Atkinson-Abutridy et al. [1] opisuju primjenu genetičkog algoritma za evoluiranje *hipoteza* u svrhu ekstrakcije informacija iz teksta. Gordon et al. [11] opisuju primjenu genetičkog programiranja za evoluiranje težinskih funkcija koje se koriste za ocjenu relevantnosti dokumenata pri pretraživanju za specifičnim informacijama. Mnoge metode korištene u tom radu pokazale su se uspješnima u ovom diplomskom radu, a posebno odabir matematičkih operatora korištenih za izgradnju stabla funkcije.

Niz radova [4, 17, 19, 28, 35] opisuje metodu *parsimonijskog pritiska (parsimony pressure)* koja se koristi za redukciju prosječne veličine jedinki populacije u genetičkom programiranju. Primjena te metode pokazala se iznimno uspješnom u ovom diplomskom radu.

4.2. Jedinka

U idealnom slučaju jedna jedinka populacije predstavlja bilo kakvu funkciju $g : S \to \Re$, kako je opisano u poglavlju 2.5. Opisivanje općenite funkcije prezahtjevna je zadaća za računalo (postoji $|B|^{|A|}$ različitih funkcija $f : A \to B$, ako su skupovi A i B konačni), pa se umjesto toga koriste funkcije koje su kompozicija određenog broja odabranih jednostavnih funkcija. Jednostavne funkcije u ovom kontekstu su one koje se relativno lako implementiraju na računalu. Funkcija koju neka jedinka predstavlja bit će prikazana stablom. Unutarnji čvorovi stabla bit će operatori, dok će listovi stabla (vanjski čvorovi) biti operandi. Izračun vrijednosti stabla neke funkcije za odgovarajući argument vrši se rekurzivno počevši od korijena stabla. Ako je čvor koji se evaluira list (dakle, operand), vraća se njegova vrijednost. Inače, rekurzivno se računaju vrijednosti sve njegove djece, te se primjenjuje operator tog čvora na dobivene vrijednosti. Ovakav prikaz jedinke u potpunosti je u skladu s paradigmom genetičkog programiranja.

4.2.1. Operandi

Listovi stabla jedinke sadrže operande. To su:

- konstante
- statistički i lingvistički podaci *n*-grama

4.2.1.1. Konstante

Konstante se nasumično biraju iz intervala [-15, 15], do na tri decimalna mjesta. Navedeni interval odabran je zbog pretpostavke da će konstante malih apsolutnih vrijednosti (npr. -1, 2, 10) biti korisnije u izgradnji kvalitetnih jedinki (u najčešće korištenim mjerama sve konstante su takve), te da će se dobivene jedinke moći lakše interpretirati. Povećanjem širine intervala drastično bi opala vjerojatnost generiranja takvih konstanti, pa bi umjesto uniformne razdiobe bilo potrebno koristiti normalnu, koja je računalno zahtjevnija. Umjereno variranje širine intervala nije utjecalo na ponašanje opisanog algoritma (provjereno eksperimentom). Konstante većih

apsolutnih vrijednosti mogu se konstruirati od nekoliko manjih konstanti korištenjem operatora (npr. množenje).

4.2.1.2. Statistički i lingvistički podaci *n*-grama

Statistički podaci koji bi se mogli koristiti za procjenu nekog *n*-grama su:

- frekvencije svih dijelova tog *n*-grama (oznaka: f)
- vjerojatnosti pojavljivanja svih dijelova tog *n*-grama (oznaka: P)
- skup kategorija vrsta riječi svih riječi tog *n*-grama
- ukupan broj riječi i *m*-grama kolekcije dokumenata, za $m = 1, 2, ..., n$

Pri tome se dijelovima *n*-grama smatraju sve *k*-torke uzastopnih riječi tog *n*-grama, za $k = 1, 2, ..., n$. Frekvencija riječi, odnosno *n*-grama je broj pojavljivanja te riječi, odnosno *n*-grama u kolekciji dokumenata. Vjerojatnost riječi, odnosno *n*-grama aproksimira se omjerom frekvencije i ukupnog broja svih riječi (odnosno *n*-grama) u kolekciji dokumenata. Ukupan broj riječi u kolekciji je približno jednak broju *n*-grama u kolekciji, za male *n* (broje se višestruka pojavljivanja iste riječi ili *n*-grama). Stoga je dovoljno koristiti samo ukupan broj riječi kolekcije. Ako bi se ovaj sustav koristio u situacijama kad navedena pretpostavka ne bi bila istinita, prije upotrebe potrebno je proširiti definiciju jedinke. Također, nepotrebno je koristiti vjerojatnosti dijelova *n*-grama, jer se one mogu izvesti iz ostalih podataka. Već postojeći sustav koji uz lematizaciju svakoj riječi pridružuje skup svih kategorija vrsta riječi kojoj ona može pripadati opisan je u 2. poglavlju.

Tablica 2: operandi jedinke

operand	**objašnjenje**
N	ukupan broj riječi u kolekciji dokumenata
f(d)	frekvencija (broj pojavljivanja u kolekciji dokumenata) dijela *n*-grama d
konst	realan broj
vr(*w*) = S	uvjet „skup kategorija vrsta riječi riječi *w* jednak je skupu S“

4.2.2. Operatori

Operatori jedinke odabrani su tako da omoguće izgradnju što ekspresivnijeg aritmetičkog izraza, ali i pazeći na numeričku stabilnost. Odabran je minimalan broj operatora, kako bi algoritam pretraživao što manji prostor stanja. Uz četiri računske operacije, odabran je još (prirodni) logaritam, te poseban ternarni operator AKO. Dijeljenje s nulom se tolerira i rezultirat će vrlo velikim realnim brojem. Time se omogućava jedinkama koje povremeno dijele s nulom da prenesu svoje dobre osobine na ostatak populacije, u kasnijoj analizi rezultata takve jedinke neće se smatrati prihvatljivima. Kako ni logaritmiranje nije definirano na čitavom skupu realnih brojeva, umjesto funkcije ln(x) koristi se funkcija ln(|x|). Baza logaritma nije bitna jer se dijeljenjem s konstantom mogu dobiti rezultati logaritmiranja po svim ostalim bazama. Bilo bi korisno uključiti još operatore potenciranja i eksponenta, ali operator logaritmiranja najčešće može poslužiti umjesto tih operatora.

Operator AKO je ternarni operator uvjetne evaluacije, pomoću njega se koriste informacije o vrsti riječi. Prvi operand je uvjet oblika „*ako je k-toj riječi n-grama pridružen skup vrsta riječi S*“. Ostala dva operanda su aritmetički izrazi. Vrijednost AKO izraza jednaka je vrijednosti drugog operanda ako je uvjet prvog operanda istinit, inače je jednaka vrijednosti trećeg operanda. Formalno, funkcija $AKO : \{ \bot, \mathsf{T} \} \times \Re \times \Re \to \Re$, dana je sljedećim izrazom:

$$AKO(uvjet, a, b) = \begin{cases} a, & uvjet \\ b, & \neg uvjet \end{cases}$$

Tablica 3: operatori jedinke

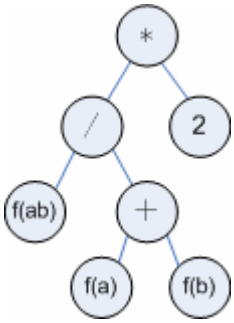
operator	**objašnjenje**
+	zbrajanje
–	oduzimanje
*	množenje
/	dijeljenje (dijeljenje s nulom rezultira velikim realnim brojem)
ln	ln (|x|) - prirodni logaritam apsolutne vrijednosti argumenta x
AKO	ternarni uvjetni operator

4.2.3. Primjeri

U nastavku su dani primjeri stabala nekih često korištenih mjera za ekstrakciju kolokacija. Operatori –, / i AKO nisu komutativni, redoslijed njihovih operanada na slici je uvijek slijeva nadesno.

Primjer 1 - DICE mjera za digrame.

$$g_1(S_N; S_K) = \frac{2 \cdot f(ab)}{f(a) + f(b)}$$

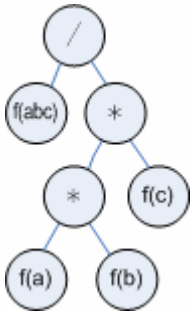


Slika 3: DICE mjera za digrame (prikaz stablom)

Napomena: faktor 2 se može izostaviti, jer on utječe samo na prag mjere.

Primjer 2 - PMI mjera za trigrame (prirodno proširenje, I_1)

$$g_2(S_N; S_K) = \log_2 \frac{P(abc)}{P(a) \cdot P(b) \cdot P(c)} \approx \log_2 \frac{f(abc) \cdot N \cdot N}{f(a) \cdot f(b) \cdot f(c)}$$

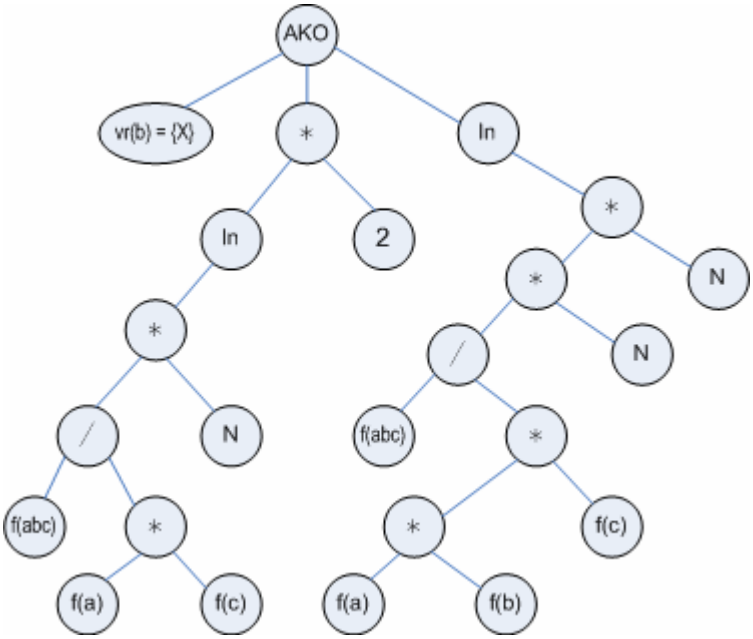


Slika 4: PMI mjera za trigrame (prikaz stablom)

Napomena: na slici su izostavljeni faktori N, te operator logaritmiranja. Oni utječu samo na prag mjere.

Primjer 3 - heuristika H [25]. Logaritmi po bazi dva zamijenjeni su prirodnim logaritmima, jer utječu samo na prag.

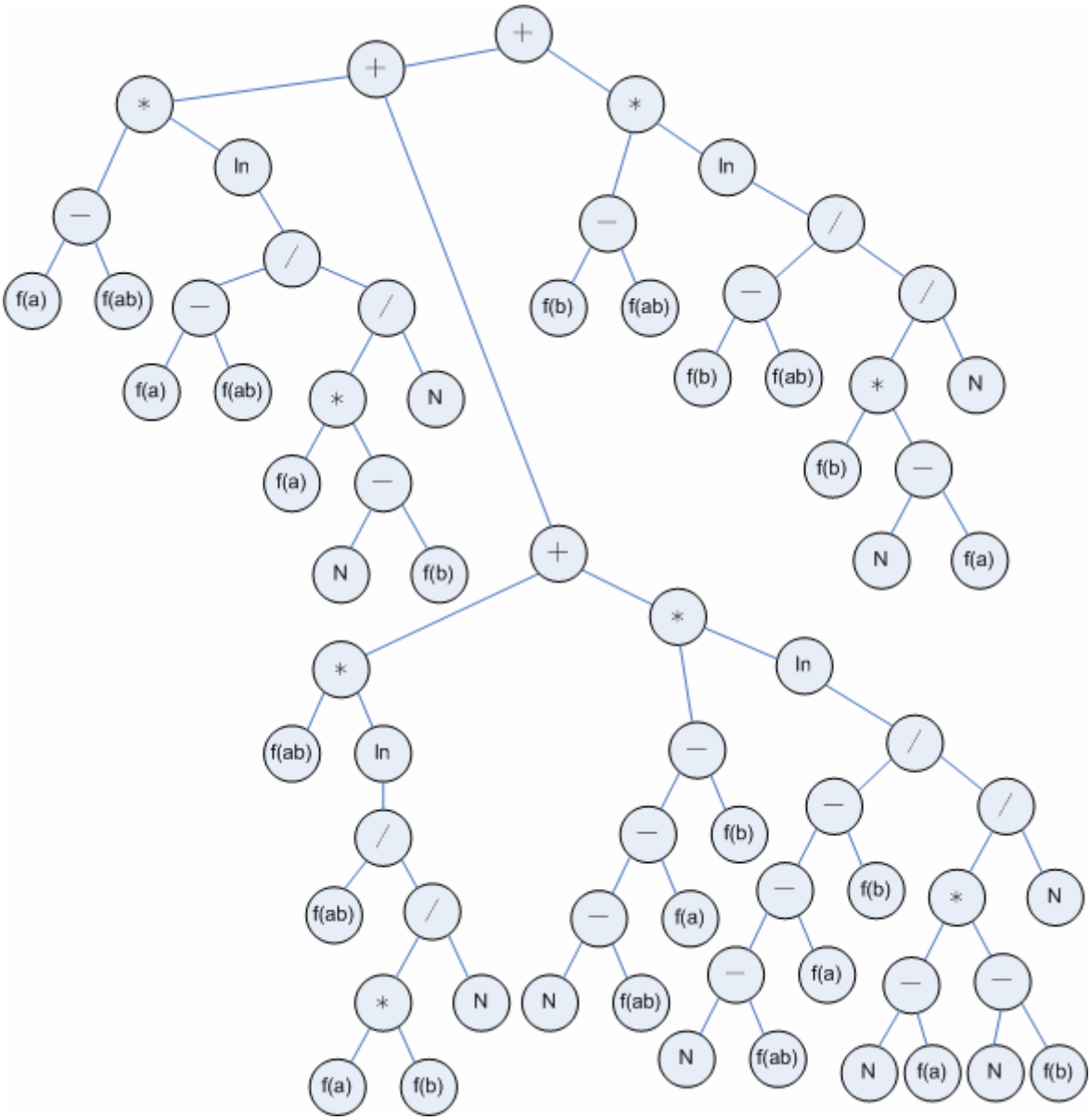
$$g_3(S_N;S_K)=\begin{cases}2\ln\dfrac{P(abc)}{P(a)P(c)}, & vr(b)=\{X\}\\ \ln\dfrac{P(abc)}{P(a)P(b)P(c)}, & vr(b)\neq\{X\}\end{cases}$$



Slika 5: heuristika H (prikaz stablom)

Primjer 4 - log-likelihood mjera za digrame. Kao i u prethodnom primjeru, baza logaritma utječe samo na prag mjere. Aproksimacije korištene za O_{ij} i E_{ij} navedene su u 2. poglavlju.

$$g_4(S_N; S_K) = \sum_{i,j} O_{ij} \ln \frac{O_{ij}}{E_{ij}}$$



Slika 6: log-likelihood mjera (prikaz stablom)

Iz ovog primjera vidi se da stablo jedinke može imati visoku razinu redundancije. U daljnjim istraživanjima potrebno je istražiti mogućnost alternativnih

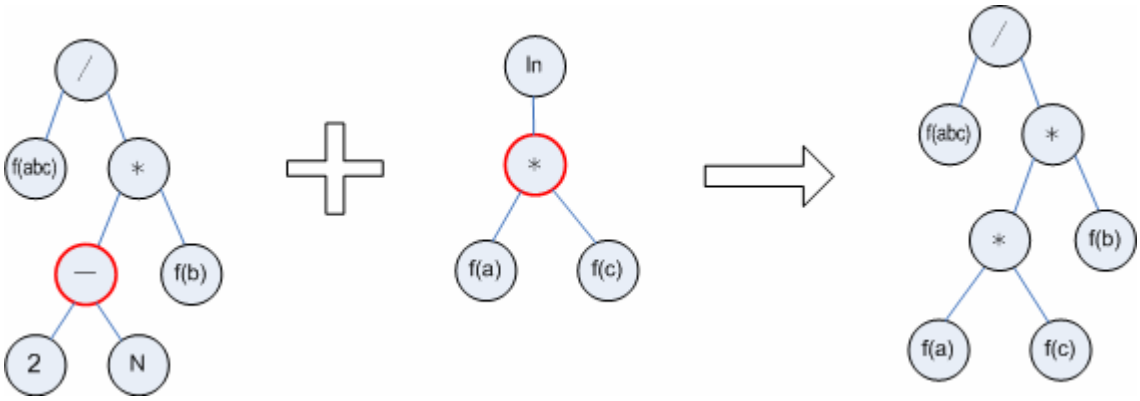
prikaza jedinki. Npr. korištenjem kolekcije stabala mogla bi se modelirati supstitucija varijabli izraza, pa bi prikazi izraza poput ovog mogli trošiti manje memorijskog prostora.

4.3. Genetski operatori

4.3.1. Križanje

Genetskim operatorom križanja od dvije jedinke (roditelji) nastaje nova jedinka (dijete) koja ima karakteristike roditelja. Kako su jedinke stabla, križanje je prirodno izvesti na sljedeći način:

1) odabire se bilo koji čvor prvog roditelja, osim čvora koji predstavlja uvjet operatora uvjetne evaluacije
2) odabire se bilo koji čvor drugog roditelja, osim čvora koji predstavlja uvjet operatora uvjetne evaluacije
3) dijete se konstruira od stabla prvog roditelja kojem se podstablo izabranog čvora zamjenjuje podstablom odabranog čvora drugog roditelja



Slika 7: primjer križanja

Ovaj postupak ne može dovesti do neispravno formirane jedinke jer su argumenti svih operatora realni brojevi (osim prvog argumenta operatora uvjetne evaluacije), a vrijednosti izraza koje predstavljaju sva podstabla jedinke također su realni brojevi (osim čvora koji je uvjet, tj. argument operatora uvjetne evaluacije). Ovakav postupak križanja u potpunosti je u skladu s definicijom operatora križanja genetičkog programiranja.

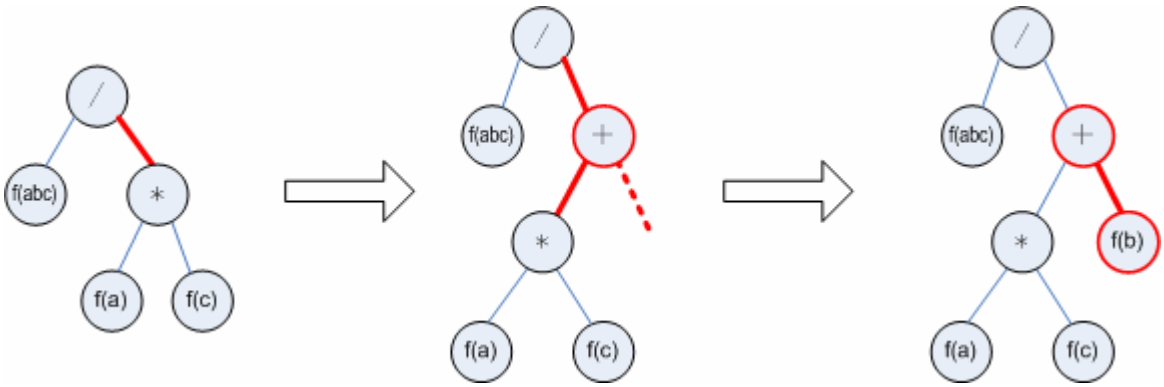
4.3.2. Mutacija

Operator mutacije odgovoran je za uvođenje malih promjena u izrazima (jedinkama) do kojih ne može doći križanjem (za unos novog „genetičkog materijala“). Mutacijom nastaje jedna od dvije promjene:

1) uklanjanje nekog čvora operatora (s vjerojatnošću 25%)
2) dodavanje novog čvora operatora (s vjerojatnošću 75%)

U slučaju uklanjanja operatora na njegovo mjesto postavlja se jedno od njegove djece, tj. jedan od njegovih argumenata čija vrijednost je realan broj.

Dodavanje novog operatora može se dogoditi na nekom bridu stabla, tj. na mjestu argumenta nekog operatora postavlja se čvor novog nasumično odabranog operatora. Argument koji je zamijenjen postaje argument upravo dodanog operatora. Također je moguće da operator koji se dodaje stablu postane novi korijen stabla, dok korijen stabla postaje njegov argument. Ako je dodani operator binaran ili ternaran, dodaje se potreban broj nasumično odabranih operanada (listova).



Slika 8: primjer mutacije

Ovakva definicija mutacije razlikuje se od klasične definicije mutacije u genetičkom programiranju (iako je mutaciju u genetičkom programiranju moguće i potpuno izostaviti). Takva mutacija bila bi izmjena nekog operatora ili operanda. Rani eksperimenti su pokazali nešto bolje rezultate uz korištenje ovako modificirane

mutacije. Ipak, izmjena nekog operanda se vrlo često dešava pri križanju (jer je broj listova velik).

4.4. Generiranje početne populacije

Svaka jedinka populacije početne populacije inicijalizirana je na sljedeći način, čime se dobija skup malenih i nasumično odabranih jedinki:

1) Stablo jedinke sadrži samo korijen. Korijen je inicijaliziran na:
   - slučajno odabranu konstantu, kako je opisano u poglavlju 4.2.1.
   - slučajno odabranu frekvenciju dijela *n*-grama, kako je opisano u poglavlju 4.2.1.
   - ukupan broj riječi u kolekciji dokumenata
2) Nasumično se odabire cijeli broj *n* iz intervala [0, 10>
3) Nad jedinkom se *n* puta uzastopce vrši mutacija

4.5. Funkcija dobrote

Da bi se mogla vršiti selekcija nad populacijom jedinki, potrebna je funkcija dobrote kojom se one mogu evaluirati. Funkcija dobrote u ovom slučaju svakoj jedinki pridružuje jedan realan broj. Veći broj dodjeljuje se uspješnijim jedinkama - onima koje bolje prepoznaju kolokacije. U tu svrhu koristi se F_1 mjera. Jedinka se testira njenom primjenom na *n*-grame neke kolekcije dokumenata, te se zatim korištenjem uzorka pozitivnih i negativnih *n*-grama računa njena F_1 vrijednost pri optimalnom pragu, kako je opisano u 2. poglavlju. Ta vrijednost mogla bi predstavljati dobrotu jedinke. U sljedećem potpoglavlju opisuje se modifikacija ovakve funkcije dobrote.

4.6. Faktor parsimonije

Stabla jedinki bi vrlo brzo (nakon nekoliko tisuća iteracija algoritma) mogla doseći veličine od nekoliko tisuća čvorova, a ubrzo zatim nekoliko desetaka tisuća čvorova i više. To ima velike posljedice na brzinu izvršavanja algoritma, a ne doprinosi kvaliteti rezultata. Osim toga, ako se dopusti neograničen rast stabala vrlo brzo se iscrpe memorijski resursi računala, ma koliki oni bili. Konačno, u svjetlu Occamovog načela poželjno je da dobiveni rezultati budu što jednostavniji. Iz

navedenih razloga potrebno je na neki način ograničiti veličine jedinki, ili barem usporiti njihov rast.

Više je strategija za rješavanje tih problema:

1. povećavanje učestalosti mutacije koja uklanja dijelove stabla
2. izvedba križanja koja bi favorizirala manje jedinke
3. implementacija algoritama reduciranja matematičkih izraza
4. periodičko uništavanje prevelikih jedinki
5. nagrađivanje malenih jedinki

Prva strategija bi naškodila kvaliteti operatora mutacije. Uloga mutacije je uvođenje novog genetskog materijala, a ovom strategijom bi se to iznimno rijetko događalo. U pravilu vrlo rijetko dolazi do mutacije, pa brisanje ionako ne bi bilo učinkovito. Jedna mogućnost je izvesti mutaciju na standardan način – kao operaciju promjene čvorova stabla - operandi bi se mogli mijenjati nasumično odabranim vrijednostima, operatori bi se mogli mijenjati slučajno odabranim operatorima (s istim brojem argumenata). To bi omogućilo transformaciju iz jedinki koje imaju nekoliko „pogrešnih“ elemenata u mnogo bolje jedinke, ali uz vrlo malu vjerojatnost. Osim toga, vrlo dobre jedinke ionako često sudjeluju u križanjima (jer prežive velik broj selekcija). Kao što je već navedeno, izmjena operanada često se događa pri križanju. Ova varijanta mutacije definitivno zaslužuje dodatno razmatranje i eksperimentiranje, no za sada se nije pokazala dovoljno dobrom, te nije implementirana. Na prvi pogled možda se čini da je mutacija jedini razlog povećanja prosječne veličine stabla, jer križanje ima podjednaku vjerojatnost stvaranja jedinki kako manjih tako i većih od prosječne veličine roditelja. To nije točno, jer su veće jedinke često i bolje jedinke, čime imaju veću šansu za preživljavanje selekcije. U daljnjim istraživanjima bilo bi dobro eksperimentirati sa mutacijama koje ovise o veličini jedinke.

Druga strategija bi se vrlo jednostavno implementirala - pri križanju se jedinki A neko podstablo zamjenjuje nekim podstablom jedinke B. Moguće je napraviti i obrnuto križanje - odabrano podstablo jedinke B može se zamijeniti

odabranim podstablom jedinke A. Pri svakom križanju mogle bi se generirati obje jedinke, manja od njih bi se zadržala, dok bi se veća odbacila. Kako su veće jedinke ponekad bolje od manjih, ova strategija bi smanjila kvalitetu križanja.

Treća strategija bi zahtijevala primjenu vrlo složenih algoritama uz veliko korištenje ekspertnog znanja matematike. Upitno je kakve rezultate bi postigla, no sigurno bi usporila rad programa. U 5. poglavlju se navodi primjer – profesionalan alat (Wolframova Mathematica 5.0) nije uspio reducirati veličinu jedne od najboljih jedinki.

Četvrta strategija se može implementirati na više načina - moguće je uvesti dodatnu selekciju koja bi uništavala najveće jedinke populacije, ali tek nakon nekog praga, kako se ne bi nepotrebno uništavale jedinke s dobrim genetskim materijalom. Druga varijanta je definiranje najveće dozvoljene veličine stabla. Ako bilo kojom operacijom nastane jedinka s većim brojem čvorova od dozvoljenog, ona se jednostavno zamijeni novom jedinkom, generiranom kako je opisano u poglavlju 4.5. Ova strategija zahtijeva pažljiv odabir najveće dozvoljene veličine. Jedinke od stotinjak čvorova trebale bi biti dovoljno velike, ali zbog moguće redundancije koja nastaje pri ovakvim heurističkim postupcima ipak je bolje dozvoliti postojanje većih jedinki. S druge strane, jedinke od desetak tisuća čvorova očito su prevelike. Ova strategija je jedina od nabrojanih koja donosi čvrsto postavljanje gornje granice utroška memorije i procesorskog vremena, te je implementirana kao sigurnosna mjera za stabilizaciju rada programa.

Peta strategija znači modifikaciju funkcije dobrote. Jedinke sa malenim brojem čvorova smatraju se boljima nego što jesu. Dodjeljuje im se veća vrijednost dobrote od vrijednosti F_1 mjere koju postižu. Na taj način relativno malene jedinke mogu preživjeti selekciju iako su nešto lošije od većih. S druge strane, ako jedinke postižu mnogo veću vrijednost F_1 mjere u odnosu na ostale, tolerirati će se njihova veličina. Ovo je najbolja strategija kontrole veličine jedinki jer je realizirana na prirodan način - umjesto zahvata koji narušavaju samu prirodu i način funkcioniranja evolucijskih metoda, ova strategija potiče evoluciju na stvaranje što elegantnijih matematičkih izraza samim time što se oni smatraju boljima od nezgrapnih i nepotrebno kompliciranih izraza.

Postoji mnogo načina na koje se ova modifikacija funkcije dobrote može realizirati. Modifikacija koja je korištena u ovom radu je dana formulom:

$$dobrota(j) = F_1(j) + \eta \frac{L_{\max} - L(j)}{L_{\max}}$$

Gdje su:

dobrota(j) - dobrota jedinke *j*

$F_1(j)$ - vrijednost F_1 mjere uspješnosti jedinke *j*

L_{max} - najveća dozvoljena veličina jedinke

L(j) - veličina jedinke *j* (broj čvorova)

η - faktor parsimonije

Desni pribrojnik izraza zove se pribrojnik parsimonije, te ima vrijednost približno jednaku η za jedinke veličine 1, linearno pada do vrijednosti 0 za jedinke najveće dozvoljene veličine i konačno pada ispod nule za eventualne jedinke veće od najveće dozvoljene veličine (takve jedinke će se evaluirati jer postoje kratko vrijeme prije uništavanja, točan pseudokod algoritma naveden je kasnije). Faktor parsimonije je broj koji određuje koliki pad F_1 će se tolerirati u korist manjih jedinki. Očito se zbog favoriziranja manjih jedinki neće dogoditi pad F_1 mjere konačnih rezultata veći od η. Vrijednost faktora parsimonije u eksperimentima se varirala od nule do 5%. Prevelik iznos mogao bi dovesti do kočenja napretka evolucije - ako sve promjene na nekoj populaciji ne mogu dovesti do povećanja mjere F_1 u iznosu većem ili jednakom vrijednosti pribrojnika parsimonije, evolucija neće proizvoditi veće jedinke.

4.7. Pseudokod korištenog algoritma

Veći dio implementacije elemenata genetičkog programiranja sada je definiran, te je moguće dati pseudokod korištenog algoritma detaljniji od onog u 3. poglavlju. Elementi koji nisu definirani su selekcija i uvjet zaustavljanja algoritma. Slijedi pseudokod algoritma. Funkcija *slučajan* vraća slučajan realan broj iz intervala $\left|0,1\right\rangle$.

```
inicijaliziraj početnu populaciju P
ako uključena opcija dodavanja poznatih jedinki u populaciju
    dodaj poznate jedinke u P
evaluiraj sve jedinke iz P
spremi najbolju od jedinki iz P kao rješenje
rješenje_gen := rješenje
k := 0
dok k < k_max
    k := k + 1
    iz P odaberi jedinke a, b i c tako da je c najlošija od njih
    nova_jedinka := križaj(a, b)
    evaluiraj (nova_jedinka)
    ako dobrota(nova_jedinka) > dobrota(rješenje)
          rješenje := nova_jedinka
    ako slučajan() < vjerojatnost_mutacije
          nova_jedinka := mutiraj(nova_jedinka)
          evaluiraj(nova_jedinka)
          ako dobrota(nova_jedinka) > dobrota(rješenje)
                rješenje := nova_jedinka
    ako duljina(nova_jedinka) > najveća_duljina
          nova_jedinka := generiraj_slučajnu_jedinku()
          evaluiraj(nova_jedinka)
          ako dobrota(nova_jedinka) > dobrota(rješenje)
                rješenje := nova_jedinka
    zamijeni jedinku c sa nova_jedinka
    ako rješenje bolje generalizira od rješenje_gen
          rješenje_gen := rješenje
          k := 0
ispiši(rješenje_gen)
ispiši(rješenje)
```

Vidi se da je korišten turnirski sustav selekcije. Preciznije, korišten je jedan od najčešće korištenih turnirskih sustava – *troturnirski sustav selekcije*, u kojem sudjeluju tri jedinke, te se najlošija od njih zamjenjuje (eventualno mutiranim) djetetom preostale dvije jedinke. Turnirski sustavi osiguravaju preživljavanje

najbolje jedinke, te općenito favoriziraju preživljavanje boljih jedinki, čime te jedinke dobijaju šansu da opet sudjeluju u križanju. Veličina turnira utječe na jačinu selekcije – veći turniri smanjuju šanse preživljavanja loših jedinki. Turnir samo tri jedinke znači najslabiju selekciju, a odabran je zbog procjene da je u ranim eksperimenata postizao najbolje rezultate. Vrlo lako je za potrebe budućih eksperimenata promijeniti način selekcije u bilo koji drugi.

4.8. Uvjet zaustavljanja algoritma

Algoritam se zaustavlja nakon određenog broja križanja tijekom kojih se nije pronašlo novo rješenje koje bolje generalizira. Što je *bolja generalizacija*? U ranim eksperimentima, u kojima se nije obraćala pažnja na sposobnost jedinke da generalizira, razvijeni algoritam je u 70 sati rada pronašao jedinku koja na određenoj kolekciji dokumenata, za određeni uzorak pozitivnih i negativnih *n*-grama postiže F_1 od 100%. Nije teško konstruirati takve mjere, baziraju se na radijalnim funkcijama, a primjer jedne takve mjere za digrame je:

$$\sum_{i=1}^{k}\left(\frac{1}{10^{-12}+(f(a)-f(a_i))^2+(f(b)-f(b_i))^2+(f(ab)-f(a_ib_i))^2}\right)$$

Gdje su $f(a_i)$, $f(b_i)$ i $f(a_ib_i)$, i = 1, ..., k odgovarajuće frekvencije i-tog pozitivnog digrama i njegovih riječi. Pod pretpostavkom da ne postoje dva digrama s identičnim statističkim podacima, te da je broj pozitivnih digrama *k* manji od 10^{12}, ova mjera postići će F_1 od 100%. Naime, za neki pozitivni digram jedan član sume poprima vrijednost 10^{12} (jer je suma kvadrata u nazivniku 0), dok svi ostali članovi sume poprimaju vrijednosti manje od 1, pa je dobivena vrijednost veća od 10^{12}. Za sve negativne digrame svi elementi sume poprimaju vrijednosti manje od 1, pa je dobivena vrijednost manja od *k*.

Očito će ova mjera postići jako loše rezultate na bilo kojem drugom uzorku digrama, jer je jednostavno memorizirala statističke podatke iz uzorka. Sve mjere s F_1 vrijednostima od preko 95% koje su dobivene ranim eksperimentima su na drugom uzorku iste kolekcije dokumenata postizale F_1 manji od 75%, što dokazuje da su te mjere „naučile“ uzorak korišten za evaluaciju jedinki. Da bi se ti efekti izbjegli, koristi se još jedan uzorak *n*-grama za ocjenu sposobnosti generalizacije

mjere. Taj uzorak će se u daljnjem tekstu zvati uzorkom za testiranje. On se ne koristi pri selekciji, pa ne može utjecati na stvaranje jedinki koje bi ga naučile. No, kako algoritam iterira, vrijednost F_1 mjere najbolje jedinke na tom uzorku će u početku postupno rasti. Nakon nekog vremena, najbolja jedinka populacije počinje učiti uzorak koji se koristi za evaluaciju funkcije dobrote, te zbog toga F_1 na uzorku za testiranje počinje padati. U tom trenutku sposobnost generalizacije najbolje jedinke je maksimalna. Izraz „A bolje generalizira od B“ u pseudokodu može se zamijeniti izrazom „A postiže veći F_1 rezultat na uzorku za testiranje od B“. Određen broj iteracija nakon zadnjeg poboljšanja generalizacije, algoritam zaključuje da je generalizacija gotova, te odustaje od daljnje pretrage i ispisuje rješenje.

4.9. Parametri

Kao što je vidljivo iz pseudokoda, u algoritmu je moguće varirati određen broj parametara. Za većinu ovih parametara nema nekih preporučenih vrijednosti tipičnih za genetičko programiranje, te je za sve potrebno eksperimentalno odrediti optimalne vrijednosti.

4.9.1. Veličina populacije

Ovaj parametar određuje ukupan broj jedinki u populaciji. Točan broj jedinki nije bitan, mnogo bitniji je red veličine. Rani eksperimenti ukazuju da je optimum za kratke vremenske intervale pretrage negdje između 100 i 2000.

4.9.2. Broj iteracija prije odustajanja

Ovaj parametar određuje broj iteracija nakon kojeg algoritam zaključuje da je generalizacija završila, te odustaje od daljnje pretrage.

4.9.3. Vjerojatnost mutacije

Nakon svake operacije križanja s određenom vjerojatnošću se primjenjuje mutacija na novonastalu jedinku. Optimalna vrijednost nije poznata, ali uočeno je da male varijacije ne utječu na rezultate (bitan je samo red veličine).

4.9.4. Uključivanje poznatih mjera u početnoj populaciji

U početnoj populaciji nalaze se nasumično generirane jedinke. Da bi se skratio postupak pretrage, te da bi se osigurala kvaliteta rješenja jednaka barem kvaliteti već poznatih mjera, u početnu populaciju mogu se dodati već poznate mjere.

4.9.5. Najveća dozvoljena veličina jedinke

Kao što je već navedeno, veličina jedinke je ograničena, sve jedinke koje prijeđu ograničenje su prvo evaluirane, eventualno zabilježene kao rješenje, te nakon toga uništene. Postavljanje ovog ograničenja na vrijednost 1000 je dovoljno dobro za generiranje vrlo kvalitetnih jedinki.

4.9.6. Faktor parsimonije

Ovaj parametar već je detaljno objašnjen. Optimalna vrijednost je nepoznata, no ne preporuča se gubljenje vrijednost F_1 mjere od preko 5%.

5. Rezultati

Kao što je već objašnjeno, za pokretanje pretrage potrebna su dva uzorka *n*-grama poznatih klasifikacija. Ekstrakcija uzoraka iz neke kolekcije dokumenata, kao i sama priprema kolekcije dugotrajan je posao koji zahtijeva rad jedne ili više stručnih osoba. Poželjno je da broj *n*-grama u svakom skupu bude barem 100, kako bi se izbjegle statističke anomalije, što dovodi do ukupnog broja od 400 *n*-grama poznate klasifikacije. Zbog navedenih razloga za potrebe ovog rada pripremljena je samo jedna kolekcija od 7008 pravnih dokumenata Narodnih Novina, korišten u [25] s preko milijun riječi, 167 911 različitih lema, 1 816 121 različitih digrama, te 4 656 013 različitih trigrama. Za potrebe stvaranja uzoraka klasificirano je preko 400 trigrama te kolekcije. Svi pokusi odvijali su se isključivo nad trigramima jer su poznate mjere za trigrame kompleksnije od mjera za digrame, dok su količine dostupnih materijala za tetragrame nedovoljne.

Program je pokrenut na tri računala (takt procesora u prosjeku 2.5 GHz). Algoritam pretrage pokretao se sa različitim parametrima, pamtio je najbolje pronađene jedinke, te na kraju kao rješenje prijavio dvije jedinke - jedinku s najboljim performansama na uzorku za testiranje, te jedinku s najboljim performansama na uzorku za evaluaciju dobrote. Nakon završene iteracije program je pokretao novu pretragu s drukčijim parametrima. Pregled parametara dan je u sljedećem potpoglavlju. Nakon 160 sati neprekidnog izvođenja programa prikupljeno je oko 800 rezultata (za oba uzorka). Uz rezultate pretrage, prikupljen je velik broj međurezultata koje je program zapisivao za vrijeme trajanje pretraga, kako ne bi bili izgubljeni u slučaju nestanka električne energije ili iz drugih razloga (jedna pretraga može trajati satima). To su jedinke koje su u nekom trenutku pretrage bile najbolje, te su često vrlo slične konačnom rezultatu pretrage, uglavnom uz nešto manji rezultat F_1 mjere.

5.1. Vrijednosti parametara pretrage

Značenje pojedinih parametara objašnjeno je u 4. poglavlju, ovdje se donosi samo pregled vrijednosti korištenih u eksperimentima.

5.1.1. Veličina populacije

Broj jedinki u populaciji birao se uniformno iz sljedećeg skupa:

$\{50,\ 100,\ 200,\ 300,\ 500,\ 700,\ 1\,000,\ 2\,000,\ 10\,000,\ 50\,000\}$

Nakon završenih eksperimenata nije utvrđena optimalna vrijednost.

5.1.2. Broj iteracija prije odustajanja

Ovaj parametar bira se uniformno iz sljedećeg skupa:

$\{10\,000,\ 20\,000,\ 100\,000,\ 200\,000,\ 1\,000\,000,\ 10\,000\,000\}$

Ovo se pokazalo lošim izborom. Generalizacija je nakon 200 tisuća iteracija sa sigurnošću završavala, te je ostatak iteracija nepotrebno trošio procesorsko vrijeme. Ovo je bilo teško utvrditi prije samih eksperimenata.

5.1.3. Vjerojatnost mutacije

Vjerojatnost mutacije bira se uniformno iz sljedećeg skupa:

$\{0.001,\ 0.005,\ 0.01,\ 0.05,\ 0.1,\ 0.2,\ 0.3\}$

Nije utvrđena optimalna vrijednost.

5.1.4. Uključivanje poznatih mjera u početnoj populaciji

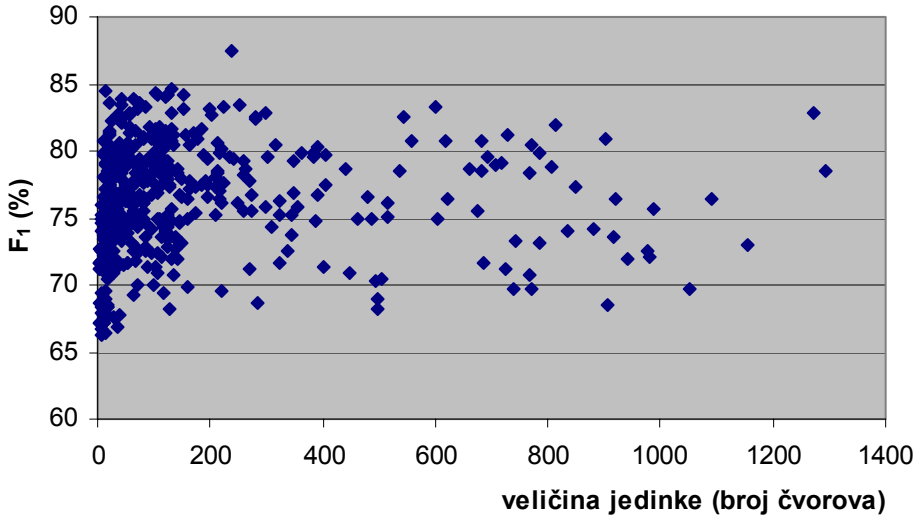
U početnu populaciju dodaju se tri poznate mjere, s vjerojatnošću od 50% (dodaju se sve tri ili niti jedna). Mjere koje se dodaju su PMI (verzija I_1), DICE, te heuristička mjera H. Kasnije u poglavlju se nalazi detaljna analiza uspješnosti ove metode.

5.1.5. Najveća dozvoljena veličina jedinke

Slijedeći Occamovo načelo, dozvoljena veličina jedinke postavlja se na nešto niže vrijednosti od preporučene (1000), u nadi da će se tako pronaći jednostavnije jedinke. Ovaj parametar bira se uniformno iz sljedećeg skupa:

$\{20,\ 100,\ 500,\ 1000\}$

Nakon prikupljanja rezultata istražena je ovisnost kvalitete rješenja o duljini. Odnos se najbolje vidi iz sljedećeg grafa:



Slika 9: distribucija F1 u ovisnosti o veličini jedinke

Može se uočiti da kvaliteta jedinke ne ovisi primjetno o njenoj veličini. Također se može uočiti da je malen broj jedinki s više od 1000 čvorova prijavljen kao rješenje pretrage. Ova zapažanja opravdavaju uvođenje ograničenja na veličinu jedinke.

5.1.6. Faktor parsimonije

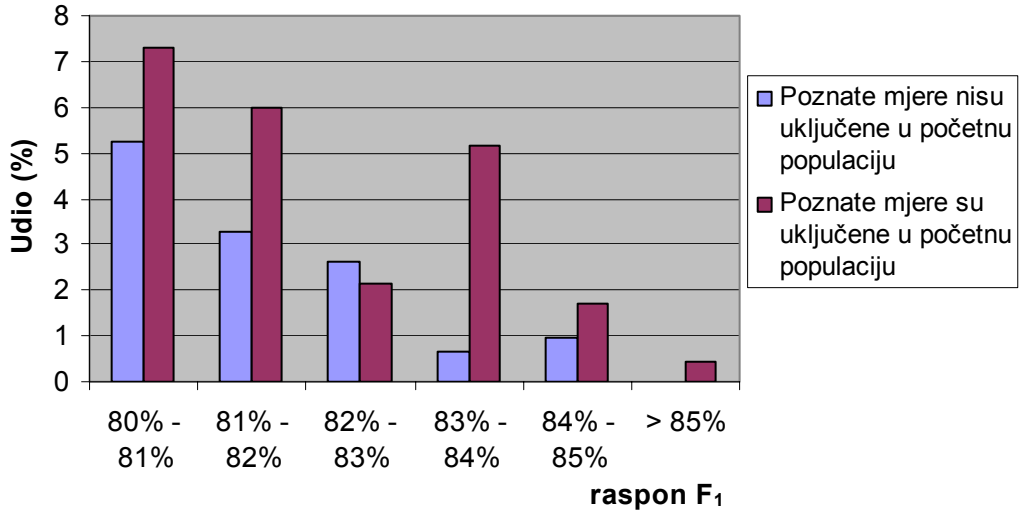
U 50% pretraga, ovaj parametar se uopće ne koristi. U preostalih 50% slučajeva, uniformno se bira iz sljedećeg skupa:

$$\{0.00001,\ 0.0001,\ 0.001,\ 0.005,\ 0.01,\ 0.02,\ 0.05\}$$

Dakle, tolerira se gubitak na vrijednosti F_1 od maksimalno 5%.

5.2. Uspješnost pretrage

Oko 20% svih jedinki rezultata postiže F_1 od preko 80%. U skupu rezultata pretraga koje su uključile poznate mjere u početnu populaciju taj udio je 23%, dok je u skupu rezultata pretraga koje nisu uključile poznate mjere u početnu populaciju taj udio 13%. Detaljan prikaz dan je sljedećim grafom:



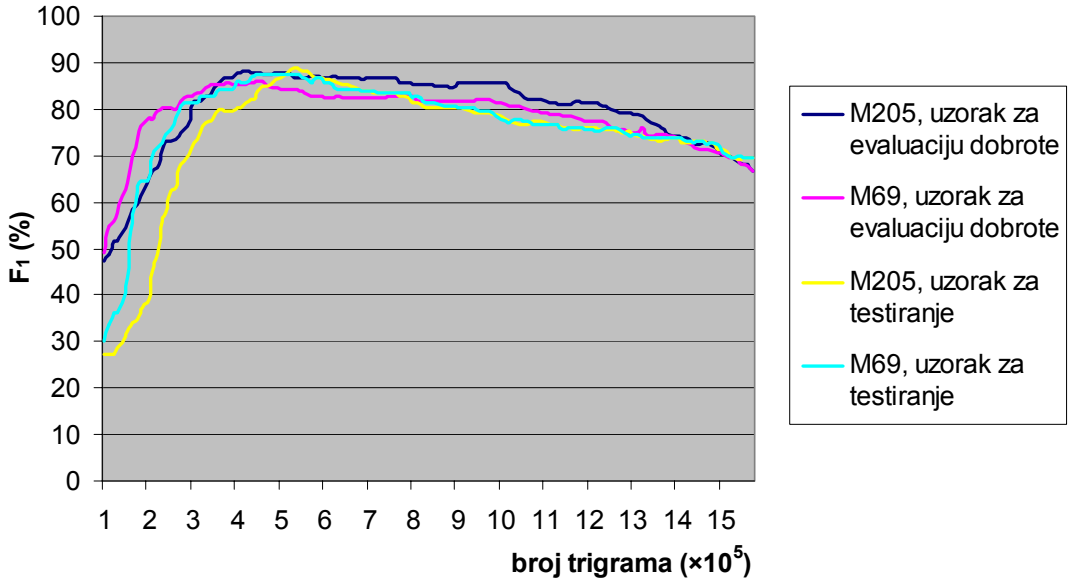
Slika 10: udio kvalitetnih jedinki u skupu rezultata

5.3. Najbolje mjere

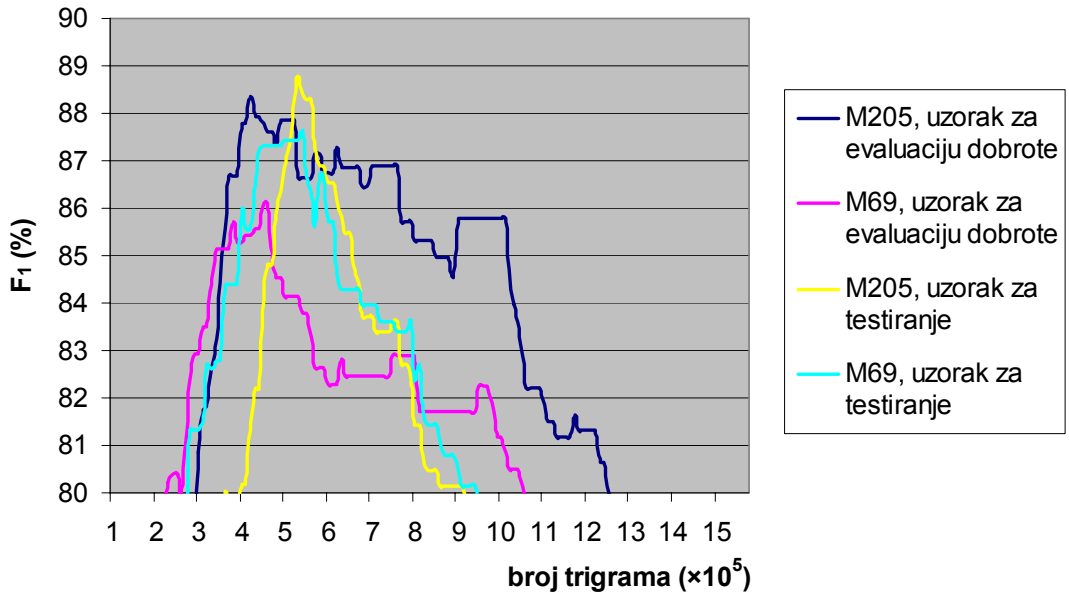
Svi rezultati i međurezultati su uzeti u obzir (postoji šansa da su međurezultati, iako lošiji od konačnih rezultata, nešto kraćeg ili elegantnijeg zapisa). Ukupno 50 mjera s F_1 od preko 85% je prikupljeno, veličina od 34 do 1361.

Najbolja među njima ima F_1 od 87.45%, veličine je 238 čvorova. Sadrži nekoliko AKO operatora, no neki od njih su bili suvišni (npr. argumenti su drugog AKO operatora s istim uvjetom), te su uklonjeni. Ostala podstabla su raspisana u izraze (infiks oblik), te analizirana. Niti jedan od dobivenih izraza nije bilo moguće pojednostavniti, čak niti uz pomoć profesionalnih alata kao što je Wolframova Mathematica 5.0 (korištena je funkcija FullSimplify). Upitna je korisnost toliko nezgrapne mjere, a osim toga postavlja se pitanje da li je mjera zaista univerzalno dobra, ili je imala slučajno dobar rezultat na dva korištena uzorka. Kako konvencionalne metode reduciranja izraza nisu uspjele, ta mjera je vraćena natrag u postupak pretrage. Ubačena je kao jedna od mjera u početnoj populaciji, te se nekoliko puta pokrenula pretraga uz variranje parametara. Faktor parsimonije podizan je do čak 10%, u nadi da će se naći dobra i elegantna aproksimacija. Dobivena su tri zanimljiva rezultata - dvije mjere s F_1 od 88.37%, duljina 256 i 205, te jedna mjera s F_1 od 86.23%, ali duljine samo 69. Ove mjere će se u daljnjem tekstu

nazivati redom M_{256}, M_{205} te M_{69}. Zapisi svih navedenih jedinki dani su u dodatku, zajedno s popisom korištenih parametara pretrage. Mjere M_{256} i M_{205} ponašaju se identično, zbog čega se navodi grafički prikaz performansi mjera M_{205} i M_{69}.

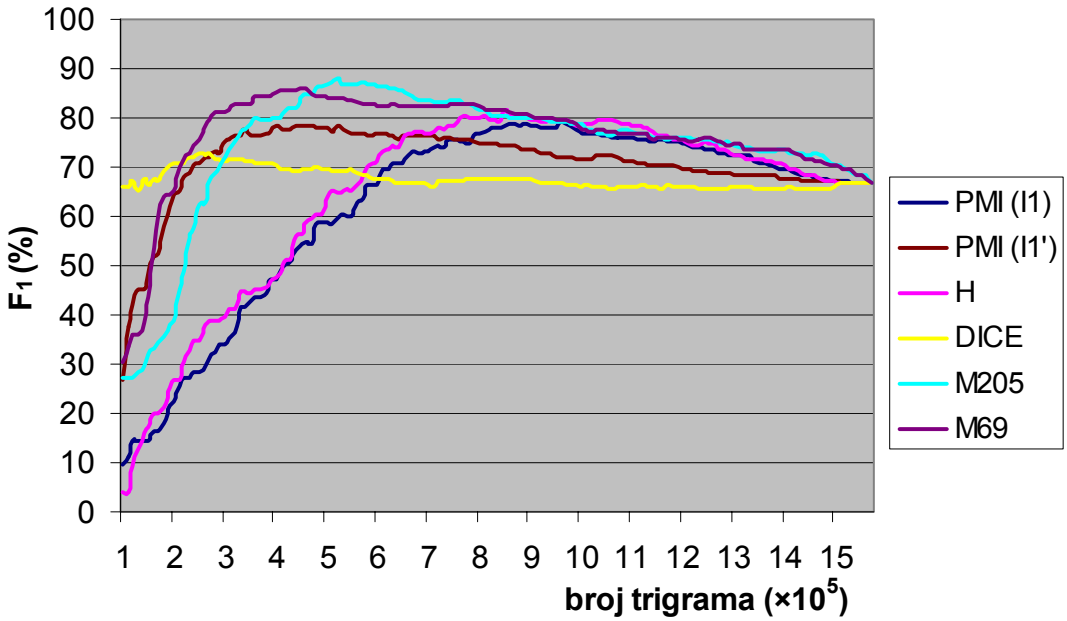


Slika 11: performanse najboljih mjera

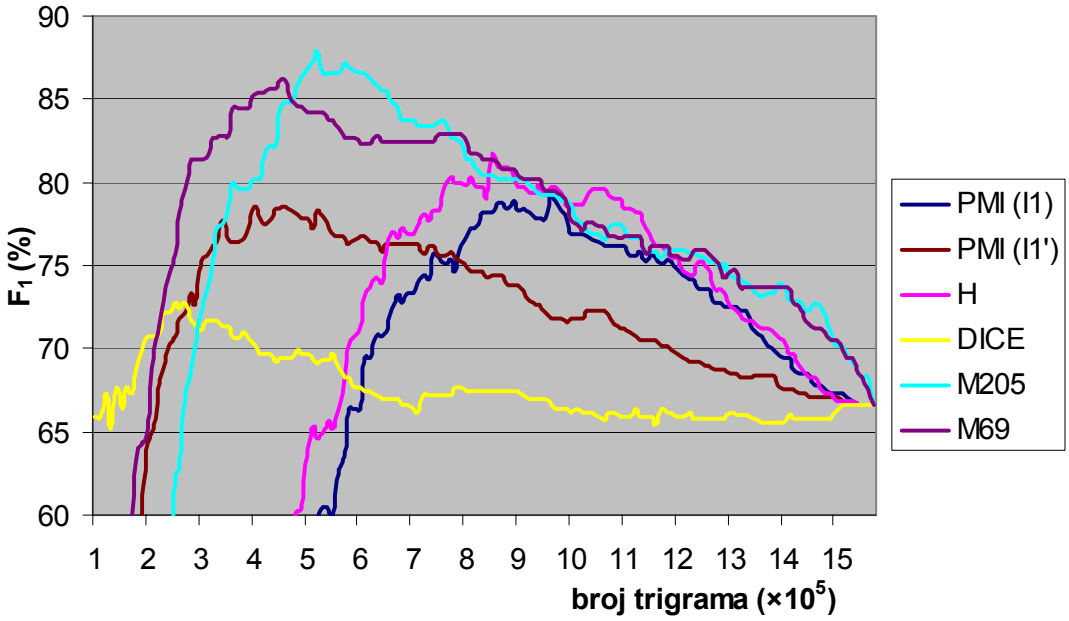


Slika 12: performanse najboljih mjera (detalj)

Interpretiranje ovakvih grafova objašnjeno je u 2. poglavlju. Grafovi prikazuju F_1 vrijednosti navedenih mjera za svaki mogući prag, mjereno na kolekciji trigrama nakon primjene filtera frekvencije i vrste riječi. Prag mjere može se odabrati proizvoljno, tako da se kolokacijama proglašava bilo koji broj trigrama iz kolekcije, što znači da je najvažnije gledati maksimume mjera - bolje mjere imaju veće globalne maksimume. Kao što je već objašnjeno, mjerenje performansi se aproksimira korištenjem uzoraka klasificiranih trigrama, zbog čega su za svaku mjeru prikazane dvije različite krivulje. Za svaki prag mjere, realna F_1 vrijednost jednaka je manjoj od vrijednosti na obje krivulje. Slijedi usporedba ovih mjera sa ostalim često korištenim mjerama.



Slika 13: usporedba M_{205} i M_{69} s poznatim mjerama



Slika 14: usporedba M_{205} i M_{69} s poznatim mjerama (detalj)

Korištena su oba uzorka, a prikazane su krivulje minimalnih vrijednosti. Vidi se da je najbolja mjera M_{205}, slijedi M_{69}, zatim heuristika H, pa obje verzije PMI mjere, te na kraju DICE (ostale mjere, kao npr. frekvencijska, log-likelihood i hi-kvadrat postižu rezultate u rangu DICE mjere).

5.4. Najjednostavnije mjere

U potrazi za „logičnim“, elegantnim mjerama pogodnima za daljnju analizu i raspravu, pretražene su sve jedinke s F_1 od preko 84%, ali veličine 50 ili manje. Prikupljeno je 16 takvih jedinki. Samo dvije jedinke imaju F_1 preko 85%, uz veličine od 34 i 39. No, te jedinke pri evaluaciji nekih trigrama uzrokuju numeričke pogreške - postižu beskonačnu vrijednost u nekom od podizraza (dijele sa izrazom f(a)-f(ab)). To ne znači nužno da su neupotrebljive - računalo zna računati s beskonačnošću, ali zbog toga se smatraju neelegantnima. Sljedeće tri jedinke imaju F_1 od 84.76% (veličina 31), 84.61% (veličina 35), te 84.54% (veličina 13). Preostalih 11 jedinki imaju F_1 manji od 84.3%. Jedinka od samo 13 čvorova je naročito zanimljiva, ona predstavlja izraz:

$$\begin{cases} -0.423\dfrac{f(a)f(c)}{f(abc)^2}, & vr(b) = \{X\} \\ 1-\dfrac{f(b)}{f(abc)}, & vr(b) \neq \{X\} \end{cases}$$

Parametri pretrage koji su doveli do pronalaska ove mjere, kao i pravi zapis mjere nalaze su u dodatku. Ova mjera u daljnjem tekstu nazivat će se M_{13}.

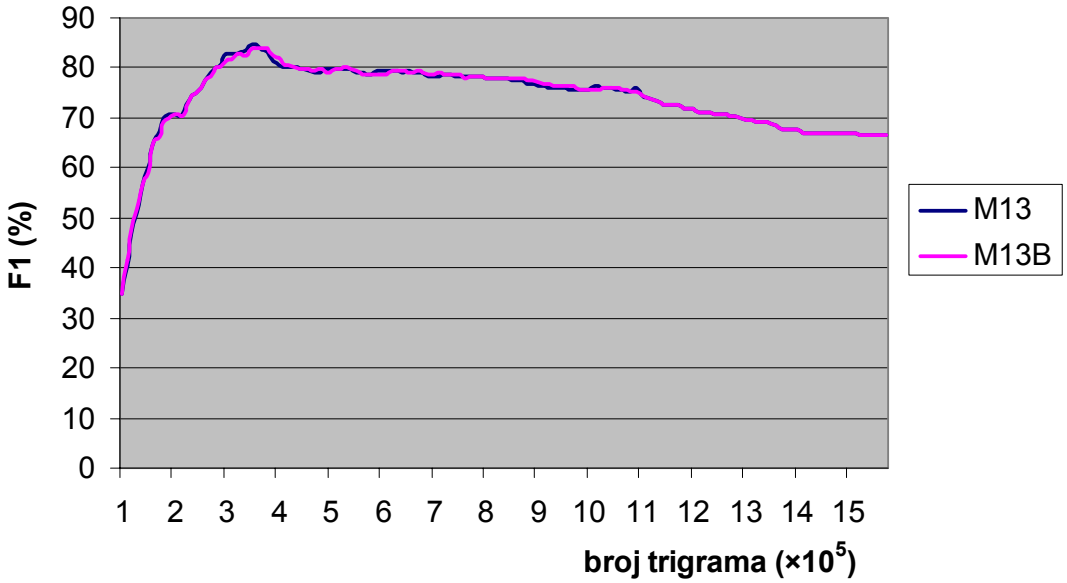
Prag pri kojem ova mjera postiže maksimalan F_1 rezultat je negativan, između -641 i -685. Gornji dio izraza se primjenjuje ako je srednja riječ trigrama stop riječ (najčešće veznik). U tom slučaju frekvencija srednje riječi se ne uzima u obzir, a sam izraz izgleda kao negativna recipročna vrijednost modificiranog PMI izraza za digrame (cite). Donji dio izraza primjenjuje se ako srednja riječ nije stop riječ. U tom slučaju frekvencije prve i treće riječi se ne uzimaju u obzir, a sam izraz je sličan gornjem izrazu - negativna recipročna vrijednost omjera frekvencija pojedinih riječi i frekvencije samog trigrama. Kako je u brojniku samo jedna frekvencija riječi, tako je u nazivniku jedan faktor *f(abc)* manje nego u gornjem izrazu. Pribrojnik 1 u donjem izrazu ne igra nikakvu ulogu, jer je raspon praga mjere velik, te se može izostaviti (provjereno eksperimentom). Faktor 0.423 u gornjem izrazu je naročito zanimljiv, jer je vrlo blizu broju 0.5, odnosno faktoru 2 u nazivniku. Nakon ovih aproksimacija dobija se sljedeći izraz:

$$\begin{cases} -\dfrac{f(a)f(c)}{2f(abc)^2}, & vr(b) = \{X\} \\ -\dfrac{f(b)}{f(abc)}, & vr(b) \neq \{X\} \end{cases}$$

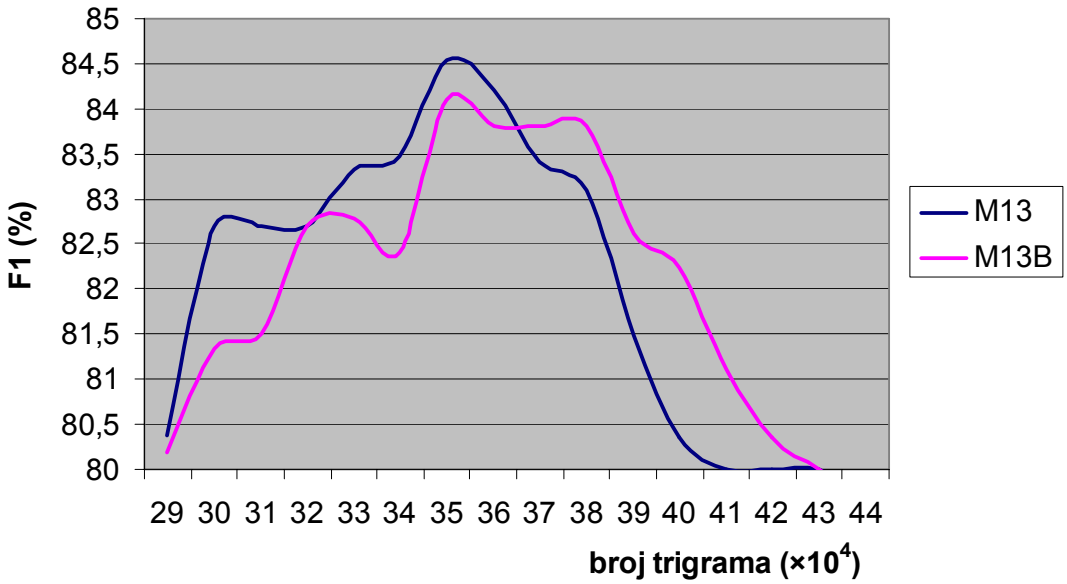
Taj izraz postiže rezultate identične rezultatima sljedećeg izraza, jer je za uspješnost mjere bitan samo rang *n*-grama, a ne konkretne vrijednosti:

$$\begin{cases} \dfrac{2f(abc)^2}{f(a)f(c)}, & vr(b) = \{X\} \\ \dfrac{f(abc)}{f(b)}, & vr(b) \neq \{X\} \end{cases}$$

Ovaj izraz zvat će se mjerom M_{13B}. Slijedi prikaz performansi mjera M_{13} i M_{13B}.

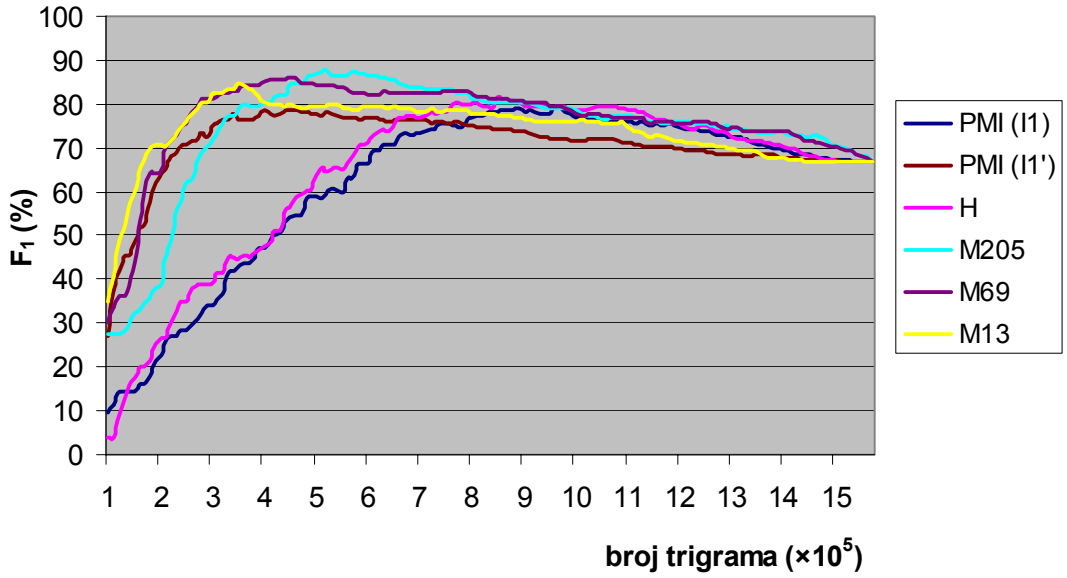


Slika 15: usporedba mjera M_{13} i M_{13B}

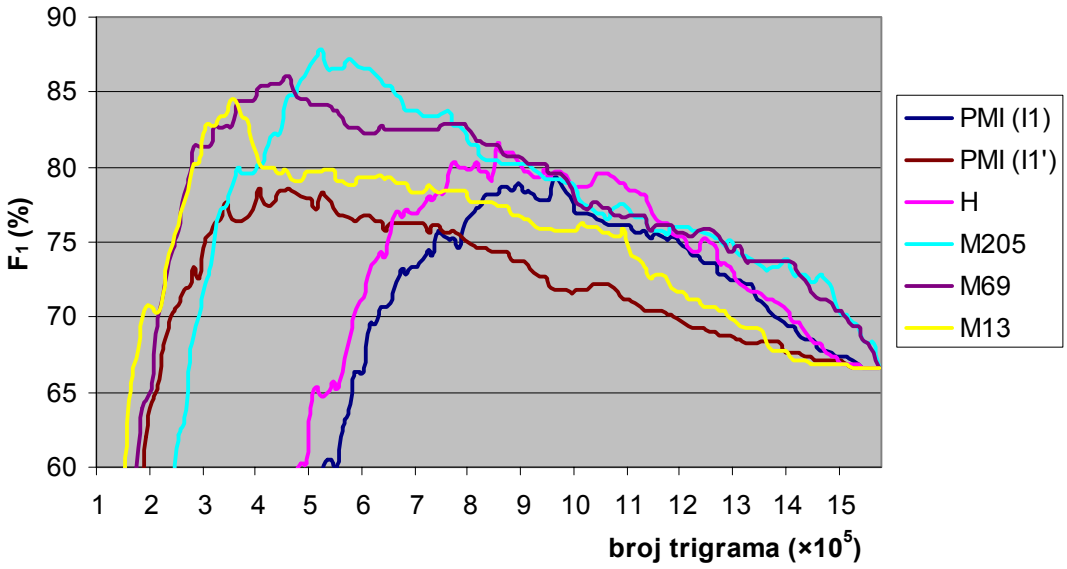


Slika 16: usporedba mjera M_{13} i M_{13B} (detalj)

Prikazani su minimumi krivulja na oba uzorka. Vidi se da je originalna mjera nešto bolja. To je možda zbog zanemarivanja pribrojnika 1 u donjem izrazu, zbog izmjene koeficijenta 0.423 u 0.5, ili jednostavno statistička anomalija. Slijedi usporedba s ostalim važnijim mjerama.



Slika 17: usporedba M_{13} s ostalim mjerama



Slika 18: usporedba M_{13} s ostalim mjerama (detalj)

Prikazane su minimalne krivulje mjera na oba uzorka. Mjera M_{13} zauzima treće mjesto.

Još jednom se naglašava da se na temelju navedenih rezultata ne može ništa zaključiti o općenitoj kvaliteti mjera, rezultati pokazuju samo *aprosksimaciju*

performansi mjera na jednoj kolekciji dokumenata, koristeći navedene uzorke klasificiranih *n*-grama.

5.5. Ostala zapažanja

Jedan od ciljeva ovog rada je utvrditi koliko su kvalitetne najčešće korištene mjere za ekstrakciju kolokacija. Mnoge od mjera pronađenih ovim eksperimentima pokazuju bolje rezultate (na korištenoj kolekciji dokumenata, uz navedene uzorke za procjenu F_1), što ukazuje na moguća poboljšanja, te poziva na daljnja istraživanja. Petrović et al. [25] tvrde da je za različite vrste kolokacija potrebno koristiti različite mjere, te navode primjer - heuristiku koja koristi jedan izraz za trigrame kojima je srednja riječ stop riječ (najčešće veznik), te drugi izraz za sve ostale trigrame. Takvo ponašanje u jedinkama modelira se korištenjem AKO operatora sa uvjetom *vr(b) = {X}*. Od svih mjera dobivenih eksperimentima s vrijednošću F_1 od preko 82%, *samo četiri mjere ne sadrže* AKO(*vr(b)={X}*, ?, ?) podstablo! Sve četiri mjere nisu imale poznate mjere uključene u početnoj populaciji. Ukupno 116 mjera s F_1 većim od 82% nije imalo poznate mjere u početnoj populaciji, što znači da ih je čak 112 (preko 96%) potpuno samostalno (bez kvalitetnog genetskog materijala u početnoj populaciji) došlo do istog zaključka - trigrami sa stop riječi u sredini trebali bi koristiti različite mjere. Ovakvi rezultati podupiru tvrdnju iznesenu u [25].

6. Zaključak

Cilj ovog diplomskog rada bio je istražiti područje evolucijskih algoritama i evolucijskog računanja općenito, te utvrditi primjenjivost tih metoda na postupke ekstrakcije kolokacija. U tu svrhu izgrađen je sustav koji standardnim metodama genetičkog programiranja optimira skup mjera za ekstrakciju kolokacija. Rezultati su vrlo dobri – pronađen je niz mjera za koje se procjenjuje da će dati vrlo dobre rezultate na korištenoj kolekciji dokumenata. Većinu dobivenih mjera nemoguće je jednostavno interpretirati u terminima statističke obrade teksta, ali ih je moguće implementirati kao „crne kutije“ koje postižu zadovoljavajuće rezultate. Uz to, malen broj kvalitetnih mjera vrlo je jednostavne strukture, moguće ih je analizirati i navesti razloge zbog kojih se postižu dobri rezultati. To je posao za stručnjake iz područja dubinske analize teksta. Autor se nada da će im rezultati ovog rada dati nove ideje za daljnja istraživanja. Konačno, mnoge od dobivenih uspješnih mjera vrlo su slične raznim mjerama predloženim u mnogim znanstvenim radovima, čime se dodatno potvrđuju pretpostavke iznesene u tim radovima na kojima se temelji njihova uspješnost.

U daljnjim istraživanjima valja usavršavati razvijeni sustav, te iskušati ostale postupke evolucijskog računanja. Potrebno je eksperimente ponoviti na mnogim drugim kolekcijama dokumenata, proširiti pretragu na mjere za digrame, tetragrame i veće *n*-grame, te napraviti statističku analizu svih rezultata kako bi se objektivno utvrdila njihova kvaliteta i moguća primjenjivost u sofisticiranim alatima za dubinsku analizu teksta.

Literatura

[1] Atkinson-Abutridy, J.A., Mellish, C., Aitken, J.S.: *Combining Information Extraction with Genetic Algorithms for Text Mining*, Časopis „IEEE Intelligent Systems“, broj 19(3), str. 22-30, 2004.

[2] Bergstrom, A., Jaksetic, P., Nordin, P.: *Enhancing information retrieval by automatic acquisition of textual relations using genetic programming*, IUI 2000, ACM Press, 2000.

[3] Boulis C.: *Clustering of Cepstrum Coefficients Using Pairwise Mutual Information,* Tehnical Report EE516, University of Washington, 2002.

[4] Cavaretta, M. J., Chellapilla, K.: *Data mining using genetic programming: The implications of parsimony on generalization error*, Proceedings of the Congress on Evolutionary Computation, broj 2, str. 1330-1337, IEEE Press., 1999.

[5] Church, K., Hanks, P.: *Word Association Norms, Mutual Information, and Lexicography*, Časopis „Computational Linguistics“, broj 16(1), str. 22-29,1990.

[6] Cummins, R., O'Riordan, C.: *Evolving, analysing and improving global term-weighting schemes in information retrieval*, Technical Report NUIG-IT-071204, National University of Ireland, Galway, Ireland, 2004.

[7] Evert, S., Krenn, B.: *Using small random samples for the manual evaluation of statistica evaluation measures,* Časopis „Computer speech and language“, broj 19, str. 450-466, 2005.

[8] Horng, J., Yeh, C.: *Applying Genetic Algorithms to Query Optimization in Document Retrieval*, Časopis „Information Processing & Management“, broj 36(1), str. 737-759, 2000.

[9] Hrvatski nacionalni korpus

http://www.hnk.ffzg.hr [01/16/2005]

[10] Goldman, J.P., Wehrli E.: *FipsCo: A syntax-based system for terminology extraction*, Grammar and Natural Language Processing Conference, Universite du Quebec at Montreal, 2001.

[11] Gordon, M., Fan, W., Pathak, P.: *Adaptive Web Search: Evolving a Program That Finds Information*, Časopis „IEEE Intelligent Systems“, broj 21(5), str. 72-77, 2006.

[12] Kolar, M., Vukmirović, I., Dalbelo Bašić, B., Šnajder, J.: *Computer-aided document indexing system*, Časopis „Journal of Computing and Information Technology“, broj 13(4), str. 299-305, 2005.

[13] Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* MIT Press, 1992.

[14] Koza, J.R.: *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

[15] Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A.: *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann, 1999.

[16] Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Kluwer Academic Publishers, 2003.

[17] Langdon,W. B., Poli, R.: *Genetic programming bloat with dynamic fitness*, Proceedings of the First European Workshop on Genetic Programming, volume 1391 of LNCS, str. 96-112, Springer-Verlag, 1998.

[18] Liu, H., Dougherty, E. R., Dy, J. G., Torkkola, K., Tuv, E., Peng, H., Ding, C., Long, F., Berens, M., Parsons, L., Zhao, Z., Yu, L., Forman, G:. *Evolving Feature Selection*, Časopis „IEEE Intelligent Systems“, broj 20(6), str. 64-76, 2005.

[19] Luke, S. Panait, L.: *Lexicographic Parsimony Pressure*, Proceedings of the Genetic and Evolutionary Computation Conference (2002), Morgan Kaufmann Publishers, 2002.

[20] Luke, S., Panait, L.: *A comparison of bloat control methods for genetic programming*, Časopis „Evolutionary Computation“, broj 14(3), str. 309-344, 2006.

[21] Manning, C., Schütze, H.: *Foundations of statistical natural language processing*, MIT Press, 1999.

[22] McInnes, B. T.: *Extending the loglikelihood measure to improve collocation identification*, Master thesis, University of Minnesota, 2004.

[23] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, 2. izdanje, Springer-Verlag Berlin / Heidelberg, 1994.

[24] Oakes, M. P.: *Statistics for corpus linguistics*, Edinbourgh University Press, 1998.

[25] Petrović, S., Šnajder, J., Dalbelo Bašić, B., Kolar, M.: *Comparison of Collocation Extraction Measures for Document Indexing*, Proceedings of ITI 2006. 28th International Conference on Information Technology Interfaces, University of Zagreb, 2006.

[26] Smadja, F., McKeown, K.: *Automatically extracting and representing collocations for language generation*, Proceedings of the 28th Annual Meeting of the ACL, str. 252-259, 1990.

[27] Smith, M.G., Bull, L.: *Feature Construction and Selection Using Genetic Programming and a Genetic Algorithm,* Časopis „Genetic Programming and Evolvable Machines“, broj 6(3), str. 265-281, 2005.

[28] Soule, T., Foster, J. A.: *Effects of code growth and parsimony pressure on populations in genetic programming*, Časopis „Evolutionary Computation“, broj. 6(4), str. 293-309, 1998.

[29] Soule, T., Foster, J. A., Dickinson, J.: *Code growth in genetic programming*, Proceedings of the First Annual Conference, str. 215-223, Stanford University, MIT Press, 1996.

[30] Šnajder, J.: *Rule-based automatic acquisition of large-coverage morphological lexicons for information retrieval*, Tech. Report, MZOS 2003-082, ZEMRIS, FER, University of Zagreb, 2005.

[31] Tadić, M., Šojat, K.: *Finding multiword term candidates in Croatian*, Proceedings of IESL2003 Workshop, Borovets, Bulgaria, str. 102-107, 2003.

[32] Thanopoulos, A., Fakotakis, N., Kokkinakis, G.: *Comparative evaluation of collocation extraction metrics*, Proceedings of the LREC 2002 Conference, str. 609-613, 2002.

[33] Vechtomova, O.: *Query expansion with long-span collocates,* Časopis „Information Retrieval“, broj 6(2), str. 251-273, 2003.

[34] Wu, C-C, Chang, J. S.: *Bilingual collocation extraction based on syntactic and statistical analyses*, Časopis „Computational Linguistics and Chinese Language Processing“, broj 9(1), str. 1-20, 2004.

[35] Zhang, B.T., Mühlenbein, H.: *Balancing accuracy and parsimony in genetic programming*, Časopis „Evolutionary Computation“, broj 3(1), str. 17-38, 1995.

Dodatak A – Detalji programske implementacije

Implementacija algoritma genetičkog programiranja izvedena je u programskom jeziku C++, a razvijena je korištenjem razvojnog sučelja Microsoft Development Environment 2003. Prevođenjem nastaje win32 izvršna datoteka za operativni sustav Microsoft Windows. Za stvaranje Linux izvršne datoteke uspješno je iskorišten alat *gpp*, dakle napisan izvorni kod je pisan koristeći standardne elemente jezika C++, te je prenosiv na druge platforme. Generirani kod je brz, pa se rezultati mogu dobiti već nakon nekoliko sati, iako je za veće količine kvalitetnih rezultata potrebno minimalno nekoliko dana.

Uz tekst ovog rada priložen je CD na kojem se nalazi komentirani izvorni kod s uputama za prevođenje, te već pripremljene izvršne datoteke sa svim potrebnim ulaznim podacima i uputama za pokretanje, te prikupljanje i interpretiranje ispisa rezultata.

Dodatak B – Popis najboljih jedinki

Na priloženom CD-u nalaze se arhive svih rezultatata generiranih sustavom opisanim u ovom diplomskom radu, kao i sve što je potrebno za pripremu novih eksperimenata. U nastavku je dan detaljan opis nekih mjera koje se navode u diskusiji rezultata ovog rada, te parametara pretrage koji su se koristili pri nalaženju tih mjera.

Mjere $M_{256,}$ M_{205} i M_{69} nasatale su evolucijom iz lošije mjere sa 238 čvorova. Parametri pretrage koji do doveli do te mjere bili su:

Tablica 4: parametri pretrage najboljeg rezultata

veličina populacije	300
broj iteracija prije odustajanja	100 000
vjerojatnost mutacije	0.1 %
poznate jedinke u početnoj populaciji	DA
najveća dozvoljena duljina	100
faktor parsimonije	0.1%

Postupak pretrage trajao je 327 sekundi, a ukupan broj izvedenih operacija križanja bio je 130366. Optimalni prag mjere je reda veličine 50 tisuća.

Zapis mjera dan je u postfiks notaciji, konstante se uvijek navode u zagradama. Operator AKO navodi se kao binarni, pri čemu se prvi argument (uvjet) navodi kao parametar (u zagradama).

Mjera M_{256}:

```
f(abc) f(a) f(c) * / f(abc) f(ab) f(c) - f(c) f(bc) f(b) - f(abc) +
/ + / N * f(b) + * ln f(c) f(b) * * N f(a) * f(abc) f(a) f(abc) f(a)
f(c) * / f(bc) * f(bc) f(b) + / f(a) N AKO(vr(b)={X}) * (-14.426000)
f(b) + / N * f(abc) f(abc) f(a) f(a) f(abc) f(a) f(c) * / f(abc)
f(ab) f(c) - f(c) f(bc) f(b) - f(abc) + / + / N * f(b) + *
AKO(vr(b)={X}) (-14.426000) f(b) + * / N * / N * ln N * / f(a) f(b)
+ * ln ln (2.000000) * ln ln / f(abc) f(a) f(c) * / f(b) * ln ln
(2.000000) * ln ln / N * ln * / f(bc) * f(bc) f(b) + / N * (-
14.426000) f(b) + / N * f(abc) N f(a) * f(a) f(abc) f(a) f(c) * /
f(bc) * f(abc) f(b) + / N * (-14.426000) f(b) + / N * f(b) f(c) * ln
ln / f(abc) f(a) f(c) * / f(c) * ln ln (2.000000) * ln ln / N * / N
* / N * ln f(c) * / f(a) f(b) + * ln ln f(abc) f(abc) f(a) f(a) N
```

```
AKO(vr(b)={X}) (-14.426000) f(b) + * / N * / N * ln f(c) * / f(a)
f(b) + * ln ln * ln ln / f(abc) f(a) f(c) * / f(a) f(b) + * ln ln
(2.000000) * ln ln / N * ln ln AKO(vr(c)={X}) N * AKO(vr(b)={X})
```

Mjera M_{205}:

```
f(abc) f(a) f(c) * / f(abc) f(ab) f(c) - f(c) f(bc) f(b) -f(abc) + /
+ / N * f(b) + * ln f(c) f(b) * * N f(a) * f(abc) f(a) f(abc) f(a)
f(c) * / f(bc) * f(bc) f(b) + / f(a) N AKO(vr(b)={X}) * (-14.426000)
f(b) + / N * f(bc) f(b) -(2.000000) * ln ln / f(a) f(c) * (2.000000)
* ln ln / N * ln * / f(bc) * f(bc) f(b) + / N * (-14.426000) f(b) +
/ N * f(abc) N f(a) * f(a) f(abc) f(a) f(c) * / f(bc) * f(abc) f(b)
+ / N * (-14.426000) f(b) + / N * f(b) f(c) * ln ln / f(abc) f(a)
f(c) * / f(c) * ln ln (2.000000) * ln ln / N * / N * / N * ln f(c) *
/ f(a) f(b) + * ln ln f(abc) f(abc) f(a) f(a) N AKO(vr(b)={X}) (-
14.426000) f(b) + * / N * / N * ln f(c) * / f(a) f(b) + * ln ln * ln
ln / f(abc) f(a) f(c) * / f(a) f(b) + * ln ln (2.000000) * ln ln / N
* ln ln AKO(vr(c)={X}) N * AKO(vr(b)={X})
```

Mjera M_{69}:

```
f(abc) f(a) f(c) * / f(abc) f(ab) f(c) - f(c) f(a) / + / N * * ln
f(c) f(b) * * f(a) f(c) * f(abc) f(abc) f(ab) f(c) - f(c) f(a) / + /
f(c) * N + / f(bc) * (-14.426000) f(b) + / N N * * f(b) / N * f(c)
ln / f(a) (2.000000) * ln ln / ln ln AKO(vr(c)={X}) N *
AKO(vr(b)={X})
```

U nastavku slijedi popis parametara pretrage kojom je pronađena mjera M_{13}.

Tablica 5: parametri pretrage za mjeru M_{13}

veličina populacije	200
broj iteracija prije odustajanja	100 000
vjerojatnost mutacije	1 %
poznate jedinke u početnoj populaciji	NE
najveća dozvoljena duljina	100
faktor parsimonije	0.5%

Pretraga je trajala 83 sekunde, ukupan broj operacija križanja bio je 138376.

Zapis mjere M_{13}:

```
(-0.423000) f(c) * f(abc) / f(a) * f(abc) f(b) - AKO(vr(b)={X})
f(abc) /
```