SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1046

ALGORITAM ZA PREPOZNAVANJE PROTEINSKIH INTERAKCIJA U BIOMEDICINSKIM ČLANCIMA

Ivan Klarić

Zagreb, rujan 2006.

Hvala:
mami, tati, sestri – na podršci i trpljenju mojih većih i manjih mušica svih ovih godina
mr. sc. Mile Šikić – na vodstvu i strpljivosti kada sam kasnio sa rokovima
cijeloj ekipi sa LSS-a – radi dobrog društva dok je ovaj rad nastajao
svim prijateljima – što su to što jesu

Sadržaj

Popis slika

Popis tablica

1. Uvod

Svjedoci smo snažnog tehnološkog razvoja u zadnjih nekoliko desetljeća. Pri tom je naročito u zadnje vrijeme očigledan znatan razvoj biotehnoloških grana. Velik je trud istraživača u cijelom svijetu uložen u proučavanje proteinskih interakcija koje su osnova svih fizioloških procesa u svim poznatim organizmima. Kao rezultat tih napora, objavljene su velike količine znanstvenih članaka koji se bave proučavanjem i prepoznavanjem proteinskih interakcija u raznim organizmima. Javlja se problem sažetog i unificiranog predstavljanja znanja o proteinskim interakcijama na jednom mjestu. Zbog toga su nastale mnoge baze poznatih proteinskih interakcija [1],[2]. Problem kod većine postojećih baza podataka je da je samo dio podataka eksperimentalno potvrđen, dok je ostatak dobiven raznim metodama predviđanja. Stoga se eksperimentalno potvrđeni i objavljeni podaci najčešće dobivaju ručnim ili automatskim izdvajanjem utvrđenih interakcija iz znanstvenih članaka. Tako dobiveni podaci olakšavaju i ubrzavaju posao znanstvenicima koji održavaju i/ili razvijaju baze proteinskih interakcija. Iako već postoji nekoliko radova koji se bave sličnom problematikom, u ovom radu je predstavljen novi sustav koji omogućava automatiziranu ekstrakciju znanja iz sažetaka javno dostupnih biomedicinskih tekstova na engleskom jeziku vezanih za proteinske interakcije. Osim toga, stvorena je baza podataka tih interakcija i uspoređeni su dobiveni rezultati s rezultatima koje su dobili drugi znanstvenici. Kao metode za izdvajanje proteinskih interakcija korištene su kontekstno neovisna gramatika i metoda vektora potpore (SVM, engl. Support Vector Machine).

Teoretska osnova metoda upotrebljenih u razvoju sustava je dana u poglavlju “2. Metode”, korpusi tekstova i baze korištene u razvoju sustava su opisani u poglavlju “3. Podaci”, dok je u poglavlju “4. Rješenje” dan pregled razvijenog sustava sa detaljnim opisima pojedinih komponenti. Postupak testiranja kao i rezultati testiranja sustava su dani u poglavlju “5. Rezultati”. Praktičan primjer korištenja sustava u izgradnji baze proteinskih interakcija je dan u poglavlju “6. Implementacija”. Diskusija rezultata postignutih izgrađenim sustavom je dana u poglavlju “7. Diskusija”. Konačno, sažetak rada i zaključak su dani u poglavljima “8. Sažetak” i “9. Zaključak”. Popis korištene literature je dan u poglavlju “10. Literatura”. Praktične upute za korištenje razvijenog sustava, kao i svi korišteni podaci tokom razvoja su dani u dodacima.

2. Metode

Prilikom razvoja sustava za prepoznavanje naziva proteina u tekstu je korištena metoda strojnog učenja poznata pod nazivom “Metoda vektore potpore”, dok je za prepoznavanje sintakse rečenica engleskog jezika korištena konteksno neovisna gramatika. Također, kako bi se omogućilo prepoznavanje svih morfoloških oblika ključnih riječi, korišten je Porterov algoritam za određivanje korijena riječi. Teoretske osnove ovih metoda su detaljnije opisane u ovom poglavlju.

2.1. Metoda vektora potpore (Support Vector Machine)

Metoda vektora potpore je jedna od metoda strojnog učenja. Jedan od osnovnih problema kojima se bavi teorija učenja jest razvrstavanje ulaznih objekata (uzoraka, primjera) u klase. U najjednostavnijem primjeru se radi o razvrstavanju ulaznih primjera iz nekog skupa X koji nazivamo domena u jednu klasu iz skupa Y. Drugim, riječima potrebno je razviti funkciju $f: X \to Y, f(x) = y, x \in X, y \in Y$ gdje je x ulazni uzorak iz skupa svih ulaznih uzoraka X (domena), a y je jedna od klasa iz skupa svih mogućih klasa Y. Ovakvu funkciju f nazivamo decizijska funkcija. U nastavku će biti objašnjene osnove statističke teorije učenja potrebne za shvaćanje i definiciju metode vektora potpore.

Postavimo problem formalno. Svaki primjer (uzorak) označavamo kao uređeni par (x_i, y_i) gdje x_i označava sam uzorak (primjer), a y_i klasu kojoj x_i pripada. Pri tome, radi jednostavnosti svedimo problem na pridruživanje jednoj od dvije klase (binarna klasifikacija), te označimo te klase sa -1 i +1. Odavde slijedi da je za potrebe ove ilustracije $y_i \in \{-1, +1\}$. Algoritam radi u dvije faze. U prvoj fazi, na tzv. trening skupu (skup uređenih parova (x_i, y_i), tj. skup primjera sa pridruženim klasama) “uči”, a nakon što je učenje završeno, sustav je spreman dotad neviđenim primjerima pridruživati klasu.

U ovom trenutku definiramo funkciju $k: X \times X \to R$ gdje je R skup realnih brojeva takvu da vrijedi $k(x, x') = k(x', x), \forall x, x' \in X$. Ovo svojstvo nazivamo još svojstvom simetričnosti, tj. za funkciju k kažemo da je simetrična. Funkcija k vraća kao rezultat mjeru sličnosti ulaza x i x' i naziva se jezgra (kernel). Najjednostavniji primjer mjere sličnosti dva vektora je skalarni produkt, te skalarni produkt možemo smatrati jednim primjerom jezgrene

funkcije ako su ulazi numerički vektori. Problem se, međutim javlja ako su ulazi u nekom prostoru u kojem nije moguće definirati neku jezgru poput skalarnog produkta. Tada se koristi transformacija ulaznog prostora X u $\phi: X \to H, \phi(x) = h, x \in X, h \in H$ u novi prostor značajki (engl. feature space) H.

Pretpostavljamo da su ulazi generirani po nekoj vjerojatnosti razdiobom P(x, y). To je osnovna pretpostavka u teoriji učenja, te se takvi ulazi često nazivaju iid (od engl. independent and identically distributed). Cilj učenja je pronaći funkciju f koja će točno klasificirati nove ulaze (x, y) tako da vrijedi f(x) = y. Ti su ulazi također generirani po razdiobi P(x, y). Točnost te klasifikacije mjerimo funkcijom $c(x, y, f(x)) := \frac{1}{2}|f(x) - y|$. Treba uočiti da je vrijednost funkcije 0 kada je f(x) = y, tj primjer je dobro klasificiran, a inače je vrijednost funkcije 1. Ako se na funkciju f ne postavljaju nikakva dodatna ograničenja, postoji mogućnost da odaberemo funkciju koja jako dobro radi na uzorcima za treniranje, zadovoljavajući $f(x_i) = y_i$ za sve i=1, ..., m, ali ne radi dobro na novim uzorcima. Općenito, važno je uočiti da za svaku funkciju f naučenu na setu uzoraka $(x_1, y_1), \ldots, (x_m, y_m) \in \{X \times \{\pm 1\}\}$ i za svaki novi set uzoraka $(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_{\bar{m}}, \bar{y}_{\bar{m}}) \in \{X \times \{\pm 1\}\}$, uz $\{\bar{x}_1, \ldots, x_m\} \cap \{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\} = \emptyset$ postoji funkcija f' takva da vrijedi $f'(x_i) = f(x_i), i = 1, \ldots, m$, ali $f'(\bar{x}_i) \neq f(\bar{x}_i), \forall i = 1, \ldots, \bar{m}$. Budući da se za vrijeme učenja raspolaže samo uzorcima iz trening seta, ne može se ustvrditi koja je od te dvije funkcije bolje. Taj se problem rješava uvođenjem prosječne greške uslijed treniranja (tzv. empirijski rizik): $R_{emp}[f] = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}|f(x_i) - y_i|$. Analogno, definiramo prosječnu grešku uslijed testiranja usrednjavanjem testnih uzoraka dobivenih distribucijom P(x, y): $R[f] = \int \frac{1}{2}|f(x) - y| dP(x, y)$.

Kao što je već navedeno, potrebno je ograničiti prostor funkcija iz kojih se bira klasifikacijska funkcija. Svaka funkcija razdvaja uzorke na određeni način, te pritom označava pojedine uzorke. Pošto u najjednostavnijem primjeru postoje samo dvije moguće oznake (+1 i -1), postoji najviše 2^m različitih načina na koje se može označiti m uzoraka. VC dimenzija se

definira kao najveći m takav da postoji skup od m točaka koje se mogu razdvojiti u klase, kažemo da je dimenzija ∞ ako takav m ne postoji.

U nekom prostoru H u kojem je definiran skalarni produkt <·,·> definiramo skup hiperravnina kao $\langle \vec{w}, \vec{x} \rangle + b = 0, \vec{w} \in H, b \in R$ što odgovara klasifikacijskoj funkciji $f(x) = sgn(\langle \vec{w}, \vec{x} \rangle + b)$. Za neki skup uzoraka kažemo da je linearno separabilan ako postoji hiperravnina kojom se skup uzoraka može podijeliti na dvije klase. Optimalnom hiperravninom nazivamo onu hiperravninu koja je određena maksimalnim razmakom između hiperravnine i bilo kojeg uzorka (uvjet maksimalne margine), tj. problem se može svesti na $max(min\{||\vec{x} - \vec{x}_i|| \vec{x} \in H, \langle \vec{w}, \vec{x} \rangle + b\})$. Važno je uočiti da se kapacitet skupa hiperravnina smanjuje kako povećavamo razmak između hiperravnine i uzoraka.

Opišimo sada algoritme za učenje koji određuju hiperravnine, a mogu biti realizirani u prostoru u kojem je definiran skalarni produkt. Da bi odredili optimalnu hiperravninu, moramo riješiti sljedeće:

$$minimize \ldots \tau(w) = \frac{1}{2}||w||^2, w \in H, b \in R \quad \text{uz} \quad y_i(\langle w, x_i \rangle + b) \geqslant 1, i = 1, \ldots, m \ .$$

Navedena nejednadžba osigurava da f(x_i) bude 1 za y_i = 1, te -1 za y_i = -1. Kada bi ||w|| bio jednak 1, lijeva strana nejednadžbe bi bila jednaka udaljenosti uzorka x_i od hiperravnine. Općenisto, potrebno je $y_i(\langle w, x_i \rangle + b)$ podijeliti sa ||w|| kako bi dobili tu udaljenost. Ako je nejednadžba zadovoljena za sve i = 1, ..., m sa minimalnom duljinom vektora w, tada je ukupna udaljenost uzoraka od hiperravnine maksimalna. Funkcija τ se naziva funkcija cilja (engl. objective function), a nejednadža se naziva uvjet nejednakosti (engl. inequality constraints). Problem optimiranja pod ovakvim uvjetima se naziva problem uvjetne optimizacije (engl. constrained optimization problem) i rješava se upotrebom Lagrangeovih koeficijenata $\alpha_i \geqslant 0$ i Langragiana:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle w, x_i \rangle + b) - 1)$$

Navedeni Langragian je potrebno minimizirati s obzirom na w i b i maksimizirati s obzirom na α_i. Parcijalnim deriviranjem Lagrangiana tražimo minimum:

$$\frac{\partial L(w,b,\vec{\alpha})}{\partial w} = w - \sum_{i=1}^{m} y_i \alpha_i x_i = 0$$
$$\frac{\partial L(w,b,\vec{\alpha})}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0$$
$$\rightarrow w = \sum_{i=1}^{m} y_i \alpha_i x_i$$
$$\rightarrow 0 = \sum_{i=1}^{m} y_i \alpha_i$$

Rezultantni vektor w se sastoji samo od podskupa iz skupa primjera za učenje, točnije samo od onih kod kojih je α_i različit od nule. Ti se uzorci nazivaju Support Vektori i nose sve informacije potrebne za rješavanje problema. Ovo nas dovodi do osnovne ideje: hiperravnina je određena uzorcima koji su joj najbliži a ostali se zanemaruju. Uvrstimo li sada rješenja parcijalnih jednadžbi u početni Lagrangian, dobivamo:

$$L(w,b,\vec{\alpha}) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{m} \alpha_i [y_i((w \cdot x_i) + b) - 1]$$
$$... = \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) - \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{i=1}^{m} \alpha_i$$
$$... = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_x y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

Ovime su eliminirane varijable w i b i dobiven je sljedeći problem:

$$maximize ... W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle, \alpha \in R$$

uz $\alpha_i \geq 0, i = 1, ..., m$ i $\sum_{i=1}^{m} y_i \alpha_i = 0$.

Dobivena decizijska funkcija izgleda ovako:

$$f(x) = sgn(\sum_{i=1}^{m} y_i \alpha_i \langle x, x_i \rangle + b)$$

Sada smo definirali sve pojmove potrebne kako bi mogli opisati algoritam vektora potpore. Sve operacije do sada su definirane u prostoru H gdje je definiran skalarni produkt. Upotrebom jezgrene funkcije ulazni prostor X prebacujemo u prostor H na sljedeći način:

$k(x, x') = \langle x, x' \rangle$. Drugim riječima upotrebom jezgrene funkcije smo odredili skalarni produkt vektora x i x'. Ova zamjena je jedna od glavnih ideja algoritma, a zasniva se na Coverovom teoremu o separabilnosti uzoraka:

"Veća je vjerojatnost da nelinearno transformirani vektori u višedimenzionalnom prostoru budu linearno separabilni nego u originalnom nižedimenzionalnom prostoru."

Ovaj se svojevrsni trik (engl. kernel trick) može upotrijebiti upravo zato što se svi vektori iz prostora H pojavljuju u skalarnim produktima, pa nije potrebno prebacivati sve vektore u višedimenzionalan prostor, već se radi sa vektorim iz originalnog prostora, ali se za računanje skalarnog produkta koristi jezgrena funkcija. Ako primjenimo ovaj trik na netom definiranu decizijsku funkciju i odgovarajući problem maksimizacije, dobivamo decizijsku funkciju oblika:

$$f(x) = sgn(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b)$$

i pripadni dualni problem:

$$maximize \dots W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j k(x_i, x_j), \alpha \in R \quad \text{uz} \quad \alpha_i \geq 0, i = 1, \dots, m \quad \text{i}$$

$\sum_{i=1}^{m} y_i \alpha_i = 0$.

Detalje i više informacija o metodi vektora potpore je moguće vidjeti na [3].

2.2. Kontekstno neovisna gramatika

Sljedeća definicija formalne gramatike i preuzeta je iz [4]. Formalna gramatika se definira kao uređena četvorka G = (V, T, P, S) gdje je:

- V skup nezavršnih znakova na koje se primjenjuju transformacije (produkcije), tako da vrijedi

  $T \cap V = \emptyset$, tj. T i V su disjunktni skupovi.

- T skup završnih znakova na koje se daljnje transformacije ne mogu primjenjivati.
- P je skup produkcija, tj. skup transformacijskih pravila. Produkcija se sastoji od lijevog nezavršnog znaka, te skupa završnih i nezavršnih znakova sa desne strane

produkcije. Pri tome se se takva produkcija interpretira tako da se svako pojavljivanje lijevog nezavršnog znaka u nizu koji se generira zamijenjuje nizom završnih i nezavršnih znakova sa desne strane produkcije.

- S je početni nezavršni znak od kojeg započinje generiranje niza znakova.

Dodatno, definira se prazna, ε-produkcija. Na primjer, produkcija oblika $A \rightarrow \varepsilon$ znači da se nezavršni znak A može zamijeniti praznim nizom tj. može se obrisati iz međuniza.

Korištenjem formalne gramatike se kreće od početnog nezavršnog znaka i uzastopnom primjenom produkcija se generiraju međunizovi. Međunizom se smatra svaki niz znakova koji sadrži završne i nezavršne znakove. Konačan niz sadrži samo završne znakove, te se produkcije više ne mogu primjeniti na taj niz. Skup svih nizova završnih znakova koji se mogu generirati upotrebom gramatike G nazivamo jezikom gramatike G i pišemo L(G). Ukoliko neke dvije gramatike G_1 i G_2 generiraju istovjetne jezike $L_1(G_1)$ i $L_2(G_2)$, kažemo da su te dvije gramatike istovjetne.

Kontekstno neovisna gramatika generira nizove završnih znakova za koje kažemo da pripadaju klasi kontekstno neovisnih jezika. Neka gramatika je kontekstno neovisna ako sa lijeve strane svih produkcija te gramatike stoji točno jedan nezavršni znak gramatike.

Osim generiranja konačnog niza završnih znakova, postavlja se i suprotan problem: pripada li neki niz završnih znakova skupu svih nizova završnih znakova koje generira gramtika G. Za potrebe rješavanja tog problema je razvijen poseban automat, tzv. parser, a postupak određivanja pripada li neki niz završnih znakova skupu svih nizova završnih znakova koje generira gramatika G naziva se parsiranje. Postupke parsiranja dijelimo u dvije velike skupine:

- Parsiranje od vrha prema dnu (engl. top-down parsing). Postupak kreće od početnog nezavršnog znaka i uzastopnim izborima produkcija generira međunizove znakova dok generirani niz završnih znakova ne odgovara nizu koji se parsira ili parser utvrđuje da zadani niz ne pripada skupu svih nizova završnih znakova koji se generiraju gramatikom G.
- Parsiranje od dna prema vrhu (engl. bottom-up parsing). Postupak kreće od zadanog niza završnih znakova i primjenjuje produkcije “u nazad” dok ne dođe do početnog

nezavršnog znaka S. Ukoliko ne uspije doći do početnog nezavršnog znaka S, zadani niz završnih znakova ne pripada skupu svih nizova završnih znakova koji se generiraju gramatikom G.

U obje klase parsera postoji razvijeno puno parsera, te specijaliziranih gramatika. Detaljnija lista parsera, njihov princip rada i podklase kontekstno neovisnih gramatika koje parsiraju je dana u [5]. Za potrebe ovog rada biti će korištena metoda parsiranja algoritmom rekurzivnog spusta. Riječ je o klasičnoj metodi parsiranja od vrha prema dnu.

2.3. Porterov algoritam za određivanje korijena riječi

Porterov algoritam za određivanje korijena riječi engleskog jezika je prvi puta opisan u [6] te je ovdje prenesen iz tog izvora. Algoritam je originalno nazvan “Algoritam za micanje nastavaka riječi”, tj algoritam ne određuje morfološki korijen riječi, već samo svodi sve oblike riječi na zajednički korijen. Ovo je nužno kako bi se prepoznali svi oblici riječi, kao npr. CONNECT, CONNECTED, CONNECTING, CONNECTION, CONNECTIONS. Algoritam se primjenjuje u mnogo primjena poput pretraživanja, indeksiranja dokumenata, text-miningu i slično. Pri tom je važno napomenuti da je algoritam razvijen za engleski jezik, te je za engleski jedino i primjenjiv. Ovdje će algoritam biti opisan u kratkim crtama, za detalje je najbolje proučiti [6].

Suglasnikom u riječi smatramo sva slova različita od A, E, I, O, U ili Y kada je prije njega suglasnik. Označimo suglasnik sa znakom c (od engl. consonant), a samoglasnik znakom v (od engl. vowel). Niz uzastopnih suglasnika u riječi duljine veće od nula (npr, ccccc) označavamo znakom C, a niz uzastopnih samoglasnika duljine veće od nula (npr, vvvvvvv) u riječi označavamo znakom V. Sada se svaka riječ, ili dio riječi može prikazati na jedan od sljedeća četiri načina:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

Ova četiri oblika se skraćeno mogu zapisati na sljedeći način:

[C] VCVC ... [V]

gdje uglate zagrade označavaju da je taj niz znakova opcionalan. Također, ponavljanje kombinacije “VC” m puta možemo označiti sa (VC){m}, pa je sada konačan skraćeni oblik prikaza svih riječi:

[C](VC){m}[V]

Neka se m zove mjere neke riječi ili njenog dijela kada je riječ prikazana na ovaj način. Kada je m=0 kažemo da je to NULL riječ.

Pravila za micanje nastavaka riječi su dana u sljedećem obliku:

(uvjet) S1 -> S2

i interpretiraju se na sljedeći način: ako riječ završava nastavkom S1, a ostatak riječi prije S1 zadovoljava uvjet dan u zagradi, S1 će se zamijeniti sa S2. Uvjet je najčešće zadan u ovisnosti o mjeri riječi. Primjerice,

(m > 1) EMENT ->

znači da ako ostatak riječi prije nastavka EMENT ima mjeru veću od 1, onda se nastavak EMENT zamjenjuje ni sa čime, tj nastavak EMENT se briše. Ako ovo pravilo primjenimo na riječ REPLACEMENT, nastavak EMENT će biti izbrisan jer je ostatak riječi prije -EMENT REPLAC, koji ima mjeru m=2, pa će se pravilo izvršiti i riječ REPLACEMENT će biti zamijenjena riječi REPLAC.

Dodatno, uvjet još može sadržavati i sljedeće elemente:

*S – označava da ostatak riječi završava sa slovom S (isto je moguće definirati i za ostala slova)

v - označava da ostatak riječi sadrži samoglasnik

*d – označava da ostatak riječi završava sa dvostrukim suglasnikom (npr, -TT ili -SS)

*o – označava da ostatak završava sa cvc kombinacijom gdje drugi c nije W, X ili Y

Također, uvjet može sadržavati i logičke operator AND, OR i NOT. U nizu pravila napisanih jedno ispod drugoga, samo jedno se ispunjava, a to će biti ono koje ima najdulji S1 koji odgovara slučaju. Sada kada su definirana pravila i način na koji se primjenjuju, može se

definirati i višekoračni algoritam kroz ta pravila. Puna lista pravila i redoslijed kojim se izvode je dan u [6], te ovdje radi opširnosti neće biti navođena.

3. Podaci

U ovom su poglavlju opisani postojeći ulazni podaci koji su korišteni prilikom razvoja i testiranja sustava, kao i baze koje se koriste u radu sustava.

Prilikom razvoja sustava korištena su dva izvora znanja:

- ručno označeni korpusi biomedicinskih tekstova za razvoj sustava
- postojeće baze proteina

Korpusi biomedicinskih tekstova su korišteni prilikom razvoja i testiranja sustava, dok su se postojeće baze proteina koristile kao svojevrsni riječnici prilikom ekstrakcije naziva proteina iz teksta. GENIA korpus i Yapex korpus su korpusi označenih biomedicinskih tekstova korišteni u svrhe učenja i testiranja ekstrakcije naziva proteina iz tekstova, dok je PICorpus korpus korišten za svrhu razvoja i testiranja sustava za prepoznavanje proteinskih interakcija u tekstu.

3.1. Korpusi biomedicinskih tekstova

3.1.1. GENIA korpus

GENIA korpus je razvijen na Department of Information Science, Faculty of Science, University of Tokyo. Riječ je o ručno označenom korpusu koji sadrži 2000 sažetaka biomedicinskih tekstova izdvojenih iz MEDLINE baze, National Library of Medicine. U korpusu su označene tvari i njihovi podskupovi te lokacije vezane za proteinske interakcije iz biološke domene u XML formatu podataka. Sažeci su odabrani pretragom sa ključnim riječima “Human, Blood Cells, Transcription Factors”.

Također, na istom je sveučilištu u sklopu istog projekta razvijen označivač uloga pojedinih riječi u rečenici (engl. Part-Of-Speech tagger) specijaliziran za biomedicinske tekstove nazvan GeniaTagger. Više detalja o tome kako se GeniaTagger koristi unutar ovog sustava je dano u četvrtom poglavlju.

3.1.2. Yapex korpus

Yapex korpus biomedicinskih tekstova je nastao prilikom razvoja programa Yapex za

ekstrakciju naziva proteina iz teksta. Yapex je program za prepoznavanje naziva proteina u tekstu temeljen na ručno izgrađenim pravilima. Ovaj je korpus zanimljiv radi usporedbe rezultata razvijenog prepoznavanja naziva proteina u tekstu sa Yapex sustavom. Program Yapex je razvijan na uzorku za učenje koji sadrži 99 sažetaka, a testiran na uzorku od 48 sažetaka.

3.1.3. PICorpus

PICorpus (engl. Protein Interaction Corpus) je korpus ručno označenih rečenica izdvojenih iz biomedicinskih članaka koji govore o proteinskim interakcijama. Korpus se sastoji od 363 označene rečenice u XML formatu. Problem sa ovim korpusom je u tome što ne postoji jedinstvena datoteka rješenja u kojoj piše eksplicitno koje se interakcije navode u kojoj rečenici, već su samo pojedini elementi u rečenicama (poput naziva proteina i ključnih riječi koje označavaju interakciju) označeni, te je stoga potrebno ručno provjeravati rezultate. Također, ne govore sve rečenice u korpusu o proteinskim interakcijama, već neke govore o interakcijama proteina sa drugim molekulama, kao i o nekim ponašanjima pojedinih proteina u određenim kemijskim uvjetima (poput uvjeta niske PH vrijednosti u okolini i slično).

3.2. Baze proteina

Kao jedna od značajki korištenih prilikom ekstrakcije naziva proteina iz teksta korišteni su upiti u UniProtKB/Swiss-Prot bazu proteina. UniProtKB/Swiss-Prot baza proteina je ručno obrađena baza proteinskih sekvenci uspostavljena 1986. godine, a od 2003. godine je održavana od strane UniProt konzorcija koji je nastao kao rezultat suradnje Swiss Institute of Bioinformatics (SIB), odjela za bioinformatiku i strukturalnu biologiju sa ženevskog sveučilišta, European Bioinformatics Institute (EIB) i Georgetown University Medical Center's Protein Information Resource (PIR). Baza je s vremenom postala de-facto standard za navođenje proteina, te se jedinstvena swissprot oznaka (SwissProtID) sve ćešće koristi kod referenciranja proteina. Baza je primarno namijenjena dohvatu proteinskih sekvenci, ali se osim tih informacija u njoj nalaze i nazivi proteina, alternativna imena proteina, kao i kraći tekstualni opisi proteina.

3.3. PUBMED baza biomedicinskih članaka

PubMed Central (PMC) je središnja besplatna digitalna arhiva biomedicinske literature. Održava ju Nacionalni institut za zdravlje (NIH – National Institutes of Health) Sjedinjenih američkih država. U sklopu Nacionalnog instituta za zdravlje djeluje Nacionalni centar za biotehnološke informacije (NCBI – National Center for Biotechnology Information) koji je sastavni dio Nacionalne knjižnice medicine (NLM – National library of Medicine).

Glavna zadaća knjižnice je očuvanje i održavanje javne dostupnosti digitalne arhive stručnih publikacija, kao što to čini i sa tiskanim publikacijama. Sama knjižnica ne objavljuje nikakve publikacije, već se isključivo bavi arhiviranjem istih.

Svaki objavljeni članak po ulasku u arhivu dobiva jedinstveni PubMed identifikator (PubMedID) na temelju kojega ga je moguće na jedinstven način pronaći i prepoznati. Dodatno se uz članak još čuvaju i svi podaci o autorima, izdavaču, izdanju časopisa u kojem je objavljen i slično. Više dodatnih podataka o PubMed bazi je dostupno na [7], a pretraživanje same baze je moguće sa [8].

4. Rješenje

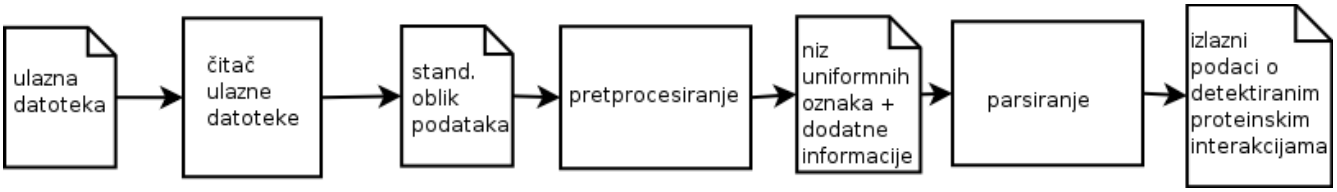
U ovom će poglavlju biti opisan cjelokupni sustav koji rješava zadani zadatak upotrebom metoda i podataka opisanih u prethodna dva poglavlja. U poglavlju “4.1. Implementirani sustav” je dan cjelovit pregled sustava na visokoj razini, dok su detalji o pojedinim podsustavima opisani u poglavljima “4.2. Pretprocesiranje” i “4.3. Sintaksni analizator”.

4.1. Implementirani sustav

4.1.1. Pregled implementiranog sustava

Pregled glavnih komponenti sustava je dan slikom 1 i sastoji se od sljedećih glavnih komponenti:

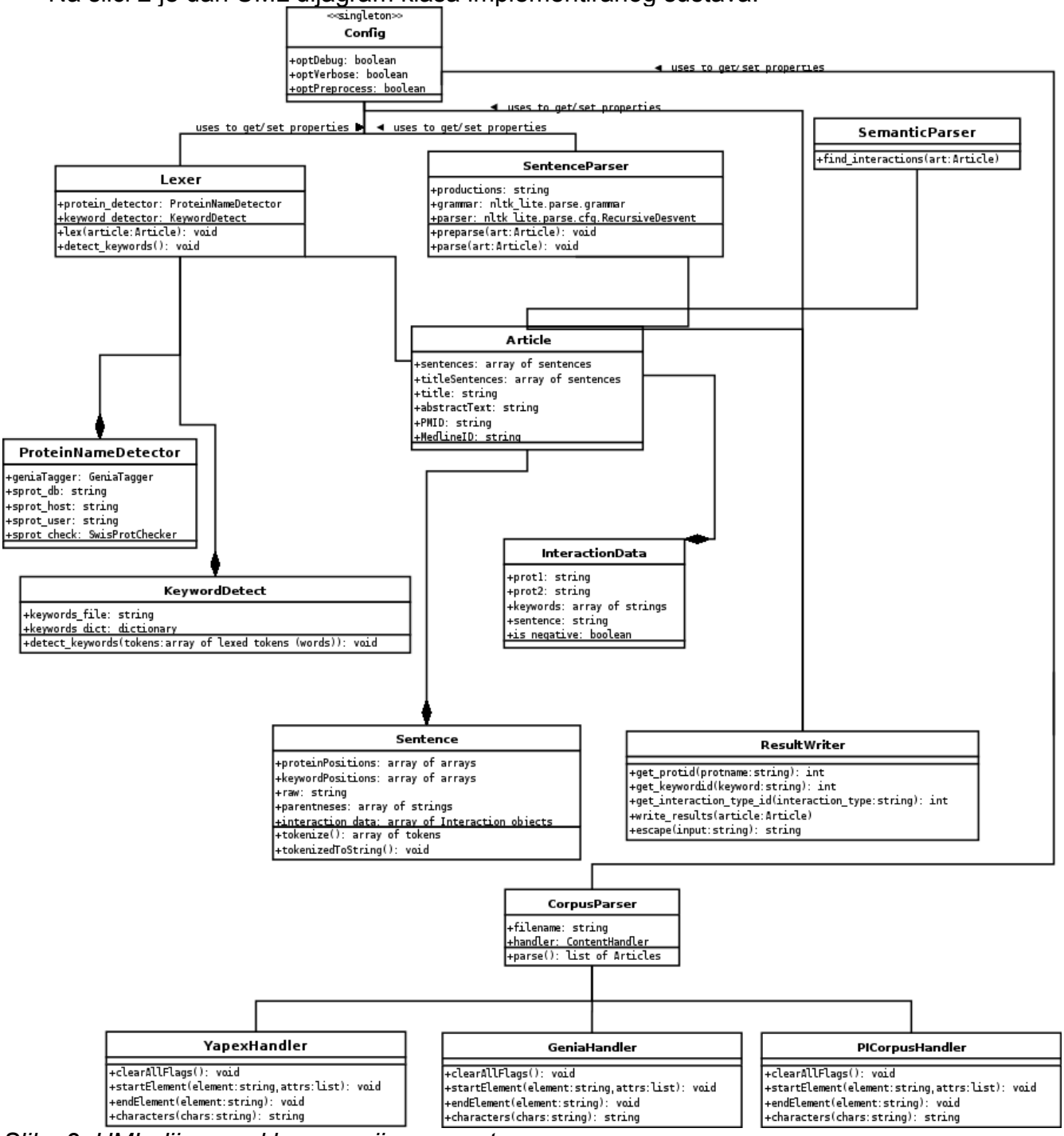
- čitač ulazne datoteke
  - ulazna je datoteka u XML obliku podataka, međutim budući da se koristi više različitih izvora ulaznih podataka, a svaki definira svoju vrstu XML oznaka, potrebno je implementirati zaseban čitač za svaku vrstu podržanih ulaznih podataka
- pretprocesiranje
  - u ovoj se komponenti sustava ulazni podaci razlažu na riječi, prepoznaju se nazivi proteina i ključne riječi među ulaznim podacima i generira se na izlazu niz uniformnih oznaka koje označavaju vrstu ulazne riječi, te dodatne podatke o riječi
- parsiranje
  - parser parsira uniformni niz znakova, te na temelju definiranih pravila prepoznaje proteinske interakcije u rečenicama



Slika 1: Pregled glavnih komponenti sustava

4.2. Objektni model implementiranog sustava

Na slici 2 je dan UML dijagram klasa implementiranog sustava:



Slika 2: UML dijagram klasa razvijenog sustava

Klasa CorpusParser upotrebom modula xml.sax parsira ulazni XML dokument, te upotrebom specifičnih tzv. “handler” klasa parsira ulaznu datoteku. U klasi Lexer je implementirano svo pretprocesiranje: prepoznavanje naziva proteina, prepoznavanje ključnih riječi, te računanje svih potrebnih značajki korištenih u toj i sljedećim fazama. Pretprocesiranje se ostvaruje pozivanjem metode lex() na objektu klase Lexer. Nakon poziva metode lex() stvoren je niz uniformnih jedinki (tokena) koji opisuje ulaznu rečenicu razloženu na pojedine riječi zajedno sa izračunatim pripadnim svojstvima.

Taj se niz uniformnih znakova obrađuje u objektu klase SentenceParser koji sadrži parser kontestkno neovisne gramatike izgrađen metodom rekurzivnog spusta. Pozivom metode parse() na objektu klase SentenceParser se ulazni niz uniformnih jedinki parsira i gradi se sintaksno stablo koje se prosljeđuje objektu klase SemanticParser. U klasi SemanticParser se interpretira značenje parsiranog niza, te se određuje je li parsirana rečenica semantički ispravna (primjerice, sadrži li minimalno dva proteina u sebi), te određuje se koji proteini od prepoznatih međusobno vrše interakciju, a koji ne. Podaci o prepoznatim interakcijama, te sve ostale pripadne informacije se pohranjuju upotrebom klase InteractionData. Klasa ResultWriter prima listu prepoznatih interakcija (lista InteractionData objekata) te ih zapisuje u MySQL bazu ukoliko je to parametrima programa definirano.

Svi programski parametri se podešavaju preko komandne linije i spremaju pomoću klase Config. Klasa implementira programski obrazac Singleton. Ovo konkretno znači da je moguće instancirati samo jednu instancu te klase. Drugim riječima, svaka klasa koja želi pročitati parametre samo mora instancirati objekt Config klase, te će sve instance Config klase pokazivati na isti objekt, tj. čitati će iste parametre koji su prilikom podizanja programa pročitani sa komandne linije i spremljeni u Config objekt.

Klase Article i Sentence su osnovne građevne klase koje se koriste za komunikaciju među svim fazama procesiranja rečenice, te se u njih i pohranjuju podaci o prepoznatim interakcijama.

4.3. Pretprocesiranje

4.3.1. Ekstrakcija naziva proteina iz teksta

Budući da ne postoji unificirana nomenklatura za imenovanje proteina, prepoznavanje naziva proteina u tekstu se pokazala kao netrivijalan problem. Ovdje opisana metoda ekstrakcije naziva proteina iz teksta je inspirirana člancima [9], [10] i [11].

Ekstrakcija naziva proteina se vrši u dva koraka kako je to pokazano slikom 3:

Slika 3: Pregled podsustava za ekstrakciju naziva proteina iz teksta

U prvom se koraku izvodi ekstrakcija značajki ulaznih riječi (detalji ekstrakcije značajki ulaznih riječi su dani u poglavlju “4.3.1.1. Pretprocesiranje”. Nakon što su izračunate relevantne značajke ulaznih riječi, koristi se sustav YamCha za prepoznavanje naziva proteina na temelju izračunatih značajki. Detalji o sustavu YamCha su dani u poglavlju “4.3.1.2. Sustav YamCha”. Izlaz iz podsustava za ekstrakciju naziva proteina u tekstu je ulazni niz riječi u rečenici sa dodanim svojstvima izračunatim u prvoj fazi ekstrakcije naziva proteina, te izlaz iz sustava YamCha. Svaka se riječ označava jednom od oznaka “B”, “I” ili “O”. Značenje tih oznaka je sljedeće:

- “B” označava da se ta riječ nalazi na početku niza riječi koji je naziv proteina
- “I” označava da se ta riječ nalazi unutar niza riječi koji je naziv proteina, ali to nije prva riječ u nizu
- “O” označava da se ta riječ nalazi izvan niza riječi koji je naziv proteina

Motivacija za ovakvo označavanje naziva proteina je sljedeća: ukoliko se dva naziva proteina u tekstu nalaze jedan pored drugoga, jednostavno označavanje oznakama “I” i “O” (ili ekvivalentnim) ne bi bilo dovoljno jer ne bi bilo moguće razlikovati ta dva naziva proteina, pa bi bili proglašeni jednim velikim proteinom.

4.3.1.1. Pretprocesiranje

U prvoj fazi ekstrakcije naziva proteina iz teksta se izračunavaju sljedeća svojstva:

- uloga pojedine riječi u rečenici (engl. *Part-Of-Speech tagging*)
    - Za prepoznavanje uloga pojedinih riječi u rečenici, koristi se sustav GeniaTagger razvijen specijalno za prepoznavanje uloga pojedinih riječi u rečenici u biomedicinskoj domeni. GeniaTagger je sustav razvijen u istom laboratoriju koji je zadužen i za razvoj i održavanje GENIA korpusa i specifičan je po tome što je specijaliziran za prepoznavanje uloga pojedinih riječi u rečenici za biomedicinske tekstove. Više informacija o GeniaTagger-u je dostupno u [12],[13].
- računanje korijena riječi (engl. *stemming*) upotrebom algoritma po Porteru
    - Računanje korijena riječi se izvodi kako bi se svi mogući oblici pojedine riječi (pri tom se misli na raličita vremena, glagolske oblike i slično) sveli na jedinstven zajednički oblik. Za ovu se namjenu koristi modul nltk_lite.tokenize.porter.PorterStemmer
- prepoznavanje imeničnih izraza (engl. *noun phrase*)
    - Rezultati prijašnjih istraživanja [9] su pokazali da se nazivi proteina nalaze najčešće unutar imeničnih izraza, a imenični se izrazi prepoznaju na temelju uloga pojedinih riječi u rečenici. Dapače, sustav GeniaTagger omogućuje jednostavno prepoznavanje imeničnih izraza, pa se taj aspekt sustava GeniaTagger također koristi.
- n-gram prefiksi i sufiksi svake riječi
    - Za neku riječ definiramo n-gram prefiks kao prvih n slova te riječi. Analogno, n-gram sufiks je definiran kao zadnjih n slova iste riječi. Ovo je svojstvo korisno jer mnoge riječi u nazivima proteina imaju karakteristične završetke i početke poput '-in' i '-ase' sufikse.
- ortografska svojstva
    - Ortografska svojstva uključuju jednostavna svojstva riječi poput svojstava da neka riječ ima velika i mala slova u sebi, da sadrži neko grčko slovo i slično. Popis

ortografskih svojstava koja se računaju za svaku pojedinačnu riječ je dan tablicom 1.

Naziv svojstva	***Značenje svojstva***
orthographic_digits	sadrži li riječ brojke
orthographic_greek	sadrži li riječ engleske prijevode naziva krčkih slova
orthographic_single_caps	sadrži li riječ samo jednu veličinu slova (npr, samo mala slova)
orthographic_only_caps_and_digits	sadrži li riječ samo velika slova i znamenke
orthographic_capitals_2_places	sadrži li riječ niz 1 ili više velikih slova na 2 mjesta
orthographic_letters_and_digits	sadrži li riječ samo mala i/ili velika slova i brojke
orthographic_init_caps	započinje li riječ velikim slovom
orthographic_low_caps	započinje li riječ malim slovom
orthographic_lowercase	je li cijela riječ pisana samo malim slovima
orthographic_-	sadrži li riječ povlaku
orhographic_[	sadrži li riječ znak '['
orthographic_]	sadrži li riječ znak ']'
orhographic_(	sadrži li riječ znak '('
orthographic_)	sadrži li riječ znak ')'
orthographic_,	sadrži li riječ znak ','
orthographic_.	sadrži li riječ znak '.'
orthographic_det	je li riječ član

Tablica 1: Popis ortografskih svojstava koja se računaju za prepoznavanje naziva proteina u tekstu

Ova se svojstva dodaju svakoj riječi u fazi pretprocesiranja kao atributi riječi kako bi se mogla koristiti u kasnijim fazama prepoznavanja naziva proteina. Nabrojana svojstva se koriste kao značajke na temelju kojih se trenira SVM implementiran u sklopu sustava YamCha.

4.3.1.2. Sustav YamCha

Sustav YamCha je razvijen za potrebe prepoznavanja specijalnih nizova riječi u rečenici poput imeničnih fraza, adresa u tekstu, naziva proteina i slično. Naziv YamCha je kratica sa engleskog: *Yet Another Multipurpose Chunk Annotator*. Riječ je o sustavu razvijenom upotrebom metode vektora potpore (SVM) posebno za ovakve namjene. Budući da se radi o algoritmu strojnog učenja, razvoj sustava upotrebom sustava YamCha zahtijeva da se provede i faza učenja. U svrhu učenja sustava su upotrebljena dva korpusa biomedicinskih članaka namijenjena isključivo za razvoj sustava za detekciju naziva proteina u tekstu: korpus Genia [14] i korpus Yapex [15].

Učenje prepoznavanja naziva proteina upotrebom sustava YamCha se izvodi u dva koraka:

1. Priprema podataka (pretprocesiranje)

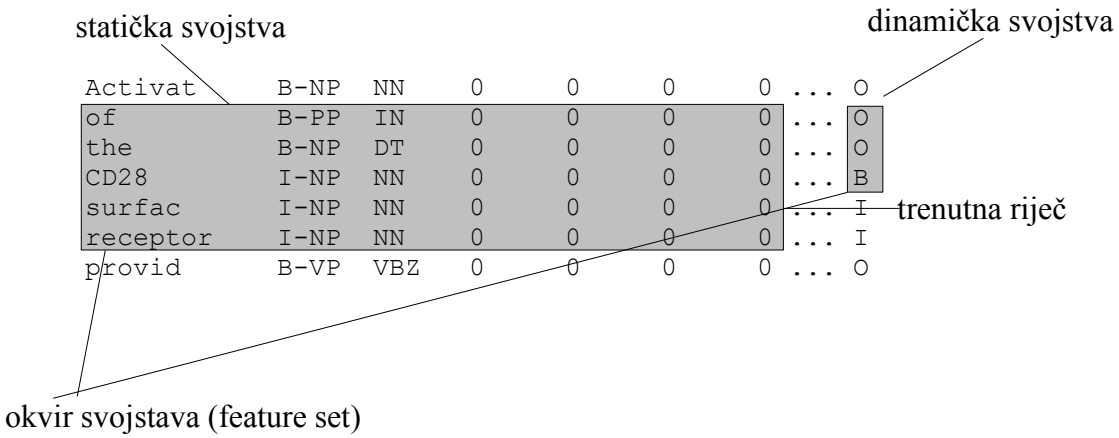
   Pretprocesiranje podataka za učenje sustava je identična faza koja će se koristiti kao prva faza jednom kada je sustav naučen. Jedina razlika je u tome što se prilikom učenja sustav uči na temelju tih podataka, a u normalnom radu sustav donosi odluke na temelju svojstava izračunatih u fazi pretprocesiranja.

2. Učenje sustava YamCha na temelju podataka pripremljenih u prethodnom koraku

   Sustav YamCha na temelju svojstava izračunatih u fazi pretprocesiranja primjenom metode vektora potpore uči prepoznavati nazive proteina u rečenicama. Detalji o svojstvima koja se računaju u fazi pretprocesiranja su opisani u prethodnom poglavlju.

   Sustav YamCha na ulazu prima ulaznu datoteku koja je formatirana organizirana po stupcima. Broj stupaca nije ograničen, ali je nužno da je u svakom retku jednak. Stupci su međusobno odvojeni jednom ili sa više praznina (praznine mogu biti razmaci ili tabulatori). Jedan redak označava jednu riječ u rečenici sa svim njenim svojstvima koja su izračunata u fazi pretprocesiranja. Drugim riječima, jedan stupac odgovara točno jednom svojstvu izračunatom u fazi pretprocesiranja. Posebno, zadnji stupac označava oznaku kojom se ta riječ treba označiti (to je oznaka koju sustav uči, u ovom slučaju 'B', 'I' ili 'O'). Prazna linija označava kraj

rečenice. Promotrimo parametre učenja na jednom primjeru ulazne datoteke:



Slika 4: Primjer za učenje sustava YamCha

YamCha koristi metodu klizećeg prozora za analizu riječi. To konkretno znači da se kod izračunavanja oznake neke riječi promatraju izračunata svojstva N prethodnih riječi, M slijedećih riječi i O prethodnih oznaka. Ovo je ilustrirano slikom 4 (dio svojstava je izdvojen sa ilustracije radi jednostavnosti i označen je sa tri točke na slici.). Okvir svojstava ili prozor se sastoji od dva dijela: statičkih svojstava (svojstva prethodnih N i slijedećim M riječi izračunata u fazi pretprocesiranja), te dinamičkih svojstava (oznake koje su dodijeljene za O prethodnih riječi). Zbog ovakve organizacije učenja, oznaka je pojedine riječi kontekstno ovisna, pa oznaka pojedine riječi ovisi o prethodnim riječima, ali i o oznakama prethodnih i slijedećih riječi u rečenici. Veličina statičkog i dinamičkog prozora se može mijenjati. Za potrebe ovog sustava odabran je statički prozor -2 .. 2 (ovo konkretno znači da se uzimaju statička svojstva dvije riječi prije trenutne, dvije riječi poslije trenutne i sama trenutna riječ), a dinamički prozor -2 .. -1 (koriste se dinamička svojstva riječi prije prethodne i prethodne riječi). Sustav YamCha koristi polinomnu jezgrenu funkciju i to je fiksno svojstvo sustava koje se ne može mijenjati.

4.3.2. Ekstrakcija ključnih riječi

Ključne riječi dijelimo u dvije kategorije:

1. Interakcijski bitne ključne riječi

   Ovo su ključne riječi koje indiciraju interakciju. Upotrebljena je lista ključnih riječi iz [16] proširena ključnim riječima korištenim u korpusu [17] prilagođenom za

razvoj sustava za prepoznavanje proteinskih interakcije u nestrukturiranom tekstu. Nakon ekstrakcije ključnih riječi, riječi su upotrebom Porterovog algoritma svedene na korijene kako bi se uspješno prepoznavali svi oblici ključnih riječi.

Popis korijena ključnih riječi koje su korištene je dan tablicom 2.

abrog	contain	format	ligand-independ	regulatori
accumul	control	former	link	replac
acetyl	convers	formic	linkag	repress
activ	decreas	formula	linker	repressor
add	demethyl	fuse	mediat	respons
addit	dephosphoryl	hasten	methyl	sever
affect	deplet	hyperexpress	modif	stabil
apoptosi	deriv	impair	modul	stimul
apparatu	destabil	inactiv	myogenesi	stimulatori
associ	detect	incit	obstruct	stimuli
bind	disassembl	includ	overexpress	substitut
block	discharg	increas	particip	suppress
blockad	downregul	induc	phosphoryl	suppressor
bound	down-regul	induct	potenti	supress
catalyz	effect	influenc	potential-depend	synthes
cleav	effect	inhibit	produc	synthesi
cluster	effector	inhibitor	promot	transactiv
coexpress	elev	inhibitori	react	upregul
complex	encod	initi	recogn	up-regul
compon	enhanc	interact	recognit	yield
compris	exhibit	interf	recruit	
conjug	express	interfer	reduc	
contact	form	interferon	regul	
	formalin-fix	ligand	regular	

Tablica 2: Popis interakcijski bitnih ključnih riječi

2. Ostale ključne riječi

Ostale ključne riječi služe isključivo kao modifikatori koji mijenjaju semantičko značenje rečenice. Popis ključnih riječi je preuzet iz [16] i dan je tablicom 3

Riječ (korijen)
between
by
of
on
with
and
did
was
not

Tablica 3Ostale korištene ključne riječi

4.4. Sintaksni analizator

Parsiranje niza uniformnih znakova koji je generiran pretprocesiranjem se obavlja u ovom koraku. U poglavlju “4.4.1. Upotrebljena gramatika” je opisana kontekstno neovisna gramatika koja je upotrebljena za parsiranje niza uniformnih znakova, a detalji implementacije parsera koji parsira nizove te gramatike su dani u poglavlju “4.4.2. Implementirani parser”.

4.4.1. Upotrebljena gramatika

Na slici 5 je dana upotrebljena gramatika:

$\langle S \rangle \rightarrow \langle INTERACTIONS \rangle$
$\langle INTERACTIONS \rangle \rightarrow \langle MOLEXPR \rangle \langle INTERACTIONS \rangle$
$\langle INTERACTIONS \rangle \rightarrow \varepsilon$
$\langle MOLEXPR \rangle \rightarrow \langle ASSIGNMENT \rangle$
$\langle MOLEXPR \rangle \rightarrow \langle RELATIONSHIP \rangle$
$\langle ASSIGNMENT \rangle \rightarrow \langle EXPR \rangle \langle NEGMAYBE \rangle KEY \langle RELATIONSHIPCONJMAYBE \rangle \langle EXPR \rangle \langle REPTRANS \rangle EOC$
$\langle ASSIGNMENT \rangle \rightarrow \langle EXPR \rangle \langle NEGMAYBE \rangle KEY \langle RELATIONSHIPPREP \rangle \langle EXPR \rangle \langle REPTRANS \rangle EOC$
$\langle ASSIGNMENT \rangle \rightarrow \langle EXPR \rangle \langle NEGMAYBE \rangle KEY \langle RELATIONSHIPPREP \rangle \langle KEYMULT \rangle \langle RELATIONSHIPCONJ \rangle \langle EXPR \rangle EOC$
$\langle NEGMAYBE \rangle \rightarrow NEGATOR$
$\langle NEGMAYBE \rangle \rightarrow \varepsilon$
$\langle RELATIONSHIPCONJMAYBE \rangle \rightarrow \langle RELATIONSHIPCONJ \rangle$
$\langle RELATIONSHIPCONJMAYBE \rangle \rightarrow \varepsilon$
$\langle REPTRANS \rangle \rightarrow TRANSITIVE\ KEY \langle EXPR \rangle \langle REPTRANS \rangle$
$\langle REPTRANS \rangle \rightarrow \varepsilon$
$\langle RELATIONSHIP \rangle \rightarrow KEY \langle RELATIONSHIPCONJ \rangle \langle EXPR2 \rangle \langle RELPREPOBJMAYBE \rangle \langle KEYMULTMAYBE \rangle EOC$
$\langle RELATIONSHIP \rangle \rightarrow KEY \langle RELATIONSHIPCONJ \rangle \langle EXPR \rangle \langle RELPREPOBJMAYBE \rangle \langle KEYMULTMAYBE \rangle \langle EXPR \rangle EOC$
$\langle RELPREPOBJMAYBE \rangle \rightarrow \langle RELATIONSHIPPREP \rangle \langle RELATIONSHIPOBJ \rangle$
$\langle RELPREPOBJMAYBE \rangle \rightarrow \varepsilon$
$\langle RELPREPOBJ \rangle \rightarrow \langle RELATIONSHIPPREP \rangle \langle RELATIONSHIPOBJ \rangle$
$\langle KEYMULTMAYBE \rangle \rightarrow \langle KEYMULT \rangle$
$\langle KEYMULTMAYBE \rangle \rightarrow \varepsilon$
$\langle KEYMULT \rangle \rightarrow KEY \langle KEYMULT \rangle$
$\langle KEYMULT \rangle \rightarrow \varepsilon$
$\langle RELATIONSHIPCONJ \rangle \rightarrow BETWEEN$
$\langle RELATIONSHIPCONJ \rangle \rightarrow OF$
$\langle RELATIONSHIPPREP \rangle \rightarrow BY$
$\langle RELATIONSHIPPREP \rangle \rightarrow ON$
$\langle RELATIONSHIPPREP \rangle \rightarrow WITH$
$\langle RELATIONSHIPOBJ \rangle \rightarrow \langle EXPR \rangle$
$\langle EXPR \rangle \rightarrow \langle NEGMAYBE \rangle \langle MOL \rangle \langle NEGMAYBEMOLMULT \rangle$
$\langle EXPR2 \rangle \rightarrow \langle NEGMAYBE \rangle \langle MOL \rangle \langle NEGMAYBE \rangle \langle MOL \rangle \langle NEGMAYBEMOLMULT \rangle$
$\langle NEGMAYBEMOLMULT \rangle \rightarrow \langle NEGMAYBE \rangle \langle MOL \rangle \langle NEGMAYBEMOLMULT \rangle$
$\langle NEGMAYBEMOLMULT \rangle \rightarrow \varepsilon$
$\langle MOL \rangle \rightarrow PROT$

Slika 5: Upotrijebljena kontekstno neovisna gramatika

Riječ je o modificiranoj verziji gramatike iz [16]. Pri tome su nezavršni znakovi svi oni navedeni unutar zagrada <>, a sve ostalo su završni znakovi. Znak ε označava prazan prijelaz, tj. prijelaz u ništa.

Gramatiku iz [16] je bilo potrebno modificirati jer se pokazalo da nije mogla uspješno parsirati jednostavne rečenice poput:

The wee1 protein kinase suppresses the entry into mitosis by mediating the inhibitory tyrosine phosphorylation of p34cdc2.

te je bilo potrebno omogućiti prepoznavanje ovakvih rečenica. Na temelju definirane gramatike je izgrađen sintaksni analizator (parser) upotrebom metode rekurzivnog spusta.

4.4.2. Implementirani parser

Parser koji parsira nizove definirane konteksno neovisne gramatike je implementiran upotrebom modula nltk_lite. Za implementaciju sintaksnog analizatora je upotrijebljena metoda rekurzivnog spusta. Implementirani parser na ulazu prima niz uniformnih znakova. Moguće vrste pojedinih znakova su dane tablicom 4:

Uniformni znak	***Značenje***
PROT	označava prepoznati naziv proteina
KEY	označava ključnu riječ koja opisuje interakciju
BETWEEN	znak za riječ “between”
BY	znak za riječ “by”
OF	znak za riječ “of”
ON	znak za riječ “on”
WITH	znak za riječ “with”
TRANSITIVE	znak za riječ 'and' kada nakon nje slijedi ključna riječ
NEGATOR	oznaka za negacijsku form (npr. “did not”)
EOC	oznaka za kraj rečenice

Tablica 4: Vrste uniformnih znakova na ulazu sintaksnog analizatora

Programski modul nltk_lite sadrži implementaciju sintaksnog analizator metodom rekurzivnog spusta, te je upravo ta mogućnost programskog modula nltk_lite upotrebljena za izgradnju sintaksnog analizatora.

4.4.3. Semantička analiza

Sintaksni analizator na svom izlazu daje sintaksno stablo ukoliko je ulazni niz uniformnih jedinki uspješno parsiran. Potrebno je analizirati generirano sintaksno stablo i izvršiti odgovarajuće akcije. Ovo se još naziva i semantička analiza sintaksnog stabla. Ukoliko je izlaz sintaksnog analizatora prazan, to znači da niti jedna interakcija nije uspješno prepoznata u rečenici.

Prilikom analize izgrađenog sintaksnog stabla, semantički analizator traži određene čvorove u stablu koji opisuju proteinske interakcije, te analizom djece tih čvorova prepoznaje podatke o proteinskoj interakciji.Čvorovi u sintaksnom stablu odgovaraju produkcijama u kontekstno neovisnoj gramatici.

Semantički analizator prvo traži u stablu sve čvorove tipa “ASSIGNMENT” i “RELATIONSHIP” jer oni odgovaraju produkcijama (broj_produkcije) i (broj_produkcije) kontekstno neovisne gramatike koje opisuju proteinske interakcije. Potom se provjerava nalaze li se u podčvorovima pronađenog stabla barem dva naziva proteina jer interakcija se mora definirati na barem dva proteina. Potom se na temelju podataka iz podčvorova grade informacije o nađenim proteinskim interakcijama. Svaka proteinska interakcija je opisana opisnikom proteinske interakcije koji je dan tablicom 5:

Polje	***Opis***
`prot1`	Naziv prvog proteina koji sudjeluje u interakciji
`prot2`	Naziv drugog proteina koji sudjeluje u interakciji
`keywords`	Lista ključnih riječi koje opisuju interakciju
`sentence`	Puna rečenica u kojoj je interakcija prepoznata
`is_negative`	Booleova vrijednost koja opisuje je li podatak o interakciji pozitivan ili negativan. Ako je pozitivan, znači da je interakcija točno navedena, tj da ta dva proteina sudjeluju u interakciji, a ako je negativan, znači da interakcije nema, tj da je u tekstu eksplicitno navedeno da ta dva proteina ne sudjeluju u interakciji.

Tablica 5 Opisnik proteinske interakcije

Izlaz iz semantičkog analizatora jest niz opisnika proteinskih interakcija opisuju prepoznate proteinske interakcije u tekstu.

5. Rezultati

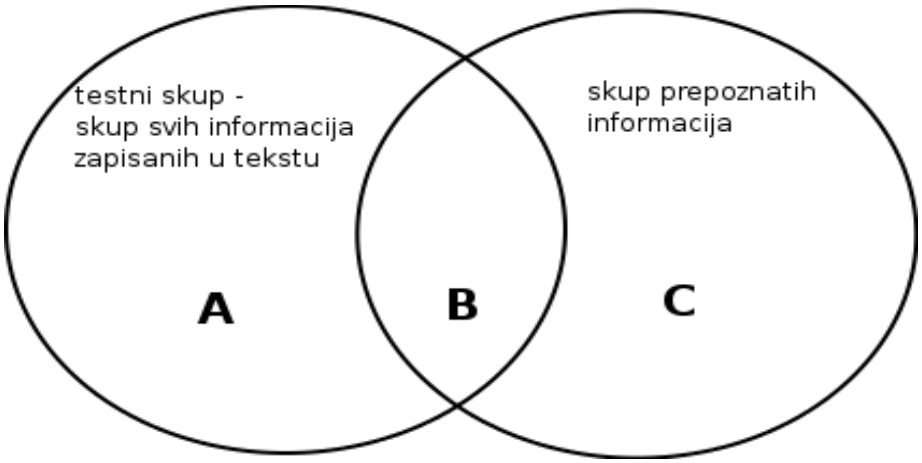
Postupak testiranja je opisan u poglavlju “5.1 Testiranje” dok su rezultati testiranja dani u poglavlju “5.2 Rezultati testiranja”. Testiranje se provodilo u dvije faze: prvo je testiran podsustav za prepoznavanje naziva proteina, dok su performase cjelokupnog sustava za prepoznavanje proteinskih interakcija u tekstu testirane u drugom koraku.

5.1. Testiranje

Prilikom testiranja sustava nad skupom za testiranje promatramo dva skupa podataka:

- skup informacija koje postoje u testnom skupu
- skup informacija koje je sustav prepoznao iz testnog skupa

Odnos ova dva skupa informacija je dan slikom 6:



Slika 6: Prikaz skupova svih i prepoznatih informacija

Na temelju ta dva skupa definiramo mjere:

- preciznost (engl. precision) je definirana kao omjer broja tocno prepoznatih informacija u skupu i ukupnog broja prepoznatih informacija. Prema slici 6, ovo bi bio omjer B/C
- odziv (engl. recall) je definiran kao omjer broja točno prepoznatih informacija u skupu i ukupnog broja točnih informacija u testnom skupu. Prema slici 6, ovo bi bio omjer B/A
- f-mjera (engl. f-measure) je definirana kao harmonijska sredina preciznosti i odziva

5.1.1. Testiranje prepoznavanja naziva proteina u tekstu

Testiranje prepoznavanja naziva proteina u tekstu je izvođeno prema mjerama performansi definiranim u prethodnom poglavlju, za svaku od moguće tri oznake (“B”, “I” ili “O”) posebno, te su onda izračunate prosječne kumulativne performanse. Tako su definirane mjere kao što su b_preciznost, b_odziv i b_f_mjera koje označavaju preciznost, odziv i f-mjeru za prepoznavanje oznake “B”, te su analogne mjere definirane i za prepozavanje znakove “I” i “O”.

Potom su mjerene performanse prepoznavanja punih naziva proteina. Pri tom su mjereni postoci prepoznavanja naziva proteina na dva načina:

- Strogo (engl. strict) prepoznavanje. Naziv proteina se smatra točno prepoznatim samo ako su sve riječi u nazivo točno prepoznate.
- Površno (engl. sloppy) prepoznavanje. Naziv proteina se smatra točno prepoznatim ako je barem jedna riječ iz naziva proteina ispravno prepoznata.

Testiranje prepoznavanja naziva proteina u tekstu se izvodilo na 1955 prethodno nekorištenih rečenica iz GENIA korpusa. Potom je isto mjerenje ponovljeno na Yapex korpusu.

5.1.2. Testiranje prepoznavanja proteinskih interakcija u tekstu

Testiranje prepoznavanja proteinskih interakcija u tekstu je izvršeno nad korpusom PICorpus proteinskih interakcija. Pri tom su rezultati ručno analizirani jer u korpusu PICorpus ne postoji jedinstvena tzv. “solution” datoteka koja bi popisivala sve proteinske interakcije u tekstu. Ovdje ćemo izdvojiti neke tipične primjere prepoznavanja proteinskih interakcija i kako su ocjenjivani pojedini primjeri.

Kod mjerenja su moguća četiri slučaja odnosa točnog i prepoznatog rješenja:

- True Positive (TP) – sustav je prepoznao interakciju u tekstu (Positive) i to ispravno

(True)

- True Negative (TN) – sustav nije prepoznao interakciju u tekstu (Negative) i to ispravno (True) jer interakcije u danom tekstu niti nema
- False Positive (FP) – sustav je prepoznao interakciju u tekstu (Positive), ali neispravno (False) jer interakcije u tekstu nema
- False Negative (FN) – sustav nije prepoznao interakciju u tekstu (Negative), ali neispravno (False) jer postoji interakcija u danom tekstu

Pomoću ova četiri slučaja dalje definiramo preciznost i odziv:

- $Preciznost = \frac{TP}{TP + FP}$
- $Odziv = \frac{TP}{TP + FN}$

Te su dvije mjere, preciznost i odziv, korištene kao mjerilo performasi sustava za prepoznavanje proteinskih interakcija u tekstu.

5.1.2.1. Primjer krivog prepoznavanja proteinske interakcije - False Positive

Promotrimo sljedeću rečenicu:

We have analyzed the cell cycle regulation of human cyclin-dependent kinase 2 (CDK2), a protein closely related to cell cycle-regulatory protein kinase CDC2.

Sustav je prepoznao sljedeće komponente:

- protein 1: “human cyclin-dependent kinase 2”
- protein 2: “(CDK2)”
- ključna riječ: “regulation”

Kao što je iz navedenog vidljivo, sustav je ispravno prepoznao prvi protein (“human cyclin-dependent kinase 2”), međutim kraticu koja se odnosi na taj protein (“(CKD2)”) je prepoznao kao drugi protein, dok protein “CDC2” nije prepoznat. Ovo krivo prepoznavanje naziva proteina je dovelo do krivog prepoznavanja proteinske interakcije.

5.1.2.2. Primjer krivog neprepoznavanja proteinske interakcije – False Negative

Promotrimo primjer:

GST-E2F-1 was phosphorylated by cyclin A-cdk2 more efficiently than by cyclin E-cdk2.

Sustav je prepoznao sljedeće komponente:

- protein 1: GST-E2F-1
- protein 2: cyclin A-cdk2
- ključna riječ phosphorylated

Sustav je uspješno prepoznao interakciju između proteina “GST-E2F-1” i proteina “cyclin A-cdk2”, međutim nije uspio prepoznati protein “cyclin E-cdk2”, te tu interakciju. Iako se ta interakcija ne navodi eksplicitno u tekstu, ona je implicitno navedena jer se navodi da protein “GST-E2F-1” učinkovitije utječe na protein “cyclin A-cdk2” nego na protein “cyclin E-cdk2”.

5.1.2.3. Primjer točnog prepoznavanja proteinskih interakcija – True Positive

The transcription factor E2F interacts with the retinoblastoma product and a p107-cyclin A complex in a cell cycle-regulated manner.

Sustav je uspješno prepoznao interakciju između “transcription factor E2F” i “retinoblastoma product”, te interakciju između “transcription factor E2F” i “p107-cyclin A complex” uz ključnu riječ “interacts”.

5.2. Rezultati testiranja

U poglavlju “5.2.1. Rezultati prepoznavanja naziva proteina u tekstu” razmatramo prvo rezultate testiranja rada podsustava za prepozanavnje naziva proteina u tekstu. O performansama prepoznavanja naziva proteina u tekstu direktno ovise ukupne performanse cijelog sustava, te je iznimno važno da ova komponenta radi dobro i pouzdano. Rezultati testiranja cjelokupnog sustava su dani u poglavlju “5.2.2. rezultati prepoznavanja proteinskih interakcija”.

5.2.1. Rezultati prepoznavanja naziva proteina

Kao što je opisano u poglavlju “5.1.1 Testiranje prepoznavanja naziva proteina u tekstu”,

testiranje je provedeno u dvije faze: prvo su mjerene performanse prepoznavanja pojedinih oznaka riječi (“B”, “I” i “O”), a zatim je provedeno mjerenje performansi prepoznavanja punih naziva proteina.

5.2.1.1. Rezultati prepoznavanja pojedinih oznaka riječi

Rezultati prepoznavanja pojedinih oznaka riječi na dijelu korpusa GENIA su dani u tablici 6:

Oznaka	*Prezicnost (%)*	*Odziv (%)*	*F-mjera (%)*
B	75.41	74.53	74.97
I	77.08	67.04	71.71
O	96.92	97.71	97.31

Tablica 6: Rezultati prepoznavanja pojedinih oznaka riječi na dijelu korpusa GENIA

Također, testiranje je provođeno i nad korpusom Yapex. Rezultati testiranje prepoznavanja pojedinih oznaka riječi na korpusu Yapex su dani tablicom 7:

Oznaka	*Prezicnost (%)*	*Odziv (%)*	*F-mjera (%)*
B	75.38	64.01	69.23
I	55.83	60.87	58.24
O	95.66	96.4	96.03

Tablica 7: Rezultati prepoznavanja pojedinih oznaka riječi na korpusu Yapex

5.2.1.2. Rezultati prepoznavanja cijelih naziva proteina

Rezultati prepoznavanja cijelih naziva proteina se mjere na dva načina. Strogo prepoznavanje podrazumijeva da su sve riječi u nazivu proteina ispravno prepoznate, dok površno prepoznavanje znači da je samo dio riječi iz naziva proteina uspješno prepoznat. Rezultati prepoznavanja cijelih naziva proteina su dani u tablici 8:

	Broj točno prepoznatih	***Broj krivo prepoznatih***	***Uspješnost (%)***
Strogo prepoznavanje	1877	962	66.11
Površno prepoznavanje	2201	638	77.53

Tablica 8: Rezultati prepoznavanja cijelih naziva proteina na dijelu korpusa GENIA

I u ovom je slučaju testiranje provedeno i nad Yapex testnim korpusom. Rezultati prepoznavanja su dani tablicom 9:

	Broj točno prepoznatih	***Broj krivo prepoznatih***	***Uspješnost (%)***
Strogo prepoznavanje	934	697	57.27
Površno prepoznavanje	1113	549	66.34

Tablica 9: Rezultati prepoznavanja cijelih naziva proteina na korpusu Yapex

5.2.2. Rezultati prepoznavanja proteinskih interakcija

Prepoznavanje proteinskih interakcija u tekstu je izvođeno na korpusu PICorpus. Analizirani su rezultati prepoznavanja na svih 363 rečenice i rezultati su dani tablicom 10.

Odnos točnog rezultata i odgovora sustava	***Broj odgovora sustava***
True Positive (TP)	125
False Positive (FP)	73
False Negative (FN)	165

Tablica 10 Rezultati prepoznavanja proteinskih interakcija na korpusu PICorpus

Iz ovih podataka možemo izračunati osnovne mjere performanse sustava: preciznost i odziv. Te su mjere dane tablicom 11.

Mjera	***Rezultat (%)***
Preciznost	63,13
Odziv	43,10

Tablica 11Performanse prepoznavanja proteinskih interakcija

Također, kako bi se mogao ocijeniti učinak performansi sustava za prepoznavanje naziva proteina na cjelokupan sustav, na PICorpusu su mjerene i performanse sustava za prepoznavanje proteina. Pri tom su postignuti rezultati prikazani tablicama 12 i 13.

Odnos točnog rezultata i odgovora sustava	***Broj odgovora sustava***
True Positive (TP)	742
False Positive (FP)	44
False Negative (FN)	451

Tablica 12: Rezultati prepoznavanja naziva proteina na korpusu PICorpus

Mjera	***Rezultat (%)***
Preciznost	94.4
Odziv	62.6

Tablica 13: Performanse prepoznavanja naziva proteina na korpusu PICorpus

Konačno, promatrani su rezultati prepoznavanja proteinskih interakcija samo u rečenicama u kojima je podsustav za prepoznavanje naziva proteina savršeno prepoznao sve proteine u rečenici. Takve rečenice su izdvojene, te je mjerenje performansi prepoznaavnaj interakcija provedeno samo nad tim rečenicama. Pri tome su od ukupno 368 rečenica u korpusu nazivi proteina savršeno prepoznati u 83 rečenice. Ovo je mjerenje izvršeno kako bi se utvrdile performanse sintaksnog i semantičkog analizatora. Rezultati ovog mjerenja su dani tablicama 14 i 15.

Odnos točnog rezultata i odgovora sustava	***Broj odgovora sustava***
True Positive (TP)	44
False Positive (FP)	8
False Negative (FN)	29

Tablica 14: Rezultati prepoznavanja proteinskih interakcija u rečenicama sa savršeno prepoznatim nazivima proteina

Mjera	***Rezultat (%)***
Preciznost	84.6
Odziv	60.3

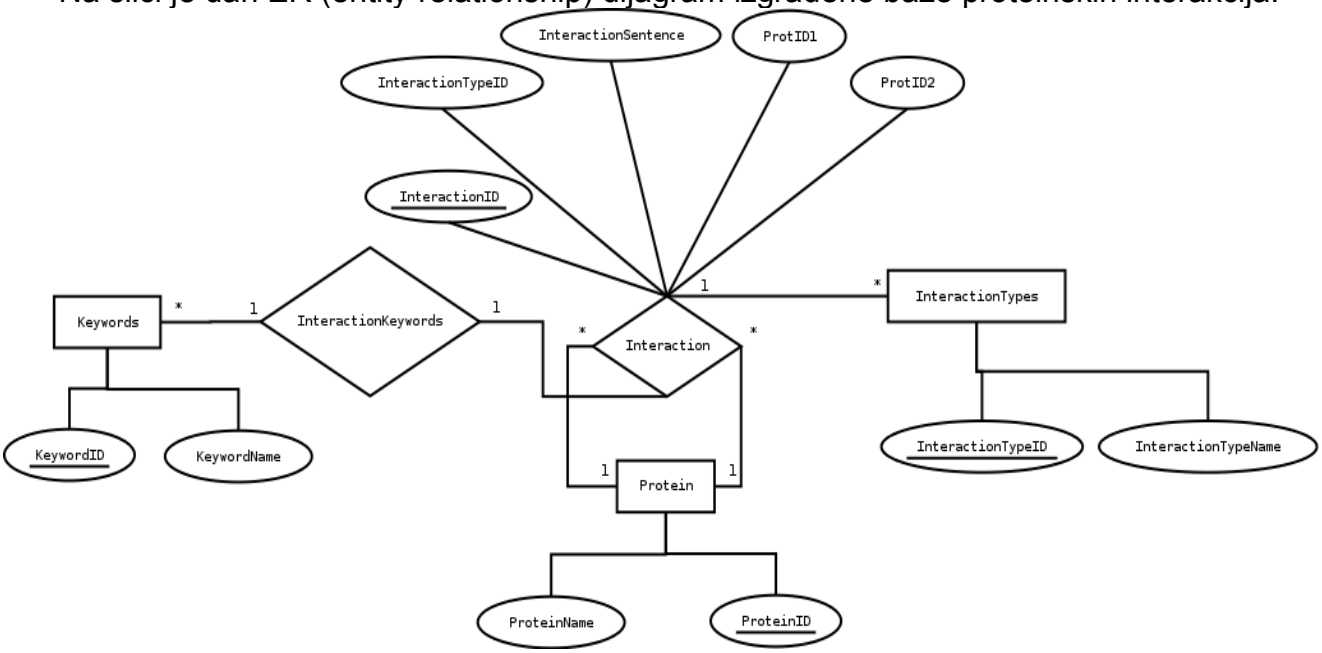
Tablica 15: Performanse prepoznavanja proteinskih interakcija u rečenicama sa savršeno prepoznatim nazivima proteina

6. Implementacija

U ovom će poglavlju biti opisana implementacija sustava na jednom problemu u stvarnom svijetu. Nakon dohvaćanja skupa biomedicinskih članaka iz baze PUBMED, sažeci članaka dohvaćeni upitom će biti obrađeni upotrebom izgrađenog sustava, te će se izgraditi baza podataka proteinskih interakcija prepoznatih u skupu sažetaka. Na taj je način pokazana praktična primjena sustava i njegova primjenjivost u stvarnom svijetu.

6.1. Baza proteinskih interakcija

Na slici je dan ER (entity-relationship) dijagram izgrađene baze proteinskih interakcija:

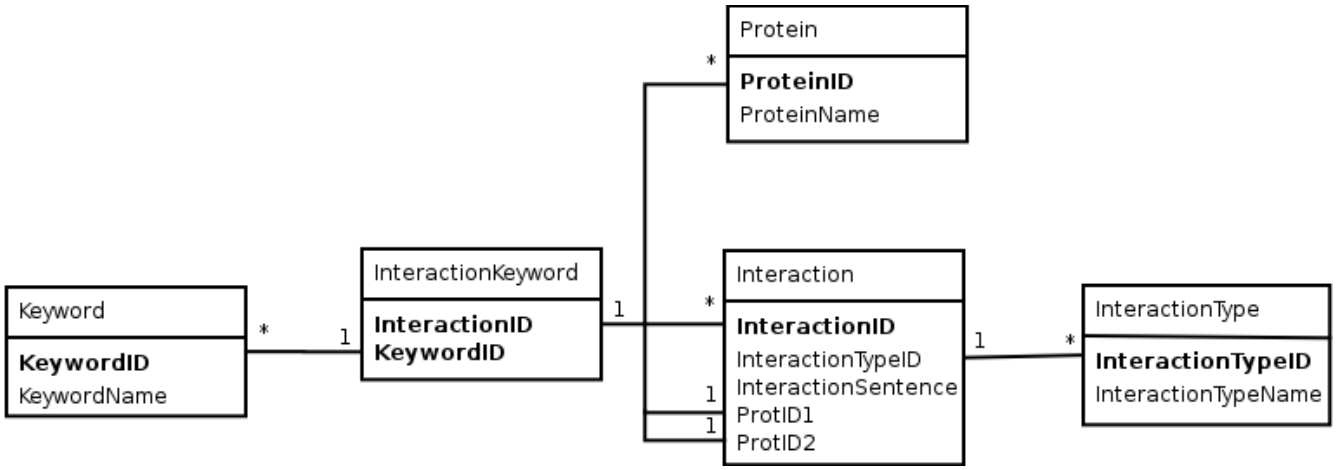


Slika 7: ER dijagram baze proteinskih interakcija

Proteine identificiramo pomoću njihovih imena (ProteinName), te jedinstvenog broja koji automatski generira baza (ProteinID). U interakciji sudjeluju uvijek dva proteina, te stoga relacija Interaction ima atribute ProtID1 i ProtID2 pomoću kojih se vezuje na entitet Protein. Dodatno, relacija Interaction sadrži i atribut InteractionSentence u kojem je zapisana rečenica u kojoj je proteinska interakcija prepoznata. Ključne riječi koje opisuju interakciju su zapisane u entiteu Keywords, a povezane su s relacijom Interaction preko relacije InteractionKeywords. Tip prepoznate interakcije zavisi o gramatičnoj produkciji pomoću koje je interakcija prepoznata (produkcije ASSIGNMENT i RELATIONSHIP), te je zapisan u entitetu

InteractionTypes.

Na temelju ovog ER dijagrama je izgrađena baza podataka upotrebom sustava za upravljanje bazama podataka MySQL, verzija 4.15. Konačan izgled tablica u bazi i njihovih veza je dan slikom 8:



Slika 8: Konačan izgled baze proteinskih interakcija

6.2. Korištenje sustava u izgradnji baze proteinskih interakcija

Korištenje sustava za izgradnju baze proteinskih interakcija se sastoji od odabira skupa sažetaka koji će biti obrađeni, te njihove obrade. U ovom će poglavlju ti postupci biti ukratko opisani.

6.2.1. Odabir sažetaka iz PUBMED baze

Pristup PUBMED bazi je moguć putem web sučelja [8]. Sažeci se mogu dobiti kao rezultat neke pretrage. Za ovu je primjenu korišten sljedeći upit u bazu: “Yeast two hybrid protein interaction”. Ovakav je upit korišten jer je riječ o jednoj od najčešćih metoda dokazivanja proteinskih interakcija, te članci čiji sažeci sadrže tu frazu stoga sigurno govore o bar jednoj proteinskoj interakciji. Dodatno je putem ovog upita moguće ograničiti pretragu na samo određenu vrstu ili slično što je korisno za izradu baze proteinskih interakcija za samo određenu vrstu. Ovakva pretraga je rezultirala sa skupom od 4685 sažetaka.

6.2.2. Obrada sažetaka

Sažeci su svi stisnuti u jednu, neformatiranu tekstualnu datoteku. Potrebno ih je prebaciti u neki od podržanih xml formata. Za tu je svrhu razvijena jednostavna aplikacija, pubmed2pi koja prebacuje ulaznu datoteku u pubmed formatu u XML datoteku zapisanu u PICorpus formatu. Budući da PUBMED datoteka nije uopće strukturirana, rečeni alat pokušava pogoditi strukturu ulazne datoteke, međutim ponekad su potrebne ručne intervencije kako bi se kompenzirale greške. Pokazalo se da alat za transformaciju PUBMED ulazne datoteke griješi u 15% slučajeva. Nakon prebacivanja ulazne datoteke u odgovarajući format, pokrenuta je obrada nad datotekom.

6.3. Rezultati

Kao rezultat obrade ulaznog skupa sažetaka, generirana je baza proteinskih interakcija koja sadrži 6666 proteinskih interakcija i 6063 proteina. Također, za svaku interakciju je navedena rečenica u kojoj je interakcija prepoznata. Cijela baza je spremljena na pratećem CD ROM mediju.

7. Diskusija

Izgrađeni sustav je visoko modularan, te je dodavanjem novih modula moguće proširivati funkcionalnost sustava u vidu podržavanja novih korpusa, računanja novih značajki i slično. Također, zbog visoke razine modularnosti, moguće je pojedine komponente sustava implementirati na drugačiji način, te se time omogućava poboljšavanje performansi sustava kroz poboljšavanje performansi samo jedne komponente, primjerice, prepoznavanja naziva proteina.

Kod prepoznavanja naziva proteina, Kaoru Yamamoto i drugi u [9] su prijavili f-mjeru 70% na GENIA korpusu, što je nešto lošije od ovdje opisanog sustava (f-mjera 77%), dok kod su Yapex korpusa postigli nešto bolje rezultate. Ovo je rezultat činjenice da su Kaoru Yamamoto i drugi dva puta učili svoj sustav – jednom na razvojnom skupu iz GENIA korpusa, a drugi puta na testnom skupu iz Yapex korpusa, dok je ovdje razvijeni i opisani sustav isključivo učen na razvojnom skupu iz GENIA korpusa te je tako naučen sustav testiran na Yapex korpusu.

Joshua M. Temkin i Mark R. Gilder su u [16] prijavili stopu odziva od 63.9% i stopu preciznosti od 70.2%, međutim rezultate ovdje opisanog sustava (stopa preciznosti 63,13% i stopa odziva 43,10%) je teško usporediti sa njihovim jer nisu testirali sustav na nekom standardnom korpusu (poput PICorpusa) i nisu javno objavili svoj korpus. Također, budući da je taj sustav zasnovan na ručno izgrađenom riječnikom naziva proteina, rezultati koje bi taj sustav postigao na nekom korpusu koji nije prije viđen ostaju upitni.

Izolirano mjerenje performansi sustava za prepoznavanje proteinskih interakcija je provedeno samo na rečenicama iz PICorpusa kod kojih su nazivi proteina savršeno prepoznati. U tom je slučaju sustav postigao preciznost od 84.6% i odziv od 60.27% (f-mjera 70.41%). Ovaj je rezultat nešto bolji nego što su Joshua M. Temkin i Mark R. Gilder prijavili u [16] što je i logično jer je kontekstno neovisna gramatika korištena u ovom sustavu nastala proširivanjem gramatike koju su oni razvili.

Izgrađeni sustav ima višu mjeru preciznosti od mjere odziva što se i očekuje od ovakvog sustava zbog njegove namjene (izgradnja baza proteinskih interakcija). Visoka preciznost znači da od onih interakcija koje je sustav prepoznao, većina je ispravna. Nizak odziv se

može anulirati većom količinom ulaznih podataka (sažetaka članaka). Budući da je javno dostupna ogromna količina biomedicinskih članaka o proteinskim interakcijama, ovo ne bi trebalo predstavljati veći problem.

Dodatna poboljšanja performansi sustava su moguća dodavanjem novih karakteristika koje se uzimaju u obzir prilikom postupka pretprocesiranja u sustavu za prepoznavanje naziva proteina. Pokazalo se da u većini slučajeva kada interakcija nije bila prepoznata, uglavnom je problem bio u neprepoznavanju naziva proteina. Upravo je ovaj problem uzrok relativno niskoj mjeri odziva sustava. Također, kao što je i pokazano u “5.2.2. Rezultati prepoznavanja proteinskih interakcija”, sustav za prepoznavanje naziva proteina također pokazuje visoku mjeru preciznosti uz nešto nižu mjeru odziva što se direktno reflektiralo i na ukupno veću preciznost i manji odziv. Poboljšanje odziva podsustava za prepoznavanje naziva proteina bi uvelike poboljšalo odziv cjelokupnog sustava.

Dodatno, sustav bi se mogao proširiti da omogući predviđanje proteinskih interakcija koje nisu eksplicitno navedene u tekstu. Ovo je moguće ostvariti vođenjem statistike zajedničkog pojavljivanja pojedinih proteina u rečenici u slučajevima da među njima nije prepoznata interakcija, te naprednim metodama text mininga prepoznavati korelacije među proteinima kod kojih nije eksplicitno navedena interakcija u rečenici.

8. Sažetak

Naglim razvojem tehnologije, a naročito biotehnologije, sve se češće javlja potreba sa sažimanjem poznatih informacija o proteinskim interakcijama na jedan konzistentan i lako dostupan način. Kao rezultat te potrebe, postoje velike baze poznatih proteina kao i baze proteinskih interakcija. Također, zbog dostupnih ogromnih količina biomedicinskih članaka vezanih za proteinske interakcije, javlja se ideja da se automastkom obradom članak o proteinskim interakcijama izgradi baza proteinskih interakcija. Ovim bi se uštedila znatna količina vremena potrebna za proučavanje članaka, te ručno unošenje rezultata analize u bazu.

U sklopu ovog diplomskog rada je razvijen algoritam za prepoznavanje proteinskih interakcija u biomedicinskim tekstovima. Algoritam je razvijen upotrebom metode vektora potpore za prepoznavanje naziva proteina u tekstu, te upotrebom kontekstno neovisne gramatike i riječnika ključnih riječi. Pri tome je postignuta preciznost oko 60%. Konkretno, ovo znači da je 60% podataka u izgrađenoj bazi točno. Stoga se razvijeni sustav može koristiti kao koristan alat prilikom izgradnje baza proteinskih interakcija. Zbog načina na koji su interakcije pohranjene u bazu podataka, iskusan istraživač može na jednostavan i brz način ukloniti pogrešno prepoznate interakcije, te time povećati točnost podataka.

Na kraju je dan primjer korištenja sustava na primjeru iz stvarnog svijeta. Obrađeno je 4685 sažetaka, te je kao rezultat toga prepoznato 6666 interakcija. U prepoznatim interakcijama je sudjelovalo 6063 proteina. Time je pokazana upotrebljivost sustava u stvarnome svijetu.

9. Zaključak

Izgrađen je modularan sustav za prepoznavanje proteinskih interakcija u tekstu opisan u diplomskom zadatku. Na praktičnom primjeru je prikazana primjena sustava u stvarnom svijetu, te je pomoću sustava izgrađena baza proteinskih interakcija.

Ovakav sustav može poslužiti istraživačima za izgradnju baze proteinskih interakcija jer eliminira potrebu za čitanjem cijelih sažetaka svih članaka. U izgrađenoj bazi proteinskih interakcija, preko 60% ih je točno prepoznato. Zbog načina na koji je baza izgrađena, izdvojene su reference na članke u kojima su interakcije prepoznate, a uključeni su i citati iz sažetaka članaka. Na temelju tih citata, istraživač može na jednostavan način eliminirati sve interakcije koje su krivo prepoznate u bazi. Cijeli bi postupak trebao dovesti do povećanja učinkovitosti istraživača kod izgradnje sustava baza proteinskih interakcija.

Konačno, izgradnjom web aplikacije pomoću koje bi administratori baze podataka mogli provjeravati prepoznate proteinske interakcije bi se zaokružio cijeli proces izgradnje baze proteinskih interakcija, no budući da ovo nije tema ovog diplomskog rada, to ostaje kao zadatak za dovršenje cijelog okruženja.

10. Literatura

[1] UCLA:"Database of Interacting Proteins", 1999-2004, http://dip.doe-mbi.ucla.edu/
[2] H.Husi:"Protein-Protein Interaction Database", 2002, http://www.anc.ed.ac.uk/mscs/PPID/
[3] C.J.C. Burges:"A tutorial on Support Vector Machines for Pattern Recognition", 1998,
[4] Siniša Srbljić:"Jezični procesori 1", 2002,
[5] Siniša Srbljić:"Jezični procesori 2", 2002,
[6] M.F.Porter:"An algorithm for suffix stripping", 1980,
[7] National Institutes of Health:"PubMed Central Overview", 2005,
http://www.pubmedcentral.nih.gov/about/intro.html
[8] NIH:"Entrez PubMed", , http://www.pubmed.gov
[9] Kaoru Yamamoto, Taku Kudo, Akihiko Konagaya, Yuji Matsumoto:"Protein Name Tagging for Biomedical Annotation in Text", 2003,
[10] Tomohiro Mitsumori, Sevrani Fation, Masaki Murata, Kouichi Doi, Hirohumi Doi:"Gene/protein name recognition based on support vector machine using dictionary as features", 2005,
[11] Zhenzhen Kou, William W. Cohen, Robert F. Murphy:"High-recall protein entity recognition using a dictionary", 2005,
[12] S. Kulick, A. Bies, M. Liberman, M.Mandel, R. McDonald, M.Palmer, A. Schein and L.Ungar:"Integrated Annotation for Biomedical Information Extraction", 2004,
[13] Yoshimasa Tsuruoka, Yuka Tateishi, Jing-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou and Jun'ichi Tsujii:"Developing a Robust Part-of-Speech Tagger for Biomedical Text, Advances in Informatics", 2005,
[14] Jun-ichi Tsujii et al.:"GENIA project", 2003,
[15] Gunnar Eriksson, Kristofer Franzen, Fredrik Olsson, Lars Asker, Per Linden:"Using Heuristics, Syntax and Local Dynamic Dictionary for Protein Name Tagging"", 2002,
[16] Joshua M. Temkin, Mark R. Gilder:"Extraction of protein interaction information from unstructured text using a context-free grammar", 2003,
[17] MITRE :"BioCreAtIvE-PPI: a corpus for protein-protein interactions", 2004,
http://www2.informatik.hu-berlin.de/~hakenber/corpora/
[18] Canonical Ltd:"Ubuntu homepage", , http://www.ubuntu.com
[19] Python foundation:"Python project homepage", , http://www.python.org
[20] Taku Kudo:"TinySVM: Support Vector Machines", 2002,
http://chasen.org/~taku/software/TinySVM/
[21] Taku Kudo:"YamCha: Yet Another Multipurpose Chunk Annotator", 2005,
http://chasen.org/~taku/software/YamCha/
[22] :"GENIA Tagger - part-of-speech tagging and shallow parsing for biomedical text", 2006,
http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/
[23] MySQL AB:"The world's most popular open source database", 1995-2006,
http://www.mysql.com
[24] :"MySQL for Python", 2006, http://sourceforge.net/project/showfiles.php?group_id=22307

11. Dodatak A: Zahtjevi i korištena programska podrška

U razvoju sustava je korištena sljedeća programska podrška:

Korištene komponente	
Operacijski sustav	Ubuntu Linux 6.06 LTS
Programski jezik	Python 2.4.3
Dodatni moduli	
SVM biblioteka	TinySVM 0.09
Specijalizirana biblioteka za prepoznavanje niza riječi (engl. chunker)	YamCha 0.33
Specijalizirana python biblioteka za razvoj sustava za rad sa nestrukturiranim tekstom	nltk_lite
Sustav za prepoznavanje uloga riječi u rečenici (engl. POS tagging)	GeniaTagger 2.0.2
Modul za pristup bazi podataka	*MySQL-python 1.2.1_p2*
Ostali programski alati	
Alat za pisanje i prijelom teksta	OpenOffice.org 2.0.2
Alat za crtanje dijagrama i ilustracija	Dia 0.94
Upotrijebljena baza podataka	MySQL 4.1.15

Tablica 16: Popis korištene programske podrške

Ubuntu Linux je trenutno jedna od vodećih Linux desktop distribucija, a ovdje je korištena radi svoje jednostavnosti i velike zajednice korisnika, te dostupne besplatne korisničke podrške. Više detalja oko same distribucije, kao i skidanje cijele distribucije je moguće sa [18].

Python je dinamički objektno orijentiran programski jezik razvijen početkom devedesetih godina dvadesetog stoljeća. Glavni programer i vođa projekta je Guido von Rossum. Jezik se intenzivno koristi u u području obrade nestrukturiranog teksta (engl. Natural Language Processing), te postoji široka podrška i mnoštvo biblioteka upravo za tu namjenu. Python je projekt otvorenog i slobodnog karaktera, te je dostupan za skidanje sa [19]. Na istoj adresi postoji i mnoštvo dokumentacije i jednostavnih lekcija za upoznavanje sa Pythonom.

TinySVM je implementacija SVM-a specijalizirana za problem prepoznavanja uzoraka.

Razvio ju je Taku Kudo, a dostupna je za skidanje, zajedno sa dokumentacijom i dodatnim informacijama sa [20].

YamCha (Yet Another Multipurpose Chunk Annotator) je sustav za prepoznavanje uzoraka u nizovima riječi (engl. chunking) zasnovan je na metodi SVM koji je implementiran upotrebom TinySVM paketa. Paket je, zajedno sa dokumentacijom i primjerima dostupan sa [21].

GeniaTagger je sustav za prepoznavanje uloga pojedinih riječi u rečenici (Part-Of-Speech tagging) specijaliziran za rad u domeni biomedicinskih tekstova. Razvijen je u sklopu projekta na Department of Information Science, Faculty of Science, University of Tokyo, Japan. U sklopu istog projekta je razvijen i GENIA korpus za prepoznavanje entiteta u biomedinskoj domeni. Dostupan je za skidanje sa [22]. Na danoj adresi su dostupne i dodatne informacije oko implementacije sustava.

MySQL je sustav za upravljanje relacijskim bazama podataka razvijen na principima otvorenog i slobodnog softvera. Sustav podržava upitni jezik po standardu SQL'92, te je jedan od najrasprostranjenih sustava za upravljanje bazama podataka danas. Pogodan je za većinu primjena, osim za one najzahtjevnije koje uključuju replkacije baza podataka na više servera i slično. Više podataka o MySQL sustavu je dostupno sa [23].

MySQL-python je modul za Python programski jezik čija je namjena povezivanje programa pisanih tim jezikom sa MySQL bazom podataka. Modul je dostupan za skidanje sa [24], zajedno sa svom potrebnom dokumentacijom.

12. Dodatak B: Sadržaj pratećeg CD ROM medija

Popis mapa na CD ROM mediju, zajednom sa opisima pojedinih mapa je dan tablicom 17:

Mapa	***Opis***
data	mapa sa testnim korpusima
...ppi	mapa sa testnim korpusima za prepoznavanje proteinskim interakcija
.....picorpus	mapa sa PICorpus korpusom
.....pubmed	mapa sa datotekom koja sadrži pripremljene sažetke iz PUBMED baze
...protein_names	mapa sa testnim korpusima za prepoznavanje naziva proteina
.....genia_corpus	mapa sa Genia korpusom
.....yapex_corpus	mapa sa Yapex korpusom
...sprot	mapa sa datotekom za izgradnju kopije SwissProt baze
...results	mapa sa SQL datotekom koja je nastala nakon obrade PUBMED datoteke
dependencies	mapa sa softverom kojeg je potrebno instalirati kako bi sustav radio
doc	mapa sa dokumentacijom, između ostalog sadrži i pdf verziju ovog diplomskog rada
src	mapa sa izvornim kodom razvijenog sustava

Tablica 17: Sadržaj pratećeg CD ROM medija

13. Dodatak C: Upute za instalaciju sustava

13.1. Upute za instalaciju programske podrške neophodne za rad sustava

13.1.1. Zahtjevi

Da bi se sustav uspješno izvršavao na nekom računalu, potrebno je da računalo ima instaliran GNU/Linux operacijski sustav, te interpreter za programski jezik Python 2.4.3. Sustav je izrađen na Ubuntu distribuciji GNU/Linux operacijskog sustava, verzija 6.06, međutim nije nužno da se koristi ta distribucija. Također, moguće je uz manje izmjene sustav je moguće prevesti i za izvođenje na drugim operacijskim sustavima, poput MS Windows operacijskih sustava i slično prije svega zbog višeplatformske prirode programskog jezika Python.

13.1.2. Instalacija nltk_lite modula

Modul nltk_lite za programski jezik Python je specijalizirani programski modul za razvoj sustava za rad sa nestrukturiranim tekstom (engl. Natural Language Processing). Da bi se uspješno instalirao, modul zahtjeva da je na sustavu instalirana inačica 2.4.3 Python programskog jezika. Potom je potrebno provesti sljedeće korake:

13.1.2.0.1. Instalacija modula NumArray

Programski modul nltk_lite za ispravan rad zahtjeva Python modul NumArray. Koraci za instalaciju modula NumArray su sljedeći:

- potrebno je posjetiti stranicu http://prdownloads.sourceforge.net/numpy/numarray-1.5.1.tar.gz?download i skinuti datoteku numarray-1.5.0.tar.gz
- otvoriti konzolu, te ući u direktorij u koji je skinuta datoteka numarray-1.5.0.tar.gz
- raspakirati arhivu sljedećim naredbama:
  - `tar zxvf numarray-1.5.0.tar.gz`
  - `cd numarray-1.5.0`
  - `sudo python setup.py install`

- Nakon ovoga bi modul NumArray trebao biti uspješno instaliran na računalo

13.1.2.0.2. Instalacija modula nltk_lite

Procedura za instalaciju modula nltk_lite je sljedeća:

- potrebno je posjetiti stranicu http://prdownloads.sourceforge.net/nltk/nltk_lite-0.6.5.tar.gz?download, te sa nje skinuti datoteku nltk_lite-0.6.5.tar.gz
- otvoriti konzolu u direktoriju u koji je skinuta datoteka nltk_lite-0.6.5.tar.gz
- raspakirati arhivu:
  - `tar zxvf nltk_lite-0.6.5.tar.gz`
  - `cd nltk_lite-0.6.5`
  - `sudo python setup.py install`

Nakon ovoga bi modul nltk_lite trebao biti uspješno instaliran na računalo. Dodatno je moguće instalirati dodatna proširenja modula nltk_lite u vidu raznih korpusa i riječnika engleskih riječi, međutim ti moduli nisu nužni za rad sustava, te su zbog toga ovdje izostavljeni.

13.1.3. Instalacija programskog paketa YamCha

Programski paket YamCha se koristi za učenje prepoznavanja naziva proteina u rečenicama. Da bi se uspješno izvršavao, potrebno je zadovoljiti sljedeće zahtjeve:

- instaliran interpreter za programski jezik Perl, verzija 5.0 ili viša
- instaliran alat GNU make
- instalirani standardni unix alati sort, uniq, rm, cat
- C++ kompajler (gcc 2.95 ili viši)
- programski paket TinySVM

13.1.3.1. Instalacija programskog paketa TinySVM

Da bi se uspješno instalirao programski paket TinySVM, jedan od paketa o kojima programski paket YamCha ovisi, potrebno je slijedno izvršiti sljedeće korake:

U konzoli operacijskog sustava upisati redom sljedeće naredbe:

- `wget http://chasen.org/~taku/software/TinySVM/src/TinySVM-0.09.tar.gz`
- `tar zxvf TinySVM-0.09.tar.gz`
- `cd TinySVM-0.09`
- `./configure`
- `make`
- `make check`
- `su`
- `make install`

Ukoliko su svi koraci uspješno izvršeni, programski paket TinySVM bi trebao biti uspješno instaliran na računalu, te se može pristupiti instalaciji programskog paketa YamCha

13.1.3.2. Instalacija programskog paketa YamCha

Da bi se programski paket YamCha uspješno instalirao na računalo, potrebno je otvoriti konzolu operacijskog sustava i slijedno izvršiti sljedeće korake:

- `wget http://chasen.org/~taku/software/yamcha/src/yamcha-0.33.tar.gz`
- `tar zxvf yamcha-0.33.tar.gz`
- `cd yamcha-0.33`
- `./configure`
- `make`
- `make check`
- `su`
- `make install`

Ukoliko su svi koraci uspješno izvršeni, programski paket YamCha je instaliran na računalo, te su time svi zahtjevi kako bi sustav uspješno radio na računalu zadovoljeni. Ovo znači da se može pristupiti instalaciji sustava na računalo.

13.1.4. Instalacija sustava za upravljanje bazama podataka

13.1.4.1. Instalacija MySQL sustava za upravaljanje bazama podataka

Sustav za upravljanje bazama podataka je potrebno imati instaliran kako bi se sustavu omogućio upit u SwissProt bazu podataka radi računanja značajki pojedinih riječi prilikom prepoznavanja naziva proteina u tekstu, te kako bi se omogućilo zapisivanje rezultata prepoznavanja proteinskih interakcija u relacijsku bazu podataka. MySQL sustav za upravaljanje bazama podataka je dostupan za skidanje sa [23] za većinu modernih softverskih platformi, zajedno sa uputama za download, te se distribuira sa većinom Linux i Unix distribucija današnjice, te stoga postupak za njegovo instaliranje neće biti ovdje opisan.

13.1.4.2. Instalacija modula MySQL_python za povezivanje na bazu podataka

Kako bi se opisani sustav mogao povezati na MySQL sustav za upravaljanje bazama podataka, potrebno je instalirati modul MySQL_python. Modul je dostupan za skidanje sa [24]. Potrebno ga je skinuti, raspakirati, kompajlirati, te instalirati upisivanjem sljedećih naredaba:

- `wget http://umn.dl.sourceforge.net/sourceforge/mysql-python/MySQL-python-1.2.1_p2.tar.gz`
- `tar zxvf MySQL-python-1.2.1_p2.tar.gz`
- `cd MySQL-python-1.2.1_p2`
- `python setup.py build`
- `sudo python setup.py install`

13.1.4.3. Izgradnja lokalne kopije SwissProt baze podataka

Za uspješan rad sustava potrebno je imati kopiju SwissProt baze podataka. Radi povećanja brzine rada sustava, poželjno je da to bude lokalna kopija baze. U mapi sprot na priloženom CD ROM mediju se nalazi komprimirana SwissProt baza podataka koju je potrebno raspakirati, te instalirati sljedećim naredbama:

- `tar zxvf sprot_dump.sql.tgz`
- `mysql sprot < sprot_dump.sql`

13.2. Upute za instalaciju sustava

13.2.1. Instaliranje sustava

Nakon što su svi preduvjeti za uspješnu instalaciju sustava omogućeni, može se krenuti sa instalacijom sustava. Prvi korak za uspješnu instalaciju sustava jest izgraditi relacijsku bazu upotrebom MySQL sustava za upravljanje bazama podataka. Za ove je potrebe izgrađena build_db.sql skripta koja se nalazi u src mapi na pratećem CD ROM mediju. Skripta se pokreće sljedećom naredbom:

```
mysql -uusername -ppassword < build_db.sql
```

pri čemu je '`username`' potrebno zamijeniti sa odgovarajućim korisničkim imenom, a '`password`' sa odgovarajućom lozinkom. Također, rečenu je naredbu potrebno pokrenuti dok se nalazimo u src mapi na pratećem CD ROM mediju. Nakon što je ovo obavljeno, moguće je jednostavno iskopirati cijelu src mapu sa pratećeg CD ROM medija u odgovarajući direktorij na tvrdi disk, te je time instalacija sustava završena.

14. Dodatak D: Upute za korištenje razvijenog sustava

Rad sa sustavom se izvodi u dva koraka. Prvo je potrebno pripremiti ulaznu datoteku sa sažecima članaka koje će sustav analizirati, potom je potrebno podesiti programske parametre sa odgovarajućim postavkama, te je tada moguće pokrenuti obradu. Nakon obavljene obrade, podaci se mogu pregledavati čitanjem log datoteke ili direktnim iščitavanjem podataka zapisanih u bazu.

14.1. Programski parametri

Sustav podržava dvije vrste parametara: zastavice i parametre sa argumentima. Zastavice se ponašaju kao prekidači, te mogu upaliti ili isključiti pojedinu mogućnost sustava, dok parametri sa argumentima postavljaju vrijednosti nekih internih varijabli poput putanje do ulazne datoteke i slično.

U tablici 18 je dan popis programskih parametara:

Parametar	***Argument parametra***	***Opis parametra***
v	Nema, zastavica	Ukoliko je ova zastavica dignuta, program ispisuje na zaslona detaljan opis operacija koje se trenutno izvode.
d	Nema, zastavica	Ukoliko je ova zastavica dignuta, program ispisuje na zaslon dodatne (tzv, debug) informacije o trenutnim operacijama.
p	Nema, zastavica	Ukoliko je ova zastavica dignuta, program vrši samo fazu pretprocesiranja i ispisuje ju na zaslon. Upotrebom standardnih mogućnosti operacijskih sustava, izlaz programa se preusmjeri u datoteku radi kasnije analize.
w	Ime baze podataka u koju će se zapisivati rezultati	Ovim se parametrom definira ime baze podataka u koju će se zapisivati rezultati prepoznavanja proteinskih interakcija
f	Putanja do ulazne datoteke	Ovim se parametrom definira putanja do ulazne datoteke koja se obrađuje.
l	Putanja do log datoteke	Ovim se parametrom definira putanja do log datoteke u koju se zapisuju sve informacije umjesto da se zapisuju na zaslon

Parametar	*Argument parametra*	*Opis parametra*
t	Vrsta ulazna datoteke.	Ovim se parametrom definira koje je vrste ulazna datoteka definirana argumentom parametra 'f'. Moguće su tri vrijednosti argumenta ovog parametra: ● PICORPUS ● GENIA ● YAPEX
u	Korisničko ime za pristup poslužitelju baze podataka	Ovim se parametrom definira korisničko ime za pristup poslužitelju baze podataka koji sadrži SwissProt bazu podataka i/ili bazu podataka u koju će se zapisivati rezultati prepoznavanja proteinskih interakcija u tekstovima.
s	Lozinka za pristup poslužitelju baze podataka	Ovim se parametrom definira lozinka koja odgovara korisničkom imenu definiranom parametrom 'u'.
r	Naziv SwissProt baze podataka na poslužitelju baze podataka	Ovim se parametrom definira naziv baze podataka koja sadrži SwissProt bazu na poslužitelju definiranom 'h' parametrom.
h	Ime poslužitelja baze podataka	Ovim se parametrom definira ime poslužitelja baze podataka koji sadrži SwissProt bazu i/ili bazu podataka u koju će se zapisivati rezultati prepoznavanja proteinskih interakcija u tekstu.

Tablica 18: Programski parametri

Sintaksa navođenja parametra je takva da se parametar navodi nakon naziva programa, te se prije imena parametra navodi znak “-”. Primjerice, poziv programa:

```
./main.py -v -f ../corpuses/pubmed/data.xml -t PICORPUS -w ppi_proteins_db -h valeria.zesoi.fer.hr -u bioinfo -p bioinfo -r sprot
```

znači da se program pokreće sa ulaznom datotekom `../corpuses/pubmed/data.xml` koja je u formatu `PICORPUS`, rezultat se treba zapisati u bazu podataka `ppi_proteins_db`, na serveru `valeria.zesoi.fer.hr`. Korisničko ime za spajanje je 'bioinfo', lozinka je 'bioinfo' te je potrebno sve popratne informacije o trenutnim akcijama ispisati na ekran.