SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Paliska

MODELIRANJE BIOLOŠKIH SEKVENCI DUBOKIM NEURONSKIM MREŽAMA

Diplomski rad

Voditelj rada:
dr. sc. Tomislav Šmuc

Zagreb, veljača 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Jedan od najvažnijih dijelova svakoga živog organizma je protein. On je ujedno i jedan od glavnih čimbenika pri razlikovanju živog svijeta od neživog. Svaki protein izgrađen je od međusobno povezanih aminokiselina zbog čega ih, u matematičkom smislu, možemo reprezentirati kao konačne nizove aminokiselina.

U novijim tehnikama strojnog učenja, od posebnog interesa je promatranje podataka kao sekvence istovjetnih gradivnih elemenata. Takve sekvence se pojavljuju u mnogim područjima. Primjerice, u području glazbe svaku pjesmu možemo modelirati kao sekvencu tonova. Nadalje, u području jezika rečenicu možemo prikazati kao sekvencu riječi, a riječ kao sekvencu slova. U području biologije, osim proteina, možemo promatrati *DNA* kao sekvencu nukleotida. Jedna od glavnih karakteristika sekvenci je njihova proizvoljna i međusobno varijabilna duljina što predstavlja velike izazove modelima strojnog učenja. Jedan od njih je da se skup sekvenci ne može prikazati standardnom matricom fiksne dimenzije jer bi to značilo da su sve sekvence u skupu jednake duljine. Posljedica je da nad takvim skupom ne možemo koristiti standardne algoritme strojnog učenja za rješavanje mnogih problema. Ako bismo mogli definirati funkciju na skupu sekvenci koja bi svaku sekvencu preslikala u realni vektor fiksne dimenzije, takva ograničenja bi se mogla zaobići.

Općenito, reprezentacija sekvenci proizvoljne duljine pomoću realnih vektora fiksne dimenzije iznimno je važna tehnika u području strojnog učenja. Ta tehnika se najprije pojavila u području obrade prirodnog jezika (engl. *Natural Language Processing*) zbog ideje da se svaka riječ može prikazati realnim vektorom fiksne duljine koji enkodira udaljenosti između riječi tako da su slične riječi u prostoru bliske jedna drugoj (engl. *word embedding*). Ostvarivanjem takvih reprezentacija, modeli strojnog učenja koji ih koriste postigli su vrlo uspješne rezultate, a jedan od primjera je *Google Translate* za prevođenje teksta između raznih jezika. Zbog toga su se ideje iz obrade prirodnog jezika počele koristiti kao inspiracija za dobivanje reprezentacija sekvenci i u ostalim područjima, pa tako i u području modeliranja proteinskih sekvenci. Primjerice, u radovima [2], [16] se za dobivanje reprezentacija proteinskih sekvenci koriste metode poznate pod nazivom *word2vec* i *Glove*. Spomenuti radovi bili su motivacija za istraživanje u ovom radu zbog čega su se te metode koristile kao prva faza modeliranja proteinskih sekvenci. Naime, bitna razlika između obrade jezika i modeliranja proteina je u tome da se proteini svijaju u prostoru

zbog čega je između dijelova proteinske sekvence bitno uhvatiti i udaljene interakcije, dok su u jeziku prisutne samo one bliske. Kako bi se mogle uhvatiti udaljene interakcije karakteristične za proteinske sekvence, u ovom radu su se, kao druga faza, koristile *rekurentne neuronske mreže*.

Rekurentne neuronske mreže su posebne vrste neuronskih mreža koje imaju sposobnost procesiranja sekvenci varijabilnih duljina pri čemu su podaci unutar iste sekvence korelirani te sposobnost pamćenja informacija o viđenim dijelovima sekvence na malim i velikim udaljenostima. Upravo zbog toga su rekurentne mreže prikladne za sekvencijalne podatke, a dodatno i omogućavaju dobivanje reprezentacija sekvenci fiksne dimenzije. Naime, treniranje takvih mreža omogućava pohranjivanje informacije o međudjelovanjima sekvence u težinama srednjih slojeva mreže. Budući da je broj težina fiksan, njihove vrijednosti u istreniranom modelu možemo koristiti kao vektorsku reprezentaciju proteinske sekvence. Možemo reći da težine srednjih slojeva otkrivaju razne apstraktne reprezentacije ulaznih podataka. Dodavanjem slojeva u neuronskim mrežama dobivamo *duboke neuronske mreže* s hijerarhijom slojeva. Takva hijerarhija omogućava hvatanje sve apstraktnijih svojstvenih koncepata čime se postiže automatsko učenje prikladne reprezentacije iz kompleksnih ulaznih podataka. Budući da mreža sama uči svojstva iz ulaza, atribute ne moramo generirati sami, nego se oni sami stvaraju u srednjim slojevima u procesu učenja duboke mreže.

Osnovni cilj ovog rada je istražiti različite načine dobivanja reprezentacije proteinskih sekvenci uz pomoć rekurentnih neuronskih mreža. Zbog važne uloge proteina u biološkim istraživanjima, reprezentacija proteina vektorom realnih brojeva fiksne dimenzije koji kodiraju bitne informacije o sekvenci bila bi od iznimne važnosti. Takve reprezentacije bi omogućile korištenje standardnih algoritama strojnog učenja za rješavanje raznih problema kao što je pronalazak klase, svojstva ili sličnih proteina na temelju primarne strukture. Reprezentacija fiksnom dimenzijom znači da svaki skup proteinskih sekvenci možemo preslikati u matricu fiksne dimenzije, dok zakodirane informacije o sekvenci u prostoru omogućavaju korištenje tog znanja za učenje i rješavanje problema. Dosadašnje reprezentacije proteinskih sekvenci su uglavnom bile temeljene na atributima dobivenim iz fizikalnih svojstava sekvence, dok reprezentacije dobivene iz rekurentnih mreža uključuju i informaciju o interakcijama i slijedu aminokiselina.

Nadalje, cilj je usporediti dobivene reprezentacije i ocijeniti njihovu kvalitetu na stvarnim biološkim problemima. Zbog toga se rad sastoji od tri faze učenja modela. U prvoj fazi definiramo pojam trigrama proteinske sekvence te koristimo modele iz obrade prirodnog jezika za dobivanje vektorskih reprezentacija trigrama. U drugoj fazi koristimo dobivene reprezentacije trigrama za treniranje rekurentne mreže. U trećoj fazi, kako bi se ocijenila kvaliteta naučenih reprezentacija, treniramo *Random Forest* klasifikator na tri stvarna biološka problema. Te faze su raspoređene u pet poglavlja.

U prvom poglavlju definiramo osnovne biološke pojmove vezane uz proteinske sekvence. U drugom poglavlju objašnjavamo metode i tehnike strojnog učenja korištene

za dobivanje reprezentacija. Najprije objašnjavamo pojam obrade prirodnog jezika, metode i modele koji se koriste za modeliranje jezika te na koji način povezujemo problem obrade jezika s problemom modeliranja reprezentacija proteinskih sekvenci. Nakon toga objašnjavamo pojam učenja iz sekvenci i arhitekturu korištenih dubokih neuronskih mreža. U trećem poglavlju navodimo radove i njihove rezultate koji su poslužili kao motivacija za istraživanje u ovom radu. Također, u koracima prikazujemo način korištenja modela objašnjenih u drugom poglavlju za dobivanje različitih reprezentacija. U četvrtom poglavlju navodimo rezultate modela iz trećeg poglavlja, dok u zadnjem poglavlju, na stvarnim problemima mjerimo uspješnost naučenih reprezentacija.

Ovom prilikom želim se zahvaliti mentoru dr. sc. Tomislavu Šmucu i Mariji Brbić na angažmanu i pruženoj pomoći tijekom ovog istraživanja.

Poglavlje 1

Proteini u biologiji

U ovom poglavlju predstavit ćemo osnovne pojmove vezane uz strukturu proteinskih sekvenci dovoljnih za razumijevanje pojmova korištenih u ovom radu. Za malo više detalja i objašnjenja pojmova, može se pogledati [14].

1.1 Građa života

Kao početna motivacija može poslužiti pitanje: *Što je život?* Ili, malo preciznije, koji faktori su ključni pri razlikovanju živog od neživog svijeta? S biološkoga gledišta, živo biće je organizam sa sljedećim svojstvima: *homeostaza, organizacija, metabolizam, rast, prilagodba, reagiranje na podražaje* te *razmnožavanje* (vidi [20]). Za razumijevanje rada, dovoljno je istaknuti samo jedno od tih, a to je *organizacija*. Organizacija govori kako je osnovna građevna jedinica svakoga živog bića *stanica* te je, kao takva, od posebnog interesa u biološkim područjima, a djelomično i u ovom radu. Naime, uz membranu, citoplazmu i genetski materijal (vidi [14]), jedan od glavnih dijelova stanice su upravo proteini čije je modeliranje glavni zadatak ovog rada. Građa i funkcija proteina bit će ukratko objašnjena u sljedećem potpoglavlju.

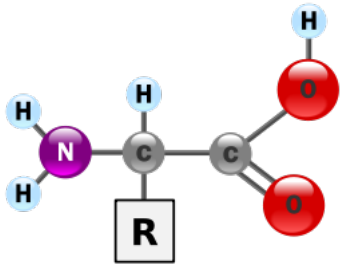
1.2 Proteini

Proteini su molekule koje su zadužene za najviše funkcija u stanici tako što omogućavaju odvijanje svih kemijskih reakcija, pružaju mehanizam za traženje i oblikovanje energije te prevođenje energije u rad. Svi proteini sastoje se od molekula koje se nazivaju *aminokiseline* te se iz tog razloga često nazivaju i proteinskom, odnosno biološkom sekvencom. Iako postoji samo 20 različitih aminokiselina, duljina proteinske sekvence može biti veća

od 4500 iz čega slijedi da prostor svih proteina teoretski može imati i više od 20^{4500} elemenata.

Aminokiseline

Sve aminokiseline sastoje se od atoma ugljika C, aminogrupe NH_3 na jednom kraju, karboksilne grupe COOH na drugom te R-skupine koja predstavlja osnovu za podjelu aminokiselina (vidi sliku 1.1). Primjerice, mogu se razlikovati polarne od nepolarnih, kisele od bazičnih. Aminokiseline se mogu razlikovati i na temelju kemijskih i fizikalnih svojstava kao što su masa, volumen, naboj. U tablici 1.1 može se vidjeti popis svih aminokiselina zajedno s odgovarajućim oznakama koje se koriste u reprezentaciji proteinske sekvence.



Slika 1.1: Osnovna građa aminokiseline

Naziv aminokiseline	Oznaka	Naziv aminokiseline	Oznaka
Alanin	A	Leucin	L
Arginin	R	Lizin	K
Asparagin	N	Fenilanin	F
Asparaginska kiselina	D	Metionin	M
Cistein	V	Prolin	P
Glutaminska kiselina	E	Serin	S
Glutamin	Q	Tirozin	Y
Glicin	G	Treonin	T
Histidin	H	Triptofan	W
Izoleucin	I	Valin	V

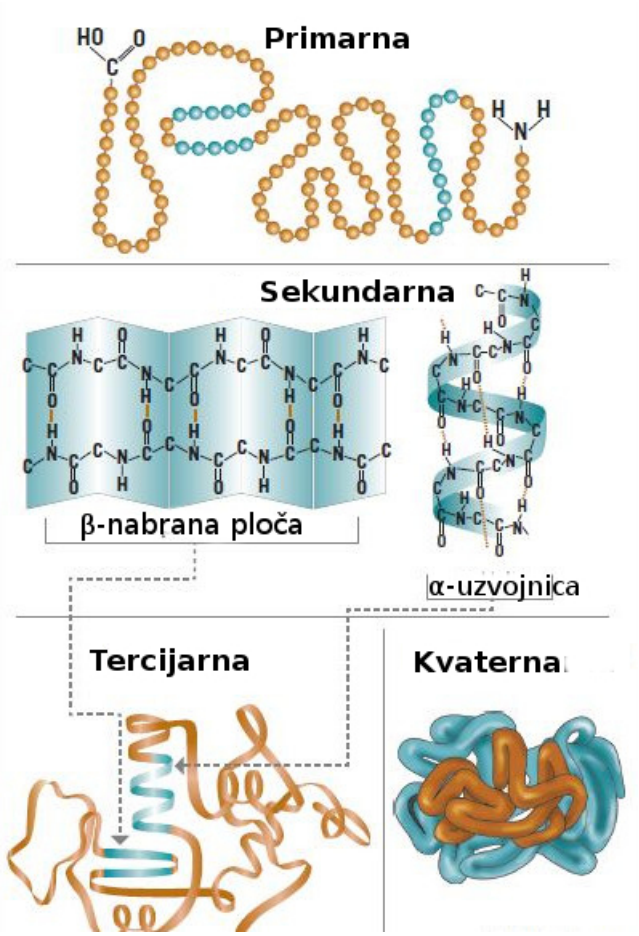
Tablica 1.1: Imena aminokiselina zajedno s odgovarajućim oznakama

1.3 Strukture proteina

Aminokiseline su međusobno povezane *peptidnom vezom* te se tako povezana sekvenca aminokiselina naziva *primarnom strukturom proteinske sekvence*. Osim primarne strukture, od posebnog interesa za istraživanje su i još tri strukture koje se međusobno nadograđuju: *sekundarna, tercijarna* i *kvaterna* struktura. Dok primarna prikazuje samo sekvencu aminokiselina koje čine odgovarajući protein, u sekundarnoj se prikazuje lokalna struktura koja ovisi o hidrogenoj vezi dijeleći se na α-uzvojnicu ili β-nabranu ploču. Tercijarna struktura definira oblik proteina u trodimenzinalnom prostoru dok kvaterna definira na koji način dijelovi proteina međudjeluju u prostoru čineći tako kompleksnu strukturu koju tada nazivamo protein (za detalje vidi [1] i [19]). Na slici 1.2 (preuzeto s [1]) može se vidjeti odnos, prijelaz i nadogradnja između struktura.

1.4 Proteini u radu

U radu je od interesa samo primarna struktura proteinske sekvence. Dakle, promatramo samo konačne nizove s elementima iz skupa $\{A, R, N, D, V, E, Q, G, H, I, L, K, F, M, P, S, Y, T, W, V\}$ (vidi tablicu 1.1), gdje duljina niza može biti od desetak pa do nekoliko tisuća elemenata (aminokiselina). Za većinu proteina poznata je primarna struktura, dok preostale strukture mogu biti nepoznate [10]. Zbog toga su proteini s poznatom primarnom strukturom temelj ovog rada. Naine, cilj je da se poznate strukture iskoriste za učenje modela koji će načiti predvijeti preostale nepoznate strukture.



Slika 1.2: Četiri strukture proteina: primarna, sekundarna, tercijarna i kvaterna (preuzeto s [1])

Poglavlje 2

Modeliranje reprezentacije

U ovom poglavlju bit će detaljnije objašnjen pojam reprezentacije te će biti prikazan prvi korak u modeliranju reprezentacije proteina. Nadalje, bit će navedeni prijašnji radovi koji se bave modeliranjem reprezentacija sekvenci. Ti radovi su ujedno bili motivacija za razvoj glavnih ideja u ovom radu.

2.1 Reprezentacija proteinske sekvence

Do sada se već na nekoliko mjesta koristio pojam *reprezentacije proteina*. Međutim, taj pojam još nismo precizno definirali zbog čega će ovo poglavlje započeti upravo tom definicijom.

Definicija 2.1.1. *Neka je $\mathcal{S}$ proizvoljan neprazan skup i neka je $n \in \mathbb{N}$. Realna vektorska reprezentacija skupa $\mathcal{S}$ je funkcija $\sigma : \mathcal{S} \to \mathbb{R}^n$.*

Analogno, ako je $\mathcal{P}$ skup svih proteinskih sekvenci, tada je realna vektorska reprezentacija skupa $\mathcal{P}$ funkcija koja svakoj proteinskoj sekvenci pridružuje realan vektor odabrane dimenzije. Takve reprezentacije koriste se i u mnogim problemima koji nisu nužno vezani uz biološke sekvence. Jedan od takvih problema je upravo problem obrade prirodnog jezika (engl. *Natural Language Processing*) te pretraživanja informacija (engl. *Information Retrieval*). U tim područjima glavni zadatak je pronaći odgovarajuću realnu vektorsku reprezentaciju riječi, fraze, rečenice ili pak cijelog dokumenta.

Općenito, distribuirane reprezentacije su se pokazale kao jedan od najuspješnijih pristupa strojnog učenja (vidi [5]). Glavna ideja tog pristupa je kodiranje informacije na način da su u preslikanom prostoru sačuvana neka poželjna svojstva koja dobro modeliraju pozadinsku distribuciju podataka. Distribuirane reprezentacije su originalno inspirirane strukturom ljudske memorije kojom se elementi pohranjuju u smislu *sadržaj-adresa* što omogućava efikasno *pretraživanje* samo na temelju opisa [2].

2.2 Problem obrade prirodnog jezika

Problem obrade prirodnog jezika jedan je od najvažnijih interesa za istraživanje i primjenu rezultata strojnog učenja. Taj problem uključuje modeliranje računalnog zapisa riječi kako bi zadaci kao što su automatsko prevođenje s jednog jezika na drugi ili predviđanje sljedeće riječi (primjer je tražilica u *Google Search* ili *Google Translate* za prevođenje čitavih rečenica između različitih jezika) bili što uspješniji. Glavna prekretnica u modeliranju jezika bila je korištenje realnih vektorskih reprezentacija riječi. Preciznije, slično kao u definiciji 2.1.1, realna vektorska reprezentacija riječi (engl. *word embedding*) je funkcija $W : \Sigma \to \mathbb{R}^n$ koja preslikava riječ nekog jezika Σ u n-dimenzionalni vektor.

Primjerice, ako za jezik uzmemo skup svih riječi hrvatskog jezika, funkcija W može riječi "pas" pridružiti neki peterodimenzionalni vektor:

$$W(\text{"}pas\text{"}) = (0.2, -0.4, 0.7, 0.5, -0.2), \tag{2.1}$$

a za detalje, vidi [22]). Najjednostavniji primjer je reprezentacija riječi nekog jezika Σ pomoću bit-vektora. Dakle, za jezik koji se sastoji od n riječi, reprezentacija riječi $r_i, i = 1, .., n$ je jedinični vektor $e^i \in \mathbb{R}^n$ sa svojstvom da su svi elementi vektora e_i jednaki 0, osim i-tog koji je jedan, odnosno, $e^i_j = \delta_{ij}$ gdje je δ_{ij} Kroneckerov simbol.

Kao još jedan primjer, promotrimo jezik $\Sigma = \{\text{"}mačka\text{"}, \text{"}pas\text{"}, \text{"}kralj\text{"}, \text{"}kraljica\text{"}\}$. Jedna reprezentacija riječi tog jezika pomoću bit-vektora (engl. *one-hot-encoding*) je sljedeća funkcija:

$$W : \Sigma \to \mathbb{R}^n$$

definirana na sljedeći način:

$$\begin{aligned} W(\text{"}mačka\text{"}) &= (1, 0, 0, 0)^T \\ W(\text{"}pas\text{"}) &= (0, 1, 0, 0)^T \\ W(\text{"}kralj\text{"}) &= (0, 0, 1, 0)^T \\ W(\text{"}kraljica\text{"}) &= (0, 0, 0, 1)^T \end{aligned}$$

Međutim, takva reprezentacija daje jedino informaciju o egzistenciji pojedine riječi, ali ne i o *kontekstu*. Pojam konteksta može se različito definirati, a u ovom ćemo radu pod kontekstom neke riječi uglavnom smatrati riječi koje se pojavljuju u blizini te riječi. Osim toga, sve riječi u preslikanom prostoru su jednako udaljene bez obzira na sličnost te u takvoj vrsti reprezentacije n ne može biti proizvoljan, nego mora odgovarati veličini rječnika. Budući da se uglavnom radi o rječnicima s nekoliko stotina tisuća riječi, u interesu nam je promatrati reprezentacije iz prostora puno manjih dimenzija. Zbog toga se za modeliranje složenijih reprezentacija koristi preslikavanje u gust vektor uz pomoć neuronskih mreža (vidi [22]). Na taj način odabrani n može biti proizvoljne veličine. Glavna ideja uvođenja takvog preslikavanja je da vektorske reprezentacije sačuvaju informaciju o kontekstu u

smislu odabranog n-dimenzionalnog prostora. Preciznije, ideja je da riječi koje često u rečenicama dolaze zajedno budu preslikane "blizu" jedna drugoj, a riječi koje se zajedno rijetko pojavljuju budu što udaljenije.

Jedna od najpoznatijih metoda kojom se postiže takvo preslikavanje je poznata pod nazivom *word2vec* autora Mikolova i suradnika (vidi [21]). Model za *word2vec* je dvoslojna neuronska mreža trenirana tako da za ulaznu riječ predviđa njezin kontekst, odnosno riječi koje ulaze u kontekst dane riječi. Arhitektura *word2vec* modela bit će detaljnije opisana u nastavku.

Nakon *word2vec* pristupa, sve više znanstvenika koji se bave dobivanjem vektorskih reprezentacija riječi u svoje je modele počela uključivati kontekst riječi. Tako je skupina autora sa Stanford sveučilišta kreirala pristup pod nazivom *Glove* (vidi [23]) koji kontekst riječi kombinira s globalnom matričnom faktorizacijom. Detaljne definicije *Glove* modela bit će također prikazane u nastavku.

Ono što se pokazalo značajnim u modelima koji koriste kontekst za dobivanje vektorskih reprezentacija riječi je smisleno grupiranje riječi u n-dimenzionalnom prostoru. Pokazalo se da su slične i povezane riječi međusobno *blizu*. Štoviše, pokazalo se kako su razlike parova vektora nekih sinonima konstantne, prikazano u primjeru 2.2.1.

Primjer 2.2.1.

$$W(\text{"}\check{z}ena\text{"}) - W(\text{"}muškarac\text{"}) \simeq W(\text{"}kraljica\text{"}) - W(\text{"}kralj\text{"})$$
$$W(\text{"}Njemačka\text{"}) - W(\text{"}Berlin\text{"}) \simeq W(\text{"}Francuska\text{"}) - W(\text{"}Pariz\text{"})$$
$$W(\text{"}toplo\text{"}) - W(\text{"}toplije\text{"}) \simeq W(\text{"}hladno\text{"}) - W(\text{"}hladnije\text{"}),$$

gdje je W reprezentacija riječi u $\mathbb{R}^n$.

Word2vec model

U ovom dijelu bit će ukratko objašnjena arhitektura *word2vec* modela. Svi detalji se mogu naći u originalnom članku autora Mikolov i suradnika [21].

Word2vec model je računski efikasan prediktivan model s arhitekturom dvoslojne neuronske mreže (vidi potpoglavlje 2.3, dio *Neuronske mreže*) dizajniran za učenje vektorske reprezentacije riječi iz teksta. Dakle, uz ulazni sloj, mreža se sastoji od jednog izlaznog sloja te točno jednoga srednjeg sloja. Vektorska reprezentacija riječi dobiva se *aktiviranjem težina* između ulaznog i srednjeg sloja. Aktiviranje težina postiže se množenjem težina istreniranog modela s ulaznim vrijednostima. Posljedica je da broj perceptrona u srednjem sloju odgovara dimenziji vektorske reprezentacije riječi.

U nastavku potpoglavlja *ulazna reprezentacija* odnosit će se na reprezentaciju dobivenu aktiviranjem težina između ulaznog i srednjeg sloja, dok će se *izlazna reprezentacija* odnositi na reprezentaciju dobivenu aktiviranjem težina između srednjeg i izlaznog sloja.

Postoje dvije različite arhitekture ovog modela: *CBOW* (engl. *Continuous Bag-of-Words*) model koji iz konteksta riječi predviđa ciljnu riječ te *Skip-Gram* model koji radi obratno, dakle iz određene riječi predviđa kontekst riječi. Iako su algoritamski ta dva modela vrlo slična, pokazalo se da u slučaju velikog skupa podataka *Skip-Gram* daje bolje rezultate (vidi [21]) zbog čega je taj model korišten i u ovom radu te će u nastavku biti opisana struktura samo tog modela. Budući da je za učenje *word2vec* modela nužan kontekst, najlakši način za treniranje tog modela je koristeći bazu velikog broja tekstova. Na taj način, kontekst možemo definirati kao fiksan broj riječi c koji se pojavljuju uz neku riječ s lijeve i desne strane te riječi. Dakle, u nizu od N riječi $\omega_1, ..., \omega_N$ iz teksta za učenje, kontekst ćemo dobiti pomicanjem prozora veličine $2c$.

Sada ćemo dati i formalnu definiciju *Skip-Gram* modela. Pretpostavimo da nam je za učenje modela dan tekst reprezentiran nizom od N riječi, $\omega_1, ..., \omega_N$. Svaka riječ ω_i je riječ iz rječnika veličine W prikazana pomoću bit-vektora dimenzije W (vidi primjer iz prošlog dijela ovog potpoglavlja). Na slici 2.1 skicirana je arhitektura *Skip-Gram* modela. Dakle, za svaku riječ kao ulaz, cilj je predvidjeti *okolinu* te riječi. Preciznije, potrebno je maksimizirati *funkciju cilja*

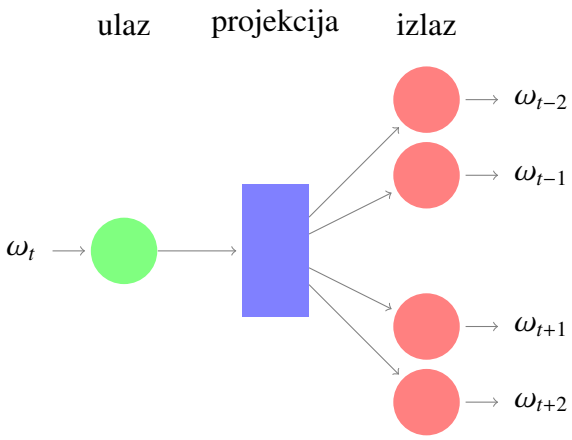
$$\frac{1}{N}\sum_{t=1}^{N}\sum_{-c\leq j\leq c, j\neq 0}\log P\left(\omega_{t+j} \mid \omega_t\right), \tag{2.2}$$

gdje je c broj riječi koje smatramo *okolinom* riječi (s obje strane ciljne riječi). Funkcija cilja (engl. *objective function*) označava funkciju koju želimo optimizirati treniranjem. Što je c veći, to je model precizniji, ali i kompleksniji. U početnom *Skip-Gram* modelu vjerojatnost $P\left(\omega_{t+j} \mid \omega_t\right)$ se definirala pomoću *softmax* funkcije

$$P\left(\omega_{t+j} \mid \omega_t\right) = \frac{\exp\left({\upsilon'_{\omega_{t+j}}}^{T}\upsilon_{\omega_t}\right)}{\sum_{\omega=1}^{W}\exp\left({\upsilon'_{\omega}}^{T}\upsilon_{\omega_t}\right)}, \tag{2.3}$$

gdje su υ_ω i υ'_ω *ulazne* odnosno *izlazne* vektorske reprezentacije od ω, a W je broj riječi u rječniku. Budući da je $\nabla P\left(\omega_{t+j} \mid \omega_t\right)$ proporcionalan s W, takva formulacija je računski presložena. Umjesto toga, koristi se računski efikasna aproksimacija softmax funkcije pod nazivom *NCE* (engl. *Noise Contrastive Estimation*) koja za svaku riječ definira fiksan broj riječi *šumova*. Riječ υ se smatra *šumom* riječi ω ako je vjerojatnost pojave riječi υ u kontekstu riječi ω minimalna. Tako je *NCE* aproksimacija vjerojatnosti $P\left(\omega_{t+j} \mid \omega_t\right)$ definirana sljedećom funkcijom

$$\log\sigma\left({\upsilon_{\omega_{t+j}}}^{T}\upsilon_{\omega_t}\right) + \sum_{i=1}^{k}\mathbb{E}_{\omega_i\sim P_n(\omega)}\left[\log\sigma\left(-{\upsilon'_{\omega_i}}^{T}\upsilon_{\omega_t}\right)\right], \tag{2.4}$$



Slika 2.1: Arhitektura neuronske mreže *Skip-Gram* modela

gdje je $P_n(\omega)$ distribucija šuma, k broj primjera šumova za svaki ulaz, a σ sigmoidalna funkcija

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

Uočimo da je cilj ove aproksimacije razlikovati ciljnu riječ ω_t od *šumova* iz distribucije koristeći logistički regresiju. Tada se maksimum funkcije cilja 2.2 dobiva za visoku vrijednost vjerojatnosti nad riječima iz konteksta (prvi sumand u 2.4) te malim vrijednostima nad riječima šumovima (drugi sumand u 2.4). Na taj način $\nabla P\left(\omega_{t+j} \mid \omega_t\right)$ više nije porporcionalan cijelom rječniku nego samo broju odabranih riječi šumova, dok je treniranje *word2vec* modela računski dovoljno efikasno. Detaljnija i preciznija objašnjenja modela te korištene aproksimacijske funkcije mogu se naći u originalnom članku [21].

Konačno, nakon istreniranog modela, vektorsku reprezentaciju riječi dobit ćemo tako da je pomnožimo s ulaznim težinama istreniranog modela.

Glove model

U prethodnom potpoglavlju prikazan je i opisan *word2vec* model koji je bio glavna motivacija za modeliranje *Glove* pristupa. Pennington, jedan od autora *Glove* modela smatrao je da je *word2vec* rješenje suboptimalno, jer se ne koristi informacija o frekvenciji pojave riječi u cijelom dokumentu. Smatrao je da je upravo statistika pojave riječi, kao što je primjerice *SVD* dekompozicija matrice (relativnih) frekvencija riječi, primaran izvor informacije za kreiranje modela s učenjem bez podrške (engl. *unsupervised*) . Iz tog razloga, baza za *Glove* model je matrica pojavljivanja definirana na sljedeći način: Neka je X matrica takva da X_{ij} označava koliko se puta riječ j pojavljuje u kontekstu riječi i, za neki

definirani kontekst. Tada $X_i = \sum_k X_{ik}$ označava koliko se puta proizvoljna riječ pojavljuje u kontekstu riječi i. Konačno, neka je $P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i}$ vjerojatnost da se riječ j pojavljuje u kontekstu riječi i.

Kao primjer možemo uzeti riječi *oblak* i *stolica*. Tada, ako uzmemo riječ *nebo* koja se s velikom vjerojatnošću pojavljuje u kontekstu riječi *oblak*, ali ne i riječi *stolica*, očekujemo da je vjerojatnost $P(nebo \mid oblak)$ relativno visoka, dok je $P(nebo \mid stolica)$ relativno mala, ali je zato omjer vjerojatnosti $P(nebo \mid oblak)/P(nebo \mid stolica)$ ponovno visok. Obratno, za riječ *stol* je vjerojatnije da se pojavljuje u kontekstu riječi *stolica*, ali ne i riječi *oblak* pa očekujemo da je omjer $P(stol \mid oblak)/P(stol \mid stolica)$ vrlo mali. Međutim, da smo uzeli riječ koja se s jednakom vjerojatnošću pojavljuje u okolini obje riječi, tada bi omjer vjerojatnosti bio blizu jedan. Iz tog razloga se, kao početna točka promatraju omjeri vjerojatnosti, radije nego obične vjerojatnosti.

Kako bismo definirali funkciju cilja, potrebno je uočiti kako omjer P_{ik}/P_{jk} ovisi o tri riječi: i, j i k. Nadalje, bilo bi korisno da upravo vrijednost tog omjera bude informacija prisutna u vektorskom prostoru riječi. Za razliku od *word2vec* metode koja optimizira funkciju modela ažuriranjem težina neuronske mreže koje kasnije koristi kao vektorsku reprezentaciju riječi, *Glove* pristup uključuje vektorske reprezentacije riječi u funkciji cilja. Time se funkcija *Glove* modela optimizira ažuriranjem vektorskih reprezentacija riječi s unaprijed odabranom dimenzijom reprezentacije d. Da bi funkcija *Glove* modela uključila informaciju o omjeru vjerojatnosti, najprije je potrebno pronaći funkciju F za koju će vrijediti sljedeće:

Za vektorske reprezentacije w_i, w_j i $w_k \in \mathbb{R}^d$ riječi i, j i k, respektivno, želimo da vrijedi

$$F\left((w_i - w_j)^T w_k\right) = \frac{P_{ik}}{P_{jk}}, \tag{2.6}$$

uz zahtjev

$$F\left((w_i - w_j)^T w_k\right) = \frac{F\left(w_i^T w_k\right)}{F\left(w_j^T w_k\right)}\,. \tag{2.7}$$

Razliku vektora u definiciji funkcije koristimo da bi se postigla konstantnost u razlikama riječi istog značenja (vidi primjer 2.2.1), dok se skalarni produkt koristi kako bismo izbjegli problem različitih dimenzija za drukčije odabranu dimenziju d. Možemo uočiti da su iskazani zahtjevi ispunjeni upravo ako je F eksponencijalna funkcija. Tada iz $F\left(w_i^T w_k\right) = \frac{X_{ik}}{X_i}$ slijedi da želimo da vrijedi

$$w_i^T w_k = \log(X_{ik}) - \log(X_i). \tag{2.8}$$

Sada se *Glove* model definira kao težinski regresijski model najmanjih kvadrata s funkcijom cilja

$$\sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T w_j + b_i + b_j - logX_{ij}\right)^2, \tag{2.9}$$

gdje je V broj riječi u vokabularu, b_i i b_j dodatni parametri linearne funkcije, w_i i w_j $\in \mathbb{R}^d$ vektorske reprezentacije riječi koje ažuriramo za odabranu dimenziju d, a f težinska funkcija definirana kao:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & x < x_{max} \\ 1 & \text{inače} \end{cases}.$$

Konačno, nakon treniranja, za svaku riječ i u vokabularu, odgovarajuća vektorska reprezentacija je w_i.

U radu [23] pokazana je i objašnjena povezanost *Glove* modela s ostalim poznatim modelima temeljenim na statističkoj informaciji iz cijelog dokumenta kao što je *SVD*, ali i povezanost sa *Skip-Gram* arhitekturom *word2vec* modela. Osim toga, performanse modela su uspoređene nad različitim skupovima podataka, uz dobivene bolje rezultate *Glove* modela u odnosu na ostale, uključujući i *word2vec*. Iz tog razloga smo u našem slučaju odlučili koristiti oba modela *word2vec* i *Glove* za dobivanje vektorskih reprezentacija proteinskih sekvenci te usporediti njihove performanse i na takvom problemu. U nastavku će biti objašnjen način na koji se problem dobivanja vektorskih reprezentacija riječi može prevesti u problem dobivanja vektorskih reprezentacija proteinskih sekvenci.

Povezivanje problema modeliranja jezika i proteinskih sekvenci

Kako bi na problem modeliranja bioloških sekvenci bilo moguće primijeniti rezultate problema obrade prirodnog jezika, potrebno je definirati korespondenciju između jezika i bioloških sekvenci. U svakom jeziku može se uočiti sljedeća građevna hijerarhija:

$$slovo \rightarrow riječ \rightarrow rečenica \rightarrow dokument.$$

S druge strane, u problemu modeliranja proteinskih sekvenci građevna hijerarhija bi mogla izgledati ovako:

$$aminokiselina \rightarrow proteinska\ sekvenca \rightarrow skup\ proteinskih\ sekvenci.$$

Međutim, možemo uočiti da se u takvom odabiru ne možemo poistovjetiti s građevnom hijerarhijom jezika. Naime, potrebno je definirati građevnu jedinicu *između* aminokiseline i proteinske sekvence. Jedan od pristupa je podjela sekvence na preklapajuće ili neprekla-pajuće *n-grame*, gdje je *n-gram* definiran kao konačan niz od točno n aminokiselina (vidi Primjer 2.2.2).

Primjer 2.2.2. *Neka je dana proteinska sekvenca:*

$$ANTVAXWFMASNQAT$$

te neka je $n = 3$*. Tada podjela gornje sekvence na nepreklapajuće 3-grame izgleda ovako:*

$$ANT\ VAX\ WFM\ ASN\ QAT.$$

S druge strane, podjela na preklapajuće 3-grame izgleda ovako:

$$ANT\ NTV\ TVA\ VAX\ AXW\ XWF\ WFM\ FMA\ MAS\ ASN\ SNQ\ NQA\ QAT.$$

Možemo uočiti da sada više ne promatramo proteinsku sekvencu kao originalnu sekvencu aminokiselina, već je promatramo kao sekvencu n-grama. Budući da je svaka proteinska sekvenca preslikana u jedinstvenu sekvencu n-grama (preklapajućom ili nepreklapajućom metodom), moguće je definirati sljedeću korespondenciju između građevnih jedinica jezika i proteinskih sekvenci:

$$\begin{aligned} slovo &\Leftrightarrow aminokiselina \\ riječ &\Leftrightarrow n\text{-}gram \\ rečenica &\Leftrightarrow sekvenca\ n\text{-}grama \\ dokument &\Leftrightarrow skup\ sekvenca\ n\text{-}grama \end{aligned}$$

Također, budući da se skup aminokiselina od kojih su izgrađene proteinske sekvence sastoji od 20 aminokiselina, dobiveni rječnik n-grama sastojat će se od 20^n elemenata. Dakle, u slučaju $n = 3$, dobivamo rječnik od 8000 trigrama, za razliku od rječnika nekog jezika koji ima više od 100 000 elemenata. Radovi vezani uz ovu tematiku koji su poslužili kao motivacija ovog rada (vidi potpoglavlje 3.1) promatrali su isključivo podjelu proteinske sekvence na trigrame. Zbog usporedbe rezultata i u ovom radu smo se udlučili promatrati podjelu za $n = 3$. Stoga će odgovarajuća korespondencija nadalje izgledati ovako:

$$\begin{aligned} slovo &\Leftrightarrow aminokiselina \\ riječ &\Leftrightarrow trigram \\ rečenica &\Leftrightarrow sekvenca\ trigrama \\ dokument &\Leftrightarrow skup\ sekvenci\ trigrama \end{aligned}$$

Zbog korespondencije riječ $\Leftrightarrow$ trigram, modeli *word2vec* i *Glove* opisani u potpoglavlju 2.2 koristit ćemo za dobivanje vektorskih reprezentacija trigrama. Možemo uočiti kako upravo modeli *word2vec* i *Glove* odgovaraju funkcijama u definiciji vektorske reprezentacije 2.1.1. Oba modela nakon treniranja s odabranim skupom za svaki trigram kao ulaz, aktiviranjem težina daju realni vektor. Glavni dio rada bavit će se upravo problemom kako iz vektorskih reprezentacija trigrama dobiti vektorsku reprezentaciju sekvence trigrama, a time i proteinske sekvence.

Napomena 2.2.3. *Iako pojmovi korišteni u ovom potpoglavlju kao što su n-gram, korespondencija između jezika i proteinskih sekvenci nisu precizno definirani, u svrhu razumijevanja rada takve definicije nisu niti potrebne. Stoga će u nastavku rada ti pojmovi biti korišteni iz prikazanih intuitivnih definicija i primjera.*

2.3 Metoda učenja iz sekvenci

Problem učenja iz sekvenci predstavlja posebno područje u strojnom učenju. Naime, većina algoritama strojnog učenja pretpostavlja da su ulazni podaci međusobno nezavisni te da su podaci fiksne duljine. Budući da se sekvencama modeliraju problemi koji ovise o vremenu kao što su vremenski nizovi te problemi koji ovise o kontekstu kao što su tekstovi i biološke sekvence, možemo odmah uočiti kako u slučaju prostora sekvenci te dvije pretpostavke nisu ispunjene. Većina takvih prostora se sastoji od nizova varijablne duljine u kojima su uzastopni dijelovi nizova međusobno strogo korelirani. Iz tog razloga, od algoritama koji se bave problemom učenja iz sekvenci zahtijevamo da ne pretpostavljaju nezavisnost podataka, da mogu prihvatiti podatke različitih duljina te da za svaki dio sekvence koriste informaciju o kontekstu tog dijela. Iako postoji više podvrsta učenja iz sekvenci (vidi [8]), za razumijevanje ovog rada dovoljno je definirati problem *klasifikacije sekvenci* (engl. *Sequence Classification*) u kojemu je cilj nizu proizvoljne duljine $(X_1, ..., X_n)$, $n \in \mathbb{N}$ pridružiti labelu $Y \in \mathbb{B}$ iz konačnog skupa labela $\mathbb{B}$. Iz toga slijedi da se skup za treniranje u tom slučaju sastoji od skupa parova (*sekvenca*, *labela*), gdje je svaka sekvenca proizvoljne duljine. Procesiranjem i modeliranjem problema bavi se familija neuronskih mreža poznata pod nazivom *rekurentne neuronske mreže* te će detaljnije biti opisane u nastavku.

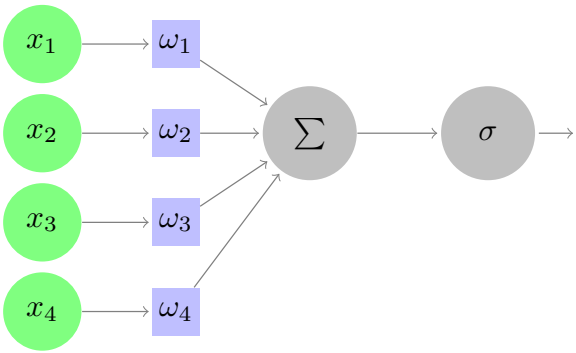
Neuronske mreže

S ciljem boljeg razumijevanja arhitekture rekurentne mreže najprije ćemo ukratko opisati arhitekturu i osnovni algoritam za učenje obične neuronske mreže. Svaka neuronska mreža sastoji se od skupa čvorova (procesnih jedinica) koje nazivamo neuroni te skupa usmjerenih bridova između neurona koji odgovaraju sinopsama u smislu biološke neuronske mreže. Najjednostavniji oblik neuronske mreže je perceptron (vidi sliku 2.2) s ulaznim podacima $x_1, .., x_N$, težinama pridruženim ulaznim podacima respektivno $\omega_1, ..., \omega_N$ te aktivacijskom funkcijom σ. Vrijednost izlaza dobiva se kao vrijednost $\sigma(\sum_i \omega_i x_i)$, gdje se za aktivacijsku funkciju najčešće uzimaju sigmoidalna (vidi jednadžbu 2.5) ili funkcija tangens hiperbolni. Složenija neuronska mreža sastojala bi se od više čvorova i slojeva, a time i većeg broja težina tako da bi primjerice težina $\omega_{jj'}$ odgovarala težini od čvora j do j'. Kako bi se neuronska mreža naučila, najprije se definira *funkcija gubitka*, u oznaci L, koja se koristi kao mjera pogreške modela. Zbog toga je cilj učenja modela iterativno ažurirati težine kako bi se smanjila vrijednost funkcije gubitka, odnosno kako bi se pronašla

točka minimuma. Najpoznatiji algoritam za ažuriranje težina neuronske mreže naziva se *backpropagation* koji koristi lančano pravilo za računanje derivacije funkcije gubitka te ažuriranje težina na temelju gradijentnog spusta. Preciznije, u svakom koraku vektor težina se mijenja u smjeru najvećeg spusta niz plohu greške. Ako se za ažuriranje težina u svakom koraku koristi samo jedan primjer, govorimo o stohastičkom gradijentnom spustu. U praksi se najčešće koristi varijanta *mini-batches* koja koristi odabrani broj primjera (više od jednog, a manje od cijelog skupa). Zbog korištenja više manjih gradijenata, pristupi koji ne koriste cijeli skup za ažuriranje težina uspijevaju zaobići prvi lokalni minimum. U slučaju stohastičkoga gradijenta, odgovarajuća težina se ažurira na način

$$\omega \leftarrow \omega - \eta \nabla_{\omega} F_i, \tag{2.10}$$

gdje je η dodatni parametar učenja, a $\nabla_{\omega} F_i$ gradijent funkcije gubitka. Preostali detalji i preciznije definicije vezani uz arhitekturu neuronskih mreža i *backpropagation* algoritam mogu se naći u [17].



Slika 2.2: Primjer arhitekture perceptrona, gdje σ predstavlja aktivacijsku funkciju

Duboke neuronske mreže

Duboke neuronske mreže su vrste neuronskih mreža čije je glavno obilježje *dubina* srednjeg sloja. Dubina neuronskih mreža znači da postoji više srednjih slojeva neurona. Svaki sloj kao ulazni skup atributa dobiva izlaz prethodnog sloja. U svakom idućem sloju, mreža uči sve kompleksnije reprezentacije ulaznih podataka. Preciznije, takve mreže mogu aproksimirati vrlo kompleksne funkcije dobivene kompozicijama funkcija iz srednjih slojeva. Na taj način, duboko učenje može podnijeti iznimno velike i visokodimenzionalne skupove podataka. U odnosu na postojeće modele strojnog učenja, veliku prednost dubokih mreža predstavlja sposobnost automatskog otkrivanja skrivenih struktura u podacima. Takve strukture također su dobivene kompozicijom, od jednostavnijih obilježja iz početnih

slojeva, pa sve do kompleksnijih iz dubljih slojeva (vidi [4]). Upravo hijerarhija slojeva omogućava hvatanje sve apstraktnijih reprezentacija ulaznih podataka čime se automatski dobivaju atributi ulaznog skupa, umjesto da se generiraju ručno što često predstavlja skup i dugotrajan posao.

Iako su duboke neuronske mreže kao ideja postojale od 1940., tek su nedavno, napredovanjem hardvera i infrastrukture te novim tehnikama treniranja i reprezentacijom podataka, uspjele ostvariti najbolje rezultate na mnogim važnim klasifikacijskim problemima kao što su raspoznavanje slike i govora te obrada prirodnog jezika. Napredovanje hardvera odnosi se na iznimno brzu paralelizaciju algoritama optimizacije uz pomoć grafičkih procesora (engl. *GPU*), dok se nove tenhike treniranja odnose na korištenje metoda kao što su *ReLU* funkcija, *batch normalization* i *Dropout* [11]. *ReLU* funkcija, definirana kao

$$f(x) = max(0, x), \tag{2.11}$$

koristi se kao aktivacijska funkcija slojeva zbog ubrzanja treniranja. *Batch normalization* odnosi se na normalizaciju ulaza svakog sloja čime pridonosi ubrzanju računanja, ali i stabilnosti treniranja zbog usporedivih vrijednosti. *Dropout* je regularizacijska tehnika za reduciranje *pretreniranja*, a odnosi se na udio neurona čije se težine ne ažuriraju prilikom treniranja. *Pretreniranje* označava pojavu u kojoj model postiže visoku točnost na skupu za treniranje, dok na neviđenim primjerima radi veliku grešku te kažemo da model nema mogućnost generalizacije. Preostali detalji vezani uz povijest i implementacija dubokih neuronskih mreža mogu se naći u [11].

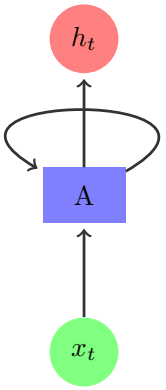
Rekurentne neuronske mreže

Rekurentne neuronske mreže (*RNN*) su neuronske mreže specijalizirane za procesiranje sekvenci varijabilne duljine. Osnovna razlika između učenja višeslojne neuronske mreže i rekurentne neuronske mreže je ideja dijeljenja parametara kroz različite dijelove modela. Za razliku od klasične mreže koja koristi zasebne parametre za svaki ulaz, rekurentna mreža dijeli iste težine kroz nekoliko koraka čime uspijeva sačuvati informaciju iz konteksta odnosno iz prethodno viđenih podataka [11].

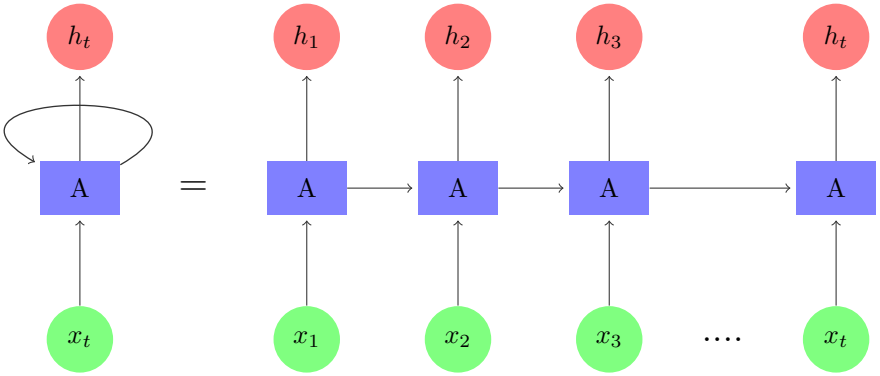
Dijeljenje težina postignuto je korištenjem petlji (vidi sliku 2.3) koja omogućava da se informacija prenosi s jednog dijela mreže na drugi. Takve petlje se mogu reprezentirati mnogostrukim kopijama iste neuronske mreže uz dijeljenje parametara što možemo vizualizirati u obliku razmotane petlje (vidi sliku 2.4). Na taj način umjesto cikličke arhitekture postiže se arhitektura duboke neuronske mreže s jednim slojem u svakom koraku, ali uz dijeljenje težina između susjednih koraka.

Napomena 2.3.1. *Iako je na početku potpoglavlja 2.3 rečeno kako nas u svrhu ovog rada zanima samo problem klasifikacije sekvenci, zbog jednostavnijeg razumijevanja, primjeri i*

skice u sljedećih nekoliko potpoglavlja prikazivat će arhitekturu mreže za problem pod nazivom sequence to sequence. U tom problemu rekurentna mreža za svaku sekvencu $x_1, ..., x_N$ *umjesto labele kao izlaz daje sekvencu iste duljine* $y'_1, ..., y'_N$*. Dakle, skup za treniranje sastoji se od parova* $(\{x_1, .., x_N\}, \{y_1, ..., y_N\})$.



Slika 2.3: Arhitektura rekurentne neuronske mreže s petljom. *A* predstavlja skupinu čvorova neurona, x_t ulaznu, a h_t izlaznu vrijednost



Slika 2.4: Arhitektura rekurentne neuronske mreže s razmotanom petljom. Svaka skupina neorona *A* predstavlja stanje srednjeg sloja neuronske mreže u nekom trenutku *t*. Između svih stanja se koriste isti parametri

Preciznije, u trenutku t, $t = 1, ..., N$ ulazne sekvence $x_1, ..., x_N$ čvorovi neuronske mreže dobivaju informaciju o ulaznom podatku x_t, ali i o srednjem sloju neuronske mreže h_{t-1} iz prethodnog stanja. Dakle, vrijednost srednjeg sloja h_t u trenutku t bismo dobili kao vrijednost rekurzivne funkcije

$$h_t = \sigma\left(W^{hx}x_t + W^{hh}h_{t-1} + b_h\right), \tag{2.12}$$

gdje je σ aktivacijska funkcija, W^{hx} matrica težina između ulaza i srednjeg sloja, W^{hh} matrica težina između susjednih stanja srednjih slojeva, dok b_h predstavlja parametar pomaka. Nadalje, izlaz y'_t u trenutku t dobiva se iz vrijednosti srednjeg sloja neuronske mreže u trenutku t kao

$$y'_t = f\left(W^{yh}h_t + b_y\right), \tag{2.13}$$

gdje, analogno, W^{yh} predstavlja matricu težina između srednjeg i izlaznog sloja, b_y parametar pomaka, a f izlaznu funkciju.

Na taj način postiže se prijenos informacije iz ulaza x_{t-1} na izlaz y'_t, ali i na kasnijim izlazima, što je upravo informacija o prethodnom kontekstu koju želimo sačuvati u slučaju procesiranja sekvenci.

Tako definiranu rekurentnu mrežu možemo trenirati koristeći modifikaciju standardnog *backpropagation* algoritma pod nazivom *backpropagation through time (BPTT)*. Za početak, potrebno je modificirati funkciju gubitka L tako da uključuje vrijednost gubitka po svim trenucima t, $t = 1, ..., N$ za ulazni par $(\{x_1, ..., x_N\}, \{y_1, ..., y_N\})$. Ako s L_t označimo vrijednost funkcije gubitka u trenutku t, tada ukupna vrijednost gubitka odgovara sumi po svim trenucima te sekvence:

$$L\left(\{x_1, ..., x_N\}, \{y_1, ..., y_N\}\right) = \sum_{t=1}^{N} L_t. \tag{2.14}$$

Nakon definicije funkcije gubitka, kao i kod običnog *backpropagation* algoritma (vidi [17]), cilj je ažurirati težine kako bi vrijednost funkcije L bila minimalna. Analogno kako je opisano u potpoglavlju 2.3, dio *Neuronske mreže*, potrebno je izračunati gradijente funkcije L u smjeru svih težina. U slučaju promatrane arhitekture, potrebno je izračunati gradijente težina b^y, b^h, W^{hx}, W^{hh}, W^{yh}, dakle $\nabla_{b^y}L$, $\nabla_{b^h}L$, $\nabla_{W^{hx}}L$, $\nabla_{W^{hh}}L$, $\nabla_{W^{yh}}L$ respektivno. Zbog dijeljenja parametara između različitih koraka gradijenti se računaju uz pomoć lančanog pravila za derivaciju kompozicije, a detaljni raspisi gradijenata za ovu arhitekturu rekurentne mreže mogu se naći u [11]. Zbog sekvencijalnog dijeljenja parametara između stanja, BPTT ne može se paralelizirati pa vremenska složenost za sekvencu duljine N iznosi $O(N)$. Također, budući da stanja moraju biti pohranjena u memoriji za ažuriranje gradijenata, prostorna složenost ovog algoritma je također $O(N)$ [17]. Možemo zaključiti kako je takva mreža vrlo moćna u smislu postizanja rezultata, ali isto tako vrlo skupa u smislu vremena treniranja i zahtjeva na memoriju. Samo u slučaju problema kojim se bavi ovaj rad, N (duljina sekvence) može ići i do nekoliko tisuća za samo jednu sekvencu, dok broj sekvenci u skupu podataka prelazi i nekoliko desetaka tisuća.

Na taj način mogu se graditi razne arhitekture rekurentnih mreža. Jedan od primjera je već opisan u ovom potpoglavlju: rekurentna mreža koja računa izlaz u svakom koraku te ima rekurentne veze između srednjih slojeva. Međutim, umjesto rekurentne poveznice između srednjih slojeva susjednih stanja, možemo imati rekurentnu poveznicu između izlaza u prethodnom stanju i srednjeg sloja u trenutnom stanju. Osim rekurentnih poveznica, možemo dodavati i povezivati proizvoljan broj srednjih slojeva čime gradimo duboku rekurentnu mrežu. Zbog samopovezanosti srednjeg sloja, rekurentne mreže i sa samo jednim srednjim slojem mogu se promatrati kao primjeri dubokih neuronskih mreža. Detaljniji opisi tih arhitektura kao i mnoge druge arhitekture rekurentnih i ostalih dubokih neuronskih mreža mogu se naći u [11].

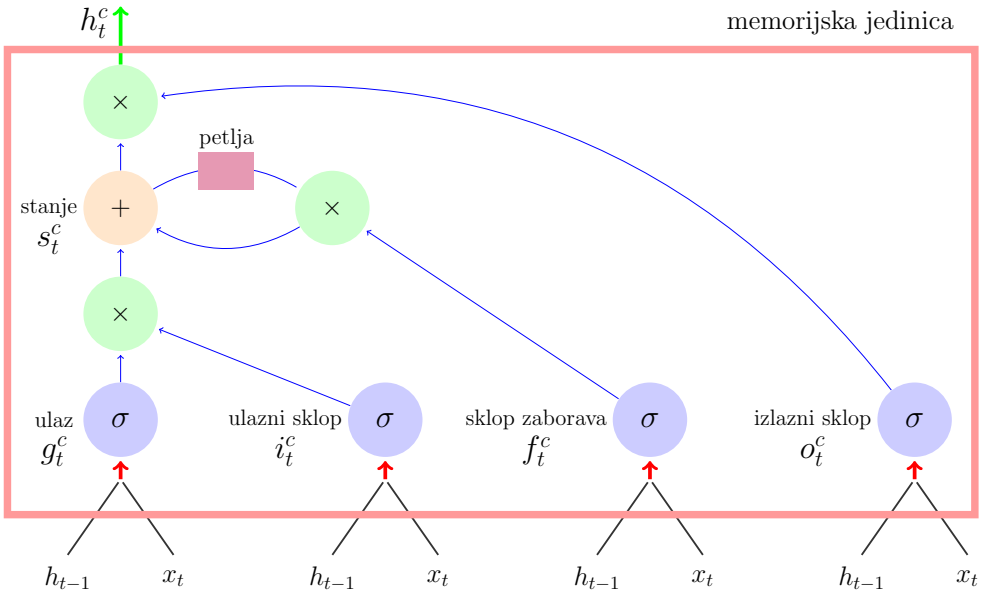
U svrhu ovog rada detaljnije ćemo proučiti dvije specijalne arhitekture rekurentnih mreža: *LSTM* (engl. *Long Short Term Memory Networks*) arhitekturu te dvosmjernu rekurentnu mrežu.

LSTM

Long Short Term Memory Networks (Hochreiter i Schmidhuber [13]) posebna je vrsta rekurentnih mreža s mogućnošću pamćenja informacija kroz dulji vremenski period. Iako bi u teoriji i obična rekurentna mreža trebala sačuvati informaciju iz prethodnog dijela sekvence na proizvoljno velikim udaljenostima (engl. *long-term dependencies*), u praksi se pokazalo drugačije. Uzrok takvom ponašanju poznat je pod pojmom nestajućeg (engl. *vanishing*) ili eksplodirajućega (engl. *exploding*) gradijenta. Budući da je glavni dio rekurentne mreže mnogostruka kompozicija iste funkcije, u slučaju računanja dugačkih sekvenci to uključuje množenje mnogostrukih jednakih Jakobijevih matrica. U slučaju jedne dimenzije radilo bi se o mnogostrukom množenju težine ω sa samom sobom. Za $|\omega| < 1$ taj bi umnožak konvergirao prema nuli te bismo imali problem nestajućega gradijenta, dok bi u slučaju $|\omega| > 1$ postizao iznimno velike vrijednosti te bismo govorili o slučaju eksplodirajućega gradijenta.

Da bi izbjegla problem nestajućeg i eksplodirajućega gradijenta, *LSTM* arhitektura (vidi sliku 2.5) sastoji se od posebno dizajniranoga rekurentnog čvora s fiksnim težinama koji nazivamo memorijska jedinica. Svi bridovi fiksne težine na slici 2.5 označeni su plavom bojom, dok su težine koje želimo optimizirati označene crvenom bojom. Memorijska jedinica je vrlo složene arhitekture, izgrađena od jednostavnih čvorova po posebnom građevnom modelu. U nastavku ćemo detaljno opisati svaki od tih čvorova s referenciranjem na sliku 2.5. Indeks c u vrijednostima na slici označava promatranu memorijsku jedinicu, dok indeks t kao i u prethodnom potpoglavlju označava korak, odnosno stanje ulazne sekvence $x_1, ..., x_N$.

Ulazni čvor na slici je označen kao čvor *ulaz* s vrijednosti g_t^c dobivenu primjenom odabrane aktivacijske funkcije σ na ulaz x_t trenutnog stanja i vrijednost srednjeg sloja h_{t-1} iz



Slika 2.5: Arhitektura memorijske jedinice *LSTM* mreže. Bridovi označeni plavom bojom su bridovi fiksne težine, dok su crvenom bojom označeni bridovi čije težine ažuriramo učenjem mreže. Zelenom bojom označen je izlazni brid. Čvorovi označeni s $\times$ označavaju množenje vrijednosti dok čvorovi označeni s + označavaju linearnu aktivaciju. Čvorovi označeni simbolom σ predstavljaju aktivacijske funkcije

prethodnog trenutka. Iako je na slici aktivacijska funkcija označena simbolom sigmoidalne funkcije (vidi 2.5), u većini slučajeva za aktivacijsku funkciju uzima se funkcija tangens hiperbolni.

Preostali čvorovi na slici označeni nazivom *sklop* su jedinice sa sigmoidalnom aktivacijskom funkcijom koji također aktivacijsku funkciju primjenjuju na ulaz x_t te vrijednost srednjeg sloja h_{t-1} iz prethodnog stanja. Sklopovi su inspirirani pojmom logičkih sklopova jer se vrijednost njihovog izlaza koristi za množenje s izlazom drugih čvorova čime filtriraju količinu informacije koju želimo sačuvati. Budući da je vrijednost sigmoidalne funkcije između 0 i 1, ako je vrijednost sklopa blizu 1, sačuvat će se sva informacija iz čvora s kojim se množi. S druge strane, ako je vrijednost sklopa blizu 0, vrijednost s kojom se množi će također biti blizu 0. Na taj način sva informacija iz čvora s kojim se množi će se izgubiti.

Na slici su označena tri sklopa: *ulazni*, *sklop zaborava* te *izlazni sklop*. Ulazni sklop s vrijednosti i_t^c množi se s vrijednosti ulaznog čvora te na taj način kontrolira količinu nove informacije iz ulaza koja će se sačuvati. Sklop zaborava s vrijednosti f_t^c kontrolira količinu informacije koja će se sačuvati iz trenutnog *stanja* memorijske jedinice. Stanje memorijske

jedinice s_t^c je jezgra svake memorijske jedinice te predstavlja čvor linearne aktivacije sa samopovezanim rekurentnim bridom i fiksnim težinama. Upravo se u toj petlji kao na slici 2.4 čuva informacija o prethodnom kontekstu kontrolirana sklopom zaborava. Tako bi vrijednost stanja dobili kao

$$s_t^c = g_t^c \cdot i_t^c + f_t^c \cdot s_{t-1}^c. \tag{2.15}$$

Za razliku od obične rekurentne mreže koja optimizira težine između samopovezanih čvorova, u *LSTM* arhitekturi takvi čvorovi su povezani fiksnim težinama čime se izbjegava problem nestajućeg i eksplodirajućega gradijenta. Konačno, izlazni sklop s vrijednosti o_t^c filtrira informaciju stanja koju ćemo dati kao rezultat h_t^c srednjeg sloja promatrane memorijske jedinice.

Ako sada s g^t, i^t, f^t, s^t, o^t označimo vektore vrijednosti ulaza, ulaznog sklopa, sklopa zaborava, stanja te izlaznog sklopa po svim memorijskim jedinicama *LSTM* mreže za trenutak t, $t = 1, ...N$ sekvence $x_1, ..., x_N$, te s W^g, W^f, W^i, W^o, b^g, b^i, b^f, b^o matrice težina i vektore parametara koje želimo optimizirati (na slici označene crvenom bojom) za ulazni čvor, ulazni sklop, sklop zaborava i izlazni sklop respektivno, dobivamo niz jednadžbi

$$\begin{aligned}
g_t &= \phi(W^g[x_t, h_{t-1}] + b^g) \\
i_t &= \sigma(W^i[x_t, h_{t-1}] + b^i) \\
f_t &= \sigma(W^f[x_t, h_{t-1}] + b^f) \\
o_t &= \sigma(W^o[x_t, h_{t-1}] + b^o) \\
s_t &= g_t \circ i_t + f_t \circ s_{t-1} \\
h_t &= \phi(s_t) \circ o_t,
\end{aligned}$$

gdje $\circ$ označava umnožak vektora po elementima, ϕ aktivacijsku funkciju, a h_t vrijednost skrivenog sloja u trenutku t.

LSTM arhitektura opisana u ovom potpoglavlju samo je jedna od mnogih koje se koriste, a neki od primjera mogu se pronaći u [17] i [11]. Također, možemo uočiti kako se u slučaju takvih arhitektura broj parametara koje treba optimizirati dodatno povećao u usporedbi s opisanom rekurentnom mrežom zbog čega su za treniranje takvih mreža potrebni dovoljno dobri hardverski resursi.

Iako je *LSTM* arhitektura posebno dizajnirana s ciljem eliminacije nestajućeg i eksplodirajućega gradijenta, pokazalo se da sama promjena arhitekture nije dovoljna da bi se osiguralo postizanje globalnog minimuma. Štoviše, Blum i Rivest su dokazali kako je optimizacija mreže sa samo jednim slojem NP-težak problem [6]. Ipak, dobro dizajnirane arhitekture i implementacije gradijentnog spusta omogućavaju da treniranje *LSTM* rekurentne mreže daju zadovoljavajuće rezultate [17].

Dvosmjerna rekurentna mreža

Već na početku rada spominjali smo pojam konteksta dijela sekvence kao fiksnu količinu informacije koja se nalazi u okolini tog dijela. Dakle, zanima nas informacija koja prethodi, ali i koja slijedi nakon promatranog dijela sekvence. Međutim, u opisanim arhitekturama rekurentnih mreža, rekurzivna petlja za očuvanje konteksta odnosi se samo na prethodno viđeni dio sekvence što smo, u slučaju *LSTM*, postigli jednadžbom 2.15 te ulazom h_{t-1} u memorijskoj jedinici. Budući da nam je u mnogim primjenama, a modeliranje proteinskih sekvenci je samo jedna od njih, nužna i informacija o budućem kontekstu, potrebno je bilo osmisliti arhitekturu rekurentne mreže koja će uhvatiti i tu informaciju. Upravo je dvosmjerna rekurentna mreža (engl. *Bidirectional Recurrent Neural Network*) (Schuster i Paliwal [24]) dizajnirana kako bi riješila taj problem. Dakle, kao što joj samo ime govori, takva mreža trenira se podacima u oba smjera kako bi uhvatila prethodni, ali i budući kontekst. Informaciju o prethodnom dijelu za trenutak t dobiva rekurzivno od $t-1$ prema 1, dok informaciju o sljedećem dijelu dobiva od $t+1$ do N za ulaznu sekvencu $x_1, ..., x_N$.

Iz tog razloga, ta se arhitektura sastoji od dva sloja skrivenih čvorova povezanih s ulazom i izlazom. Vrijednost prvog sloja bismo dobili analogno kao u 2.12:

$$h_t = \sigma\left(W^{hx}x_t + W^{hh}h_{t-1} + b_h\right), \tag{2.16}$$

dok bi vrijednost drugog sloja dobili na isti način uz promjenu smjera kretanja po sekvenci:

$$z_t = \sigma\left(W^{zx}x_t + W^{zz}z_{t+1} + b_z\right). \tag{2.17}$$

Konačno, izlaznu vrijednost y' bismo dobili dodavanjem vrijednosti z_t u jednadžbi 2.17:

$$y'_t = f\left(W^{yh}h_t + W^{yz}z_t + b_y\right), \tag{2.18}$$

gdje su, analogno, W^{yz}, W^{zx}, W^{zz} odgovarajuće matrice težina, a f izlazna funkcija[17]. Možemo uočiti kako broj težina koje treba optimizirati u ovom slučaju odgovara dvostrukom broju težina za odabran model rekurentne mreže.

Model korišten u radu

U svrhu ovog rada koristit će se varijanta rekurentne mreže koja se sastoji od *dvosmjerne LSTM rekurentne mreže*. *LSTM* arhitektura se koristi kako bi se sačuvala informacija na velikim udaljenostima i izbjegao problem nestajućega gradijenta, dok se arhitektura dvosmjerne rekurentne mreže koristi kako bi se koristila informacija o prethodnom i sljedećem dijelu sekvence. Zbog načina na koji protein međudjeluje u prostoru u svojoj tercijarnoj i kvaternoj strukturi (vidi 1.3, a za detalje [19]), određeni dio proteinske sekvence nije blisko povezan samo sa susjednim aminokiselinama, već i s mnogim udaljenim dijelovima koji su se, zbog spiralnih oblika, našli blizu u prostoru. Budući da je cilj ovog

rada dobiti vektorsku reprezentaciju proteina koja će generalno davati dobre rezultate, u toj reprezentaciji želimo uključiti informaciju o globalnoj strukturi. Dvosmjerna *LSTM* rekurentna mreža će nam upravo to omogućiti.

Poglavlje 3

Modeliranje reprezentacija proteina

Nakon objašnjene teorijske pozadine neuronskih mreža koje se koriste u ovom radu, u ovom poglavlju ćemo u koracima objasniti na koji način smo modelirali reprezentacije proteina uz pomoć rekurentnih neuronskih mreža. Poglavlje započinje navođenjem radova koji se bave modeliranjem reprezentacija proteina.

3.1 Prijašnji radovi

U ovom potpoglavlju bit će navedeni i ukratko objašnjeni prijašnji radovi s tematikom reprezentacije proteinske sekvence koji su poslužili kao motivacija za razvoj ovog rada.

Glavna ideja traženja reprezentacije proteinskih sekvenci rezultat je rada [2]. U tom radu vektorska reprezentacija proteina dobivena je korištenjem *word2vec* modela sa *Skip-Gram* arhitekturom. Dobivene reprezentacije autori rada su koristili za klasifikacijski problem određivanja familije proteina te su dobili bolju točnost od do tada postignute.

Za treniranje *word2vec* modela koristila se Swiss-Prot [7] baza podataka koja se sastoji od 546 790 proteinskih sekvenci. Kako je spomenuto u potpoglavlju 2.2, da bi se na taj problem primijenio model *word2vec* potrebno je definirati građevnu jedinicu koja će odgovarati riječi u kontekstu. U ovom radu, taj pristup odgovara podjeli sekvence na trigrame. Međutim, autori originalnog rada nisu koristili standardnu podjelu na preklapajuće ili nepreklapajuće trigrame opisane u potpoglavlju 2.2, već su sekvencu preslikali u tri sekvence trigrama na način prikazan u primjeru 3.1.1.

Primjer 3.1.1. *Kao u primjeru 2.2.2 neka je dana proteinska sekvenca:*

$$ANTVAXWFMASNQAT$$

Tada sekvencu dijelimo na nepreklapajuće trigrame uz pomak za jedan kako bismo dobili tri različite sekvence trigrama:

1. *ANT VAX WFM ASN QAT*

2. *NTV AXW FMA SNQ*

3. *TVA XWF MAS NQA.*

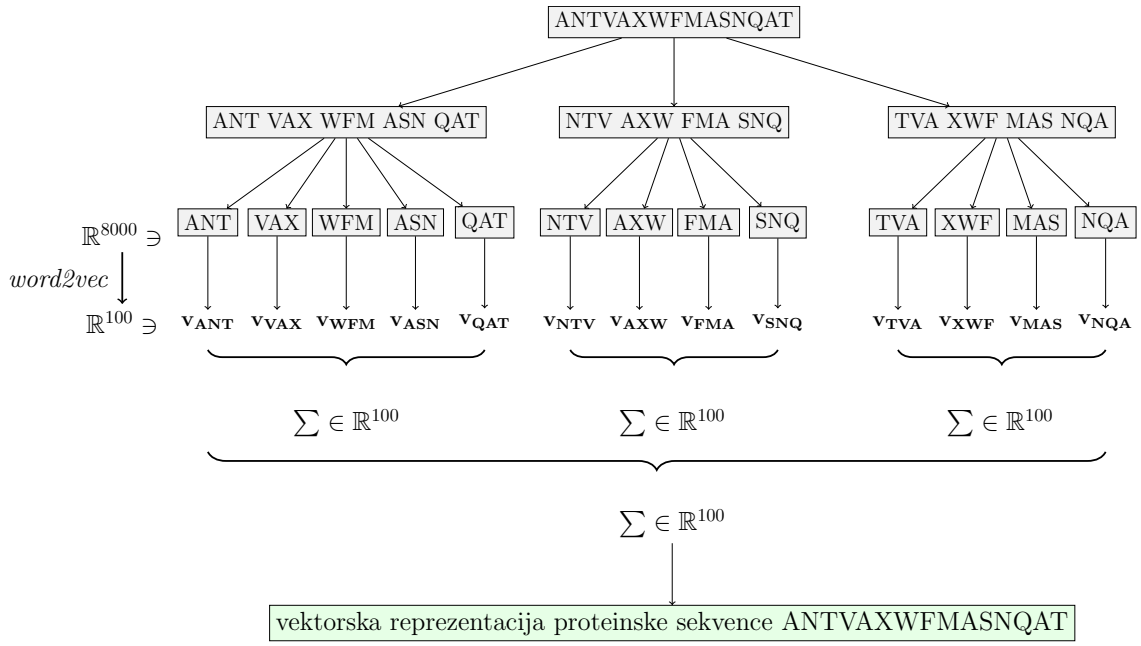
Možemo uočiti da se takvom podjelom dobiva kombinacija preklapajuće i nepreklapajuće podjele. U smislu preklapajuće podjele uključene su sve kombinacije trigrama čime se maksimalno uključuje informacija o lokalnoj strukturi. S druge strane, u smislu nepreklapajuće podjele, svaka od tri novodobivene sekvence zadržava informaciju o originalnom redoslijedu sekvence.

Dakle, nakon takvog pretprocesiranja, dobiven je tri puta veći skup sekvenci trigrama (1 640 370 sekvenci trigrama) koji se koristio kao skup podataka za treniranje *word2vec* modela uz definirani kontekst od 25 riječi sa svake strane te dimenziju 100 srednjeg sloja neuronske mreže. Budući da je veličina rječnika trigrama 8000, ulaz za neuronsku mrežu su trigrami prikazani 8000-dimenzionalnim bit-vektorom. Nakon istreniranog modela, za svaki trigram je dobivena stodimenzionalna vektorska reprezentacija. Da bismo iz vektorskih reprezentacija trigrama dobili vektorsku reprezentaciju početne proteinske sekvence, autori originalnog rada su se odlučili na pristup koji sumira reprezentacije svih trigrama kroz sve tri sekvence trigrama. Tada reprezentaciju proteinske sekvence iz primjera 3.1.1 dobivaju na način prikazan na slici 3.1 .

Dakle, možemo uočiti kako je tim pristupom svaka proteinska sekvenca reprezentirana vektorom dimenzije 100 zbog čega svaki skup protenskih sekvenci od N elemenata možemo modelirati matricom $N \times 100$. Kako bismo provjerili kvalitetu dobivenih reprezentacija, odlučili su koristiti te reprezentacije na problemu klasifikacije proteina u familije. Budući da je problem sveden na vektore fiksnih dimenzija, mogli su upotrijebiti bilo koji standardni algoritam strojnog učenja pa su se, u radu [2], odlučili na *SVM* algoritam (vidi [25]). Dobivena prosječna točnost na 7027 familija bila je 93 ± 0.06 %.

Glavna motivacija za razvoj ovog rada je bio korak dobivanja reprezentacije proteinske sekvence iz reprezentacija trigrama. Zbog komutativnosti sume, smatramo da se pristupom koji sumira sve reprezentacije trigrama gubi informacija o globalnoj strukturi sekvence. Upravo je ta informacija iznimno bitna za dobivanje reprezentacije sekvence koja bi davala dobre rezultate na svim problemima, a ne samo na problemu predviđanja proteinske familije. Naime, sumiranjem dobivamo isti vektor za primjerice *ANT WAX VFM* i *WAX VFM ANT* što može biti veliko ograničenje za postizanje generalizacije. Već i mala permutacija u slučaju proteinskih sekvenci znači iznimno veliku promjenu u preostalim strukturama proteina (sekundarnoj, tercijarnoj i kvaternoj), a onda i u konačnom proteinu (vidi potpoglavlje 1.3).

Iz tog razloga, u ovom radu smo se odlučili promijeniti pristup dobivanju reprezentacije sekvence iz reprezentacija trigrama te koristiti pristup koji će, za razliku od sume,



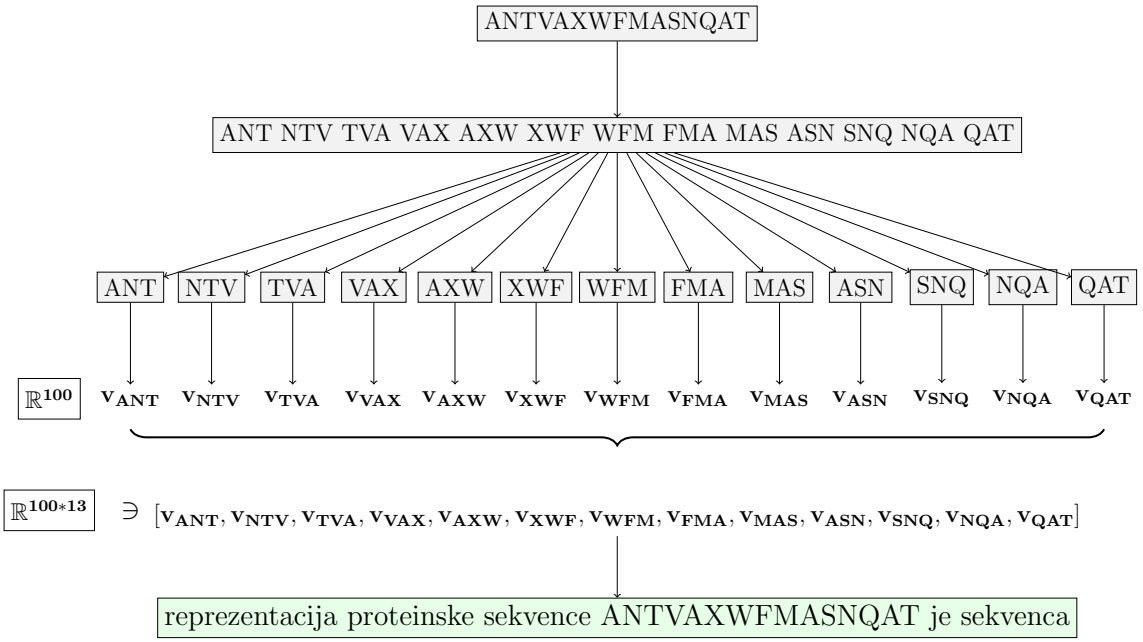
Slika 3.1: Primjer dobivanja vektorske reprezentacije proteinske sekvence iz vektorskih reprezentacija trigrama

omogućiti čuvanje informacije o globalnoj strukturi u dobivenoj reprezentaciji. Ideja je takvim pristupom dobiti bolje rezultate nad problemom predviđanja familija od rezultata postignutih u ovom radu. Osim toga, konačni cilj nam je dobiti dobre rezultate i na ostalim problemima koji koriste infomaciju o primarnoj strukturi proteinske sekvence čime bismo dobili dovoljno generalne reprezentacije za rješavanje mnogih problema iz tog područja.

Da bi naši rezultati bili usporedivi s navedenim radom, prvi zadatak u našem radu je imitirati zadatak ovog rada da bismo potvrdili sve dobivene rezultate.

Nakon objave opisanog rada, sve više autora počelo se baviti rješavanjem ovog problema pomoću korespondencije s obradom prirodnog jezika. Kako su u području modeliranja prirodnog jezika nastajala nova saznanja i otkrića, tako su ih mnogi znanstvenici počeli primjenjivati za dobivanje boljih rezultata nad proteinskim sekvencama. U potpoglavlju 2.2 spomenuti su rezultati u kojima vektorske reprezentacije riječi dobivene treniranjem *Glove* modela daju bolje rezultate od reprezentacija riječi dobivenih treniranjem *word2vec* modela. Upravo je taj rezultat motivirao autore sa Stanford sveučilišta (vidi [16]) da imitiraju postupak rada iz prethodnog dijela ovog potpoglavlja, ali umjesto da reprezentacije trigrama dobiju iz *word2vec* modela, odlučili su se za *Glove* model.

Ključna razlika ovog rada u odnosu na rad [2] je u tome da se nisu bavili problemom dobivanja vektorske reprezentacije proteinske sekvence iz reprezentacije trigrama. Umjesto toga, nakon dobivenih reprezentacija trigrama, autori rada [16] odlučili su proteinsku sekvencu promatrati kao sekvencu vektorskih reprezentacija trigrama nakon razdvajanja na preklapajuće trigrame. Tada bi reprezentaciju proteinske sekvence iz primjera 2.2.2 mogli dobiti na način prikazan na slici 3.2.



Slika 3.2: Primjer dobivanja vektorske reprezentacije proteinske sekvence iz vektorskih reprezentacija trigrama

Uočimo da, koristeći opisani pristup, reprezentacija sekvence više nije vektor fiksne dimenzije, nego ovisi o broju aminokiselina u početnoj sekvenci. Nad takvim reprezentacijama, za rješavanje problema kao što su predviđanje familija proteina ne možemo koristiti standardne algoritme strojnog učenja kao što su *SVM* ili rekurentne neuronske mreže, nego se takvi problemi rješavaju pristupom učenja iz sekvenci opisanog u 2.3. Takvim pristupom, uspjeli su dobiti bolje rezultate od pristupa koji uključuje samo sumiranje trigrama.

Ipak, možemo uočiti da reprezentacija sekvence u obliku sekvence vektora visokih dimenzija nije praktična niti u smislu memorije niti u smislu efikasnosti. Već za pohranu samo jedne sekvence duljine 1000, potreban nam je realni vektor dimenzije 1000×100. Osim toga, pristup učenja iz sekvenci je iznimno kompleksan te zahtijeva korištenje većih hardverskih resursa koji nisu uvijek dostupni. Zbog toga nije nimalo praktično svaki problem rješavati pomoću pristupa učenja iz sekvenci, jer su mnogi problemi dovoljno jed-

nostavni da se mogu efikasno riješiti i uz pomoć jednostavnijih metoda strojnog učenja kao što su stabla odluke ili čak logistička regresija.

Pristup koji smo se odlučili koristiti u ovom radu je kombinacija tih dvaju radova. Najprije ćemo, kao u radu [2], tražiti reprezentacije trigrama. Nakon toga ćemo, kao u radu [16] svaku proteinsku sekvencu reprezentirati sekvencom reprezentacija trigrama. U ovom trenutku ćemo uključiti dodatan korak kojim ćemo iz sekvence vektora dobiti vektor fiksne dimenzije. Taj vektor ćemo koristiti kao reprezentaciju početne proteinske sekvence. Dodatan korak modelirat ćemo pomoću metode učenja iz sekvenci opisane u potpoglavlju 2.3.

3.2 Korak 1: Reprezentacija trigrama

U sljedećim potpogljavljima će u koracima biti opisan način na koji smo dobili različite stodimenzionalne realne reprezentacije proteinskih sekvenci.

Prvi korak u modeliranju vektorske reprezentacije proteina je modeliranje vektorske reprezentacije trigrama. U radu smo koristili dvije različite reprezentacije trigrama koje smo dobili metodama *word2vec* i *Glove* (vidi potpoglavlje 2.2) te ćemo, ovisno o korištenom modelu, te reprezentacije nadalje nazivati *word2vec*, odnosno *Glove* reprezentacije. Budući da nam je jedan od ciljeva usporediti rezultate s radom [2], za dobivanje reprezentacija trigrama smo odlučili koristiti isti skup podataka korišten u tom radu, ali i istu dimenziju vektora i parametre. Korištena baza podataka poznata je pod nazivom *Swiss-Prot* (vidi [7]), a sastoji se od 546 790 proteinskih sekvenci. Za dimenziju vektorske reprezentacije trigrama odabrali smo 100, dok smo za parametar konteksta u definiciji oba modela odabrali 25.

Kako je već objašnjeno u potpoglavlju 3.1, iz početnog skupa proteinskih sekvenci, koristeći nepreklapajuću metodu uz pomak (vidi primjer 3.1.1), dobili smo tri puta veći skup sekvenci trigrama. Sada smo, uz već spomenutu korespondenciju

$$\begin{aligned} \textit{slovo} &\Leftrightarrow \textit{aminokiselina} \\ \textit{riječ} &\Leftrightarrow \textit{trigram} \\ \textit{rečenica} &\Leftrightarrow \textit{sekvenca trigrama} \\ \textit{dokument} &\Leftrightarrow \textit{skup sekvenca trigrama} \end{aligned}$$

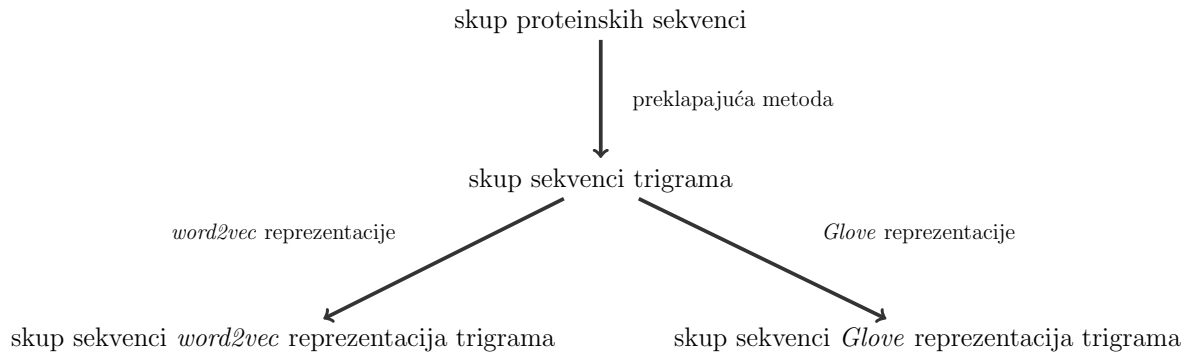
dobiveni skup koristili za treniranje dva različita modela: *word2vec* i *Glove*. Istrenirane modele koristimo za dobivanje reprezentacije trigrama na način da za svaki trigram iz skupa trigrama aminokiselina dobivamo dva vektora dimenzije 100, *word2vec* te *Glove* vektor. Za implementaciju *word2vec* korišten je *Gensim* paket [1], dok je za *Glove* implemen-

[1] Preuzeto s https://radimrehurek.com/gensim/index.html, 12/2016

taciju korišten *GloVe* paket [2]. Budući da skup trigrama aminokiselina sadrži 8000 elemenata, reprezentacije trigrama možemo jednostavno pohraniti u obliku matrice $\in \mathbb{R}^{8000\times100}$.

3.3 Korak 2: Pretprocesiranje skupa proteinskih sekvenci

Nakon dobivanja informativnih reprezentacija trigrama, možemo reći da već imamo informativnu reprezentaciju proteinske sekvence. Umjesto da proteinsku sekvencu promatramo kao niz aminokiselina, kao u radu [16] sekvencu aminokiselina možemo preklapajućom metodom (vidi primjer 2.2.2) podijeliti na sekvencu trigrama. Nakon toga, umjesto da promatramo trigrame kao stringove, možemo promatrati vektorske reprezentacije trigrama. Tako bismo, kao na slici 3.2 svaku proteinsku sekvencu mogli preslikati u sekvencu stodimenzionalnih vektora. Zbog toga će svaki skup proteinskih sekvenci koji ćemo promatrati u nastavku rada biti preslikan u dva različita skupa sekvenci realnih vektora po pravilu prikazanom na slici 3.3. S obzirom na korištenu *word2vec* ili *Glove* reprezentaciju vektora, u nastavku ćemo skupove sekvenci stodimenzionalnih vektora dobivene preslikavanjem proteinskih sekvenci nazivati *word2vec*, odnosno *Glove*.



Slika 3.3: Pretprocesiranje skupa proteinskih sekvenci u dva različita skupa sekvenci realnih vektora: *word2vec* i *Glove* skup sekvenci vektora

3.4 Korak 3: Treniranje rekurentne mreže

U ovom koraku promatramo reprezentaciju proteinske sekvence u obliku sekvence stodimenzionalnih vektora. Često ćemo, zbog jednostavnijeg izražavanja, umjesto sekvenca

[2]Preuzeto s https://github.com/stanfordnlp/GloVe, 12/2016

vektorskih reprezentacija trigrama govoriti samo o sekvenci trigrama, ali pritom misleći na sekvencu odgovarajućih reprezentacija. Dakle, kako je već objašnjeno u potpoglavlju 3.1, takva reprezentacija može biti vektor dimenzije veće od 10^5. Osim toga, zbog korištenja metoda *word2vec* i *Glove* koji treniranjem uče samo lokalni kontekst, reprezentacije trigrama kodiraju informaciju samo o lokalnoj strukturi proteinske sekvence. Da bismo u reprezentaciji čitave sekvence uključili i informaciju o globalnim interakcijama, skup sekvenci vektora ćemo uključiti u dodatnu fazu treniranja uz pomoć dvosmjerne *LSTM* rekurentne mreže (vidi 2.3). Dodatna prednost takvog pristupa je da, analogno kao u slučaju čitanja reprezentacija trigrama, aktiviranjem težina iz rekuretne mreže možemo dobiti reprezentaciju sekvence fiksne dimenzije. U neuronskim mrežama u slučaju *word2vec*, dimenzija reprezentacije je odgovarala broju čvorova u srednjem sloju, dok u slučaju *LSTM* rekurentne mreže ona odgovara broju memorijskih jedinica (vidi 2.3, dio *LSTM*) srednjeg sloja. Tako dobivena reprezentacija kodira informaciju o lokalnoj strukturi koja joj je dana na ulazu, o globalnoj strukturi uz pomoć *LSTM* rekurentne mreže te o prošlom i budućem lokalnom i globalnom kontekstu zbog dvosmjernosti rekurentne mreže.

Definiranje zadatka rekurentne mreže

Da bismo rekurentnu mrežu trenirali, potrebno je definirati zadatak s kojim ćemo je trenirati. Kako je već spomenuto u potpoglavlju 2.3, u ovom radu za metodu učenja iz sekvenci koristi se problem klasifikacije sekvenci. Dakle, želimo da rekurentna mreža u skupu proteinskih sekvenci nauči raspoznati odgovarajuće klase. Već smo u nekoliko navrata spomenuli kako nam je od interesa u reprezentaciji uključiti što više informacija o strukturi proteina, zbog čega nam je glavni zahtjev na zadatak bio da mora omogućiti mreži raspoznati takvu strukturu. Osim toga, zahtijevali smo i da skup podataka s kojim ćemo trenirati mrežu bude što veći. Iz tog razloga smo zadatak definirali kao problem raspoznavanja prave od permutirane sekvence. Takav umjetni problem omogućit će da od proizvoljno velikog skupa proteinskih sekvenci na brz način dobijemo labelirani skup. U stvarnim problemima, zbog dugotrajnosti labeliranja, samo mali dio sekvenci je označen. Svaku sekvencu trigrama koja odgovara proteinu smo, permutiranjem trigrama na način koji će biti objašnjen u nastavku, izmijenili u sekvencu trigrama čija struktura više ne odgovara proteinu. Originalnu sekvencu smo označili labelom *protein*, dok smo permutiranu sekvencu označili labelom *permutacija*. Učenjem raspoznavanja prave od permutirane sekvence na temelju sekvenca vektorskih reprezentacija trigrama, rekurentna mreža u težinama kodira informaciju o svim dijelovima strukture.

Radi usporedbe s rezultatima rada [2] (vidi 3.1) u kojemu je konačna reprezentacija proteinske sekvence dobivena sumiranjem svih reprezentacija trigrama, odlučili smo koristiti istu dimenziju i za reprezentaciju proteinske sekvence. Budući da je dimenzija reprezentacije trigrama 100, slijedi da je dimenzija reprezentacije proteina također 100. Zbog

toga smo odabrali arhitekturu *LSTM* mreže sa 100 memorijskih jedinica čime je dimenzija reprezentacija dobivenih pomoću takve mreže također dimenzije 100.

Zbog kompleksnosti arhitekture rekurentne mreže treba biti oprezan u odabiru zadatka kako se ne bi dogodio problem *pretreniranja*. U tom problemu mreža bi naučila raspoznati primjere na skupu za treniranje, ali bi izgubila mogućnost generalizacije na novim primjerima. Već sama činjenica da mreža mora raspoznati samo dvije klase znači da permutirana sekvenca mora biti dovoljno teško razlikovana od originalne sekvence. Iz tog razloga smo prije odabira načina permutacije odlučili analizirati skup podataka s kojim ćemo trenirati mrežu radi uočavanja pravilnosti koje bismo trebali uzeti u obzir prilikom permutiranja.

Skup proteinskih sekvenci

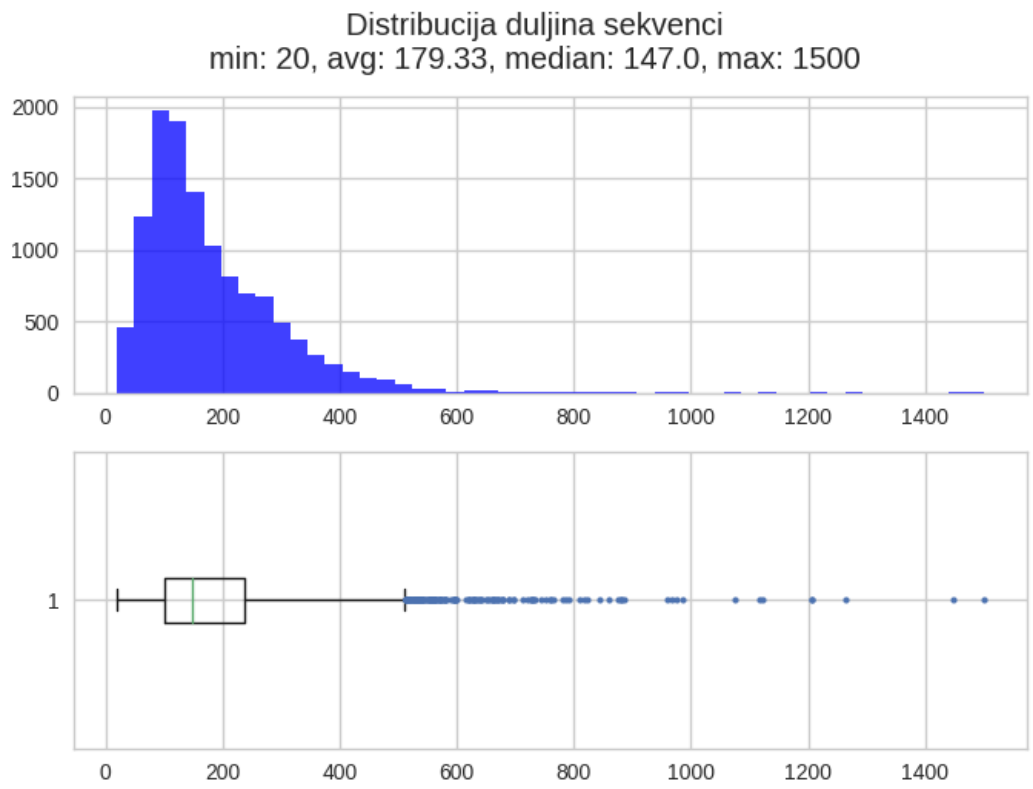
Skup koji smo odabrali je *UniProt* baza proteinskih sekvenci s filtriranim sekvencama tako da je sačuvana *mjera sličnosti* u iznosu od 35 %. Mjera sličnosti proteinskih sekvenci kao i preostali detalji ove baze mogu se pronaći u [7]. Taj skup sastoji se od 12 052 proteinskih sekvenci s duljinama sekvence od 20 do 1500 u smislu broja aminokiselina (vidi sliku 3.4). Kako je objašnjeno u drugom koraku, (vidi 3.3), u ovom trenutku umjesto proteinskih sekvenci promatramo sekvence vektorskih reprezentacija trigrama. Budući da smo sekvencu trigrama dobili preklapajućom metodom, duljina sekvence trigrama odgovara duljini sekvence aminokiselina umanjenoj za dva, zbog čega duljina najdulje sekvence trigrama u promatranom skupu iznosi 1498.

Na slici 3.5 prikazan je histrogram na kojemu možemo vidjeti koliko sekvenci iz skupa započinje ili završava s određenom aminokiselinom. Možemo uočiti da gotovo jedna petina proteinskih sekvenci započinje aminokiselinom *M* dok vrlo mali broj završava aminokiselinama *C*, *M* ili *W*.

Napomena 3.4.1. *U svrhu dobivanja reprezentacija proteina potrebne su dvije faze treniranja: u prvoj fazi treniranja dobivamo model koji nam daje reprezentaciju trigrama. U drugoj fazi treniranja dobivamo model koji nam daje reprezentaciju sekvence. Obje faze su međusobno nezavisne i ne moraju se trenirati korištenjem istog skupa podataka.*

Permutiranje sekvence trigrama

Zbog uočenih pravilnosti na početku i kraju sekvence, u sekvenci trigrama smo prije permutiranja fiksirali prva tri i zadnja tri trigrama. Kako dio koji permutiramo ne bi bio prekratak, sekvence duljine manje od 40 nismo uključili u permutiranje zbog čega će skupovi za treniranje sadržavati 12 052 primjera iz klase *proteina* te 11 707 primjera iz klase *permutacije*. Kako bismo potvrdili da je takav problem dovoljno dobar za rekurentnu mrežu, odlučili smo definirati tri različita načina permutiranja te konačno usporediti njihove re-



Slika 3.4: Distribucija duljina sekvenci u skupu za treniranje rekurentne mreže

zultate. Permutacije ćemo objasniti u obliku primjera na sekvencama trigrama, a u slučaju promatranog skupa radi se o vektorskim reprezentacijama trigrama.

Prva permutacija, koju ćemo na slikama označavati s *perm1*, slučajna je permutacija trigrama unutar nefiksiranog dijela prikazana u primjeru 3.4.2.

Primjer 3.4.2. *Pretpostavimo da nam je dana sekvenca trigrama*

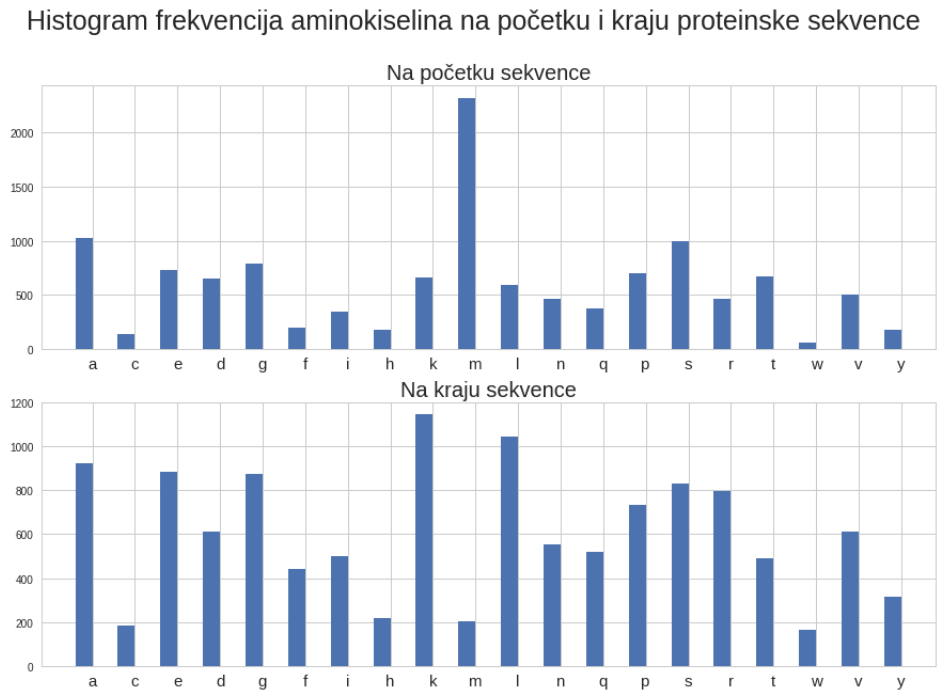
AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR DRR RRA RAF.

Nakon fiksiranja prva i zadnja tri trigrama:

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR DRR RRA RAF,

permutiranu sekvencu dobivamo slučajnom permutacijom trigrama od SAE do YDR:

AFT FTS TSA DVL YDR AED LKE VLK SAE KEY EYD EDV DRR RRA RAF.



Slika 3.5: Distribucija frekvencija aminokiselina na početku i na kraju sekvence u skupu za treniranje rekurentne mreže

Druga permutacija, na slikama označena s *perm2*, slučajna je permutacija trigrama unutar prozora veličine pet nefiksiranog dijela, a prikazana je u primjeru 3.4.3. Dakle, nefiksirani dio dijelimo na prozore s pet trigrama, nakon čega permutiramo trigrame unutar svakog prozora slučajnom permutacijom.

Primjer 3.4.3. *Pretpostavimo da nam je dana sekvenca trigrama*

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR LMN MLK LKK KKS KSD SDR DRR RRA RAF.

Nakon fiksiranja prva i zadnja tri trigrama te podjele nefiksiranog dijela na prozore veličine pet:

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR LMN MLK LKK KKS KSD SDR DRR RRA RAF,

permutiranu sekvencu dobivamo slučajnom permutacijom trigrama unutar svakog od prozora:

AFT FTS TSA SAE EDV VLK DVL AED KEY LKE EYD LMN YDR LKK SDR MLK KSD KKS DRR RRA RAF.

Konačno, treća permutacija, na slikama označena s *perm3*, slučajna je permutacija trigrama između prozora veličine pet nefiksiranog dijela, a prikazana je u primjeru 3.4.4. Dakle, nefiksirani dio dijelimo na prozore s pet trigrama, nakon čega unutar svakog prozora odabiremo slučajan trigram te, na kraju, odabrane trigrame slučajno permutiramo.

Primjer 3.4.4. *Pretpostavimo da nam je dana sekvenca trigrama*

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR LMN MLK LKK KKS KSD SDR DRR RRA RAF.

Nakon fiksiranja prva i zadnja tri trigrama, podjele nefiksiranog djela na prozore veličine pet:

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR LMN MLK LKK KKS KSD SDR DRR RRA RAF,

te odabira slučajnih trigrama unutar prozora:

AFT FTS TSA SAE AED EDV DVL VLK LKE KEY EYD YDR LMN MLK LKK KKS KSD SDR DRR RRA RAF,

permutiranu sekvencu dobivamo slučajnom permutacijom odabranih trigrama:

AFT FTS TSA SAE MLK EDV DVL VLK LKE KEY EYD AED LMN YDR LKK KKS KSD SDR DRR RRA RAF.

Možemo uočiti kako se prvom permutacijom permutirana sekvenca najviše mijenja u odnosu na originalnu zbog čega očekujemo da će taj problem rekurentna mreža najlakše riješiti. Za drugu i treću permutaciju očekujemo da će biti teže raspoznati radi li se o originalnoj ili permutiranoj sekvenci. Zbog permutiranja trigrama isključivo unutar prozora, druga permutacija u permutiranoj sekvenci uspijeva sačuvati ekvivalentnu globalnu strukturu, dok treća permutacija, zbog fiksiranja više od 80 % trigrama uspijeva sačuvati gotovo ekvivalentnu lokalnu strukturu.

Nadalje, možemo uočiti kako ćemo pomoću tri različita načina permutiranja i dvije različite reprezentacije trigrama dobiti šest različitih skupova za treniranje. Skup dobiven korištenjem *word2vec* reprezentacija trigrama uz prvi način permutiranja nazivat ćemo *word2vec perm1* skup, a analogno i preostale skupove, *word2vec perm2*, *word2vec perm3*, *Glove perm1*, *Glove perm2*, *Glove perm3*. Svih šest skupova sastoje se od 12 052 sekvence vektorskih reprezentacija trigrama klase *protein* te 11 707 sekvenci vektorskih reprezentacija trigrama klase *permutacija*. U svakom od skupova odvojili smo istih 20 % proteinskih sekvenci za validaciju modela tako da se konačno svaki od skupova za treniranje sastoji od 19 008 primjera za učenje i 4751 primjer za validaciju.

Konačno, nakon treniranja rekurentne mreže šest puta, za svaku proteinsku sekvencu ćemo dobiti šest različitih reprezentacija. Uspješnost dobivenih reprezentacija mjerit ćemo eksperimentalno u poglavlju 5.

Parametri rekurentne mreže

Na početku ovog potpoglavlja i u potpoglavlju 2.3 rečeno je kako je u svrhu ovog rada korištena dvosmjerna *LSTM* rekurentna mreža s jednim slojem od 100 memorijskih jedinica. Za aktivacijsku funkciju ulaza (vidi sliku 2.5) najbolje rezultate na skupu za treniranje je dala funkcija tangens hiperbolni, dok smo za ostale aktivacije koristili sigmoidalnu funkciju (vidi 2.5). Sve modele trenirali smo kroz 50 *epoha* tako da smo bilježili rezultate modela nakon 10, 20, 30, 40 i 50 epoha. Pod epohom smatramo jedan prolaz kroz cijeli skup za treniranje tako da svaka epoha nastavlja s optimizacijom od prethodne epohe. Posebno smo pratili svaku desetu epohu kako bismo kontrolirali treniranje radi izbjegavanja pretreniranja. Od ostalih parametara koristili smo veličinu *batch*a za računanje gradijenta od 128 primjera, stopu učenja (engl. *learning rate*) od 0.0001 te vrijednost *Dropout*a od 0.3.

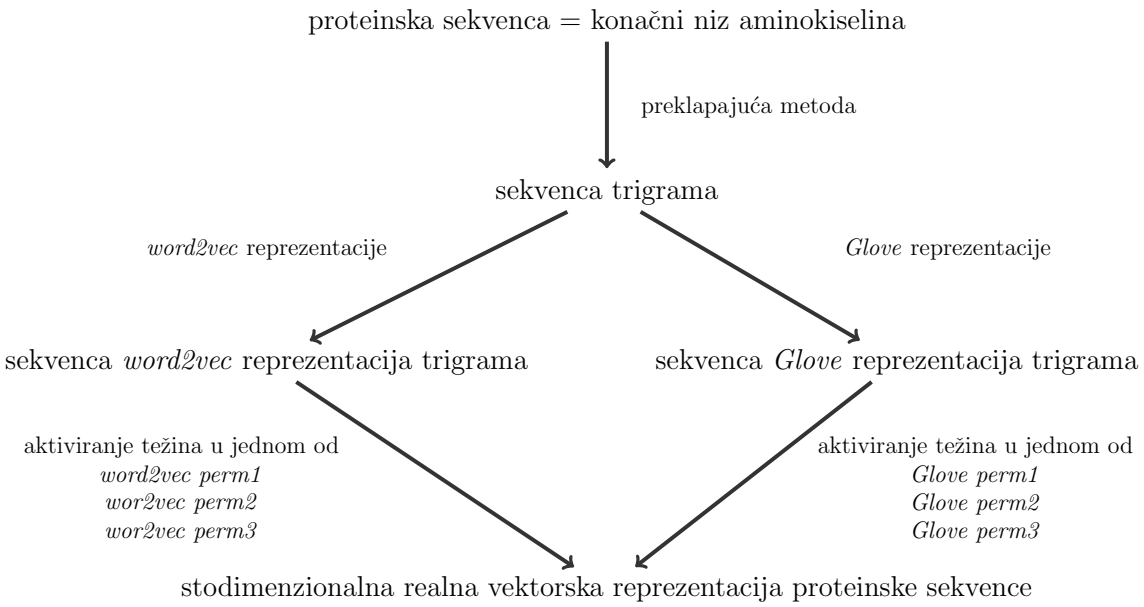
Za implementaciju rekurentne mreže korišten je *Keras* paket s *RMSprop* optimizatorom[3], a za treniranje je korišten server s GTX 980 grafičkom karticom na Institutu Ruđer Bošković. Za učenje svih modela kroz 50 epoha bilo je potrebno oko 22 sata.

3.5 Korak 4: Dobivanje reprezentacije proteina iz modela

U zadnjem koraku objasnit ćemo način dobivanja vektorske reprezentacije sekvence iz istreniranih modela. Za razliku od vektorskih reprezentacija trigrama, u kojima promatramo konačan prostor od 8000 trigrama, u slučaju proteinskih sekvenci radi se o proizvoljno velikom prostoru. Proteinske sekvence mogu biti proizvoljne duljine zbog čega vektorske reprezentacije proteinskih sekvenci nećemo pohranjivati, nego ćemo ih uvijek *pročitati* iz modela. Za proizvoljnu proteinsku sekvencu reprezentiranu nizom aminokiselina, realnu vektorsku reprezentaciju možemo dobiti kao na slici 3.6. Najprije je potrebno odabrati jedan od šest modela *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *Glove perm1*, *Glove perm2* ili *Glove perm3*. Nakon toga, proteinsku sekvencu preslikavamo u sekvencu trigrama. U slučaju da smo odabrali model koji je koristio *word2vec* reprezentacije trigrama, sekvencu trigrama preslikavamo u sekvencu *word2vec* reprezentacija trigrama, odnosno *Glove* reprezentacija, u slučaju da smo odabrali model koji je koristio *Glove* reprezentacije. Konačno, dobivenu sekvencu realnih vektora množimo s težinama odabranoga rekurentnog modela (aktiviramo težine rekurentnog modela) te dobiveni stodimenzionalni vektor koristimo kao konačnu vektorsku reprezentaciju sekvence.

Zadnji dio rada odnosi se na testiranje uspješnosti dobivenih reprezentacija, a obrađen je u poglavlju 5.

[3] Vidi https://keras.io/, 04/2017



Slika 3.6: Način dobivanja stodimenzionalne realne vektorske reprezentacije proteinske sekvence iz sekvence aminokiselina

Poglavlje 4

Evaluacija reprezentacija proteina

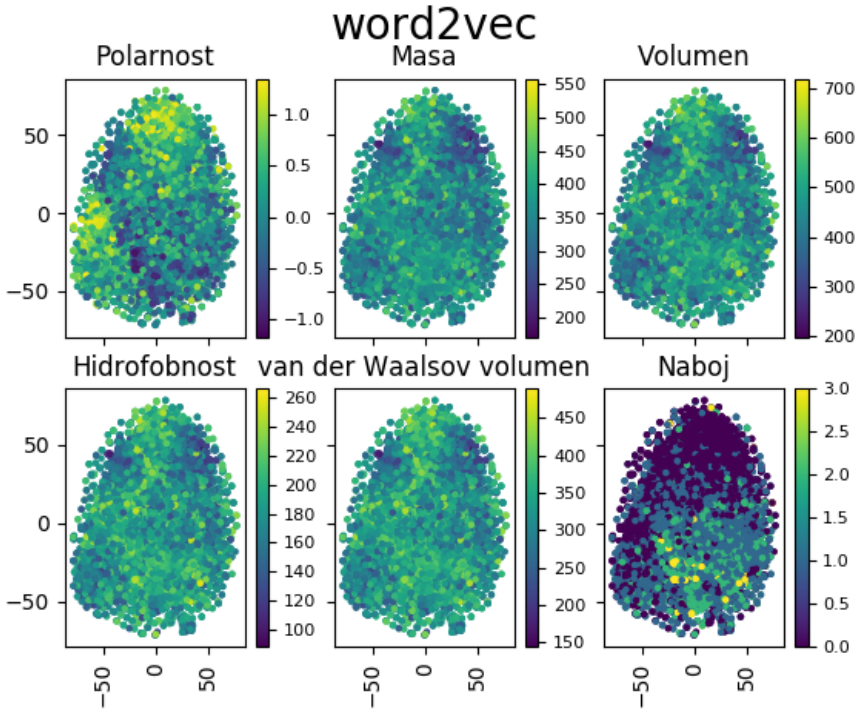
U ovom poglavlju bit će prikazani rezultati reprezentacija trigrama i treniranja modela iz koraka opisanih u prošlom poglavlju (vidi 3). Rezultate reprezentacija trigrama ćemo prikazati grafički, dok ćemo za rezultate rekurentne mreže koristiti mjeru točnosti na skupu za treniranje i validaciju.

4.1 Projekcija reprezentacija trigrama

U prvom koraku modeliranja reprezentacija proteina (vidi potpoglavlje 3.2) opisali smo dva pristupa kojima smo dobili različite reprezentacije trigrama, *word2vec* i *Glove*. Za dani skup trigrama tako dobivamo dva skupa, skup *word2vec* reprezentacija trigrama i skup *Glove* reprezentacija trigrama.

Kao i u slučaju obrade prirodnog jezika (vidi potpoglavlje 2.2) očekujemo da dobivene vektorske reprezentacije imaju smisla u stodimenzionalnom prostoru. U slučaju jezika to je značilo da su slične riječi međusobno blizu te da je razlika vektora sinonima konstantna (vidi primjer 2.2.1). U slučaju trigrama to znači da su trigrami sa sličnim svojstvima međusobno blizu te da se uočava smislena konstantnost u prostoru s obzirom na svojstva. Kako bismo to potvrdili, kao u radu [2], dobivene stodimenzionalne reprezentacije odlučili smo projicirati u dvodimenzionalnom prostoru koristeći metodu *Stohastic Neighbor Embedding* (*t-SNE*) [18] tako da dobivene projekcije obojimo s obzirom na vrijednost šest različitih svojstava trigrama: polarnost, hidrofobnost, naboj, masu, volumen i van der Waalsov volumen. Polarnost i hidrofobnost dobili smo kao aritmetičku sredinu vrijednosti svake od tri aminokiseline, dok smo preostala svojstva dobili kao sumu vrijednosti. Vizualizacija reprezentacija *word2vec* trigrama prikazana je na slici 4.1, a *Glove* trigrama na slici 4.2.

Već se na prvi pogled na obje reprezentacije mogu uočiti pravilnosti. Svojstva mase, oba volumena i hidrofobnosti grupiraju projekcije u gotovo identične grupe, dok se sličnost

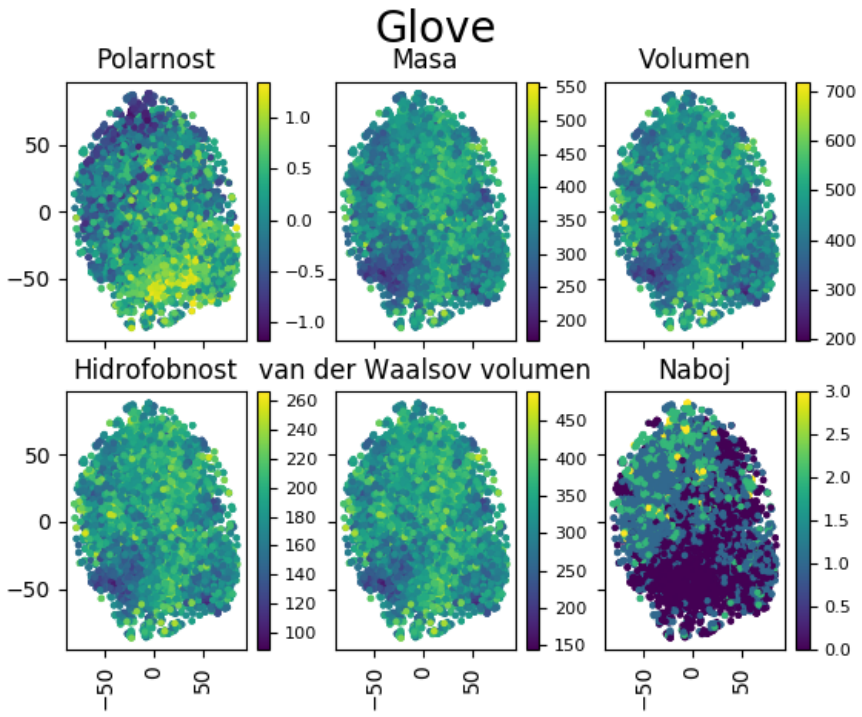


Slika 4.1: Projekcije stodimenzionalnih *word2vec* vektorskih reprezentacija trigrama obojanih po vrijednostima šest različitih svojstava

grupiranja uočava i kod svojstava polarnosti i naboja. Iako takva vizualiza ne dokazuje da su dobivene reprezentacije smislene ili povezane, ipak smatramo da rezultati vizualizacije nisu slučajni te da su dobar pokazatelj kvalitete reprezentacije u smislu sačuvane informacije.

4.2 Rezultati treniranja rekurentne mreže

U ovom potpoglavlju ćemo prikazati i objasniti rezultate istreniranih šest modela iz trećeg koraka modeliranja reprezentacija kroz 10, 20, 30, 40 i 50 epoha (vidi potpoglavlje 3.4). Ovisno o korištenom skupu za treniranje i modele ćemo nazivati model *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *Glove perm1*, *Glove perm2*, *Glove perm3*. U tablici 4.1 prikazani su rezultati treniranja i validacije zajedno za svaku epohu. Već na prvi pogled uočavamo kako modeli trenirani sa skupom *perm1* (u tablici označena s *perm1*) daju najbolje rezultate u slučaju obje *word2vec* i *Glove* reprezentacije trigrama. Takvi rezultati nas ne iznenađuju jer se tom permutacijom dobivena permutirana sekvenca najviše



Slika 4.2: Projekcije stodimenzionalnih *Glove* vektorskih reprezentacija trigrama obojanih po vrijednostima šest različitih svojstava

razlikuje od originalne sekvence. Tako mreži nije teško naučiti raspoznati pravu od permutirane sekvence čak već nakon 10 epoha. U slučaju druge permutacije (u tablici označeni s *perm2*), *Glove* modeli također postižu visoke rezultate, s nešto više potrebnih epoha u odnosu na prvu permutaciju. U slučaju *word2vec* reprezentacija trigrama postižu se lošiji rezultati koji dosta napreduju kroz veći broj epoha. Konačno, modeli trenirani sa skupovima *perm3*, (u tablici označeni s *perm3*) postižu najlošije rezultate, s time da modeli s *Glove* reprezentacijama trigrama i dalje postižu dosta bolje rezultate. Rezultate u slučaju druge i treće permutacije također smatramo očekivanima jer se permutirana sekvenca u oba slučaja vrlo malo razlikuje od originalne čime se problem otežava.

Možemo zaključiti da modeli trenirani s *Glove* reprezentacijama daju puno bolje rezultate u odnosu na *word2vec* reprezentacije za iste permutacije. Nadalje, uočava se korelacija između očekivane težine problema i uspješnosti modela. Što je problem teži, u smislu koliko se permutirana sekvenca razlikuje od originalne sekvence, to su rezultati modela lošiji. Konačno, rezultati validacijskog skupa su u svim slučajevima vrlo slični rezultatima skupa za treniranje. Iz toga zaključujemo da ni jedan model nije pretreniran te da se mogu koristiti

EPOHA	10		20		30		40		50	
PERM	TRAIN	VALID	TRAIN	VALID	TRAIN	VALID	TRAIN	VALID	TRAIN	VALID
word2vec reprezentacije trigrama										
perm1	0.6580	0.6888	0.9002	0.8796	0.9548	0.9510	0.9731	0.9802	0.9836	0.9871
perm2	0.5093	0.4717	0.5192	0.4600	0.5848	0.5449	0.7389	0.7760	0.8547	0.8848
perm3	0.5110	0.4858	0.5603	0.5550	0.6760	0.6710	0.7170	0.7051	0.7593	0.7243
Glove reprezentacije trigrama										
perm1	0.9783	0.9270	0.9957	0.9955	0.9984	0.9973	0.9986	0.9976	0.9993	0.9973
perm2	0.5221	0.4629	0.8728	0.8971	0.9640	0.9549	0.9809	0.9795	0.9881	0.9910
perm3	0.6383	0.6373	0.8504	0.8260	0.9090	0.9290	0.9381	0.9524	0.9567	0.9650

Tablica 4.1: Rezultati treniranja rekurentne mreže po epohama za šest različitih skupova: *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *Glove perm1*, *Glove perm2*, *Glove perm3*. Stupci TRAIN označavaju rezultate na skupu za treniranje, dok stupci VALID označavaju rezultate na skupu za validaciju

za generalizaciju i na neviđenim primjerima.

Zbog postizanja iznimno visokih rezultata točnosti u zadnjih 40 i 50 epoha odlučili smo kao finalne modele promatrati modele nakon 40 epoha u slučaju *word2vec* te nakon 20 u slučaju *Glove* reprezentacija. Rezultate treniranja u tim epohama smatramo dovoljno uspješnima, a možemo biti sigurniji da modeli nisu prenaučeni, odnosno da smo izbjegli problem pretreniranja.

Poglavlje 5

Testiranje reprezentacija

U ovom poglavlju opisat ćemo način na koji smo mjerili uspješnost dobivenih reprezentacija. U poglavlju 3 opisali smo kako smo dobili šest različitih modela rekurentne mreže. Svaki od dobivenih modela nam daje jednu reprezentaciju zbog čega, konačno dobivamo šest različitih reprezentacija iste proteinske sekvence (vidi sliku 3.6). Zbog usporedbe s radom [2] (vidi potpogavlje 3.1) promatrat ćemo još dvije dodatne reprezentacije, reprezentaciju proteinske sekvence dobivenu sumiranjem *word2vec* reprezentacija trigrama te reprezentaciju dobivenu sumiranjem *Glove* reprezentacija trigrama. Te reprezentacije ćemo tako nazivati reprezentacija *suma word2vec* te *suma Glove*. Preostalih šest reprezentacija ćemo, s obzirom na korišteni model za dobivanje reprezentacija, nazivati reprezentacija *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *Glove perm1*, *Glove perm2* ili *Glove perm3*. Dakle, promatramo i uspoređujemo osam različitih realnih vektorskih stodimenzionalnih reprezentacija.

5.1 Mjera uspješnosti reprezentacije

Do sada smo samo opisali način dobivanja osam različitih reprezentacija. Međutim, s tako dobivenim reprezentacijama ne možemo zaključiti koja od njih je najbolja i koju bismo trebali koristiti u primjenama. U radovima koje smo koristili za motivaciju (vidi potpoglavlje 3.1) reprezentacije su mjerili odabirom problema predviđanja familije proteina (vidi potpoglavlje 5.3) na kojem su postigli iznimno dobre rezultate. Mjerenje uspješnosti na samo jednom tipu problema ne pokazuje uspješnost reprezentacije u generalnom smislu. Određena reprezentacija može davati iznimno dobre rezultate na jednom problemu, a opet, dosta lošije rezultate na drugom problemu. Zbog toga bi u svrhu mjerenja uspješnosti reprezentacija bilo potrebno najprije odrediti prostor problema u kojima se koriste proteinske sekvence. Nakon toga, trebalo bi odrediti mjeru ekvivalentnosti problema te skup problema podijeliti u klase ekvivalentnih problema. Konačno, iz svake klase bismo odabrali

probleme predstavnike klasa. Tada bismo dobivene reprezentacije trebali testirati na svim takvim odabranim problemima. Time bismo imali precizno definiranu mjeru uspješnosti s kojom bismo mogli tvrditi koja od reprezentacija je najbolja. Ipak, takav pristup bio bi prezahtjevan za ovaj rad. Umjesto toga, odlučili smo se odabrati tri *različita* problema bitna u praksi. Različitost problema odredili smo na temelju broja klasa koje se predviđaju te na temelju informacije koja se koristi za rješavanje problema. Tri odabrana problema su problem predviđanja familije proteina, problem određivanja termofilnosti proteinske sekvence te problem određivanja klase ribosoma te će biti detaljno opisani u nastavku. Kako je već spomenuto u potpoglavlju 1.4, zbog količine dostupnih informacija o proteinima reprezentiranih u obliku primarne strukture (vidi potpoglavlje 1.3), sva tri problema modeliraju se tako da je ulaz u model informacija o primarnoj strukturi proteinske sekvence. Svaki skup proteinskih sekvenci će biti preslikan u skup realnih stodimenzionalnih vektorskih reprezentacija proteinskih sekvenci, za odabranu reprezentaciju. Budući da su sve reprezentacije fiksne dimenzije 100, preslikani skup smo modelirali realnom matricom dimenzije $N \times 100$, gdje je N broj proteinskih sekvenci u početnom skupu. Te skupove ćemo također, ovisno o korištenoj reprezentaciji sekvence, nazivati skup *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2* ili *Glove perm3*.

5.2 Model za treniranje

Cilj ovog poglavlja je usporediti rezultate problema na temelju različitih reprezentacija. Zbog toga se za rješavanje odabranih triju problema nismo usmjerili na optimiziranje modela u smislu dobivanja što boljih rezultata, već nam je ideja bila koristiti model koji je dovoljno općenit te za koji nije potrebna optimizacija parametara. Osim toga, sve reprezentacije su fiksne dimenzije pa možemo koristiti bilo koji standardni model strojnog učenja. Odlučili smo se za *Random Forest klasifikator* zbog brzog treniranja, lake paralelizacije, malog broja parametara i točnosti na mnogim problemima strojnog učenja. *Random Forest klasifikator* sastoji se od odabranog broja stabala odlučivanja koja uče na podskupovima skupa za treniranje kako bismo poboljšali prediktivnu točnost i izbjegli problem pretreniranja. U sva tri problema koristili smo *Random Forest klasifikator* s 200 stabala i maksimalne dubine 15 te smo točnost određivali kao aritmetičku sredinu točnosti kroz deset foldova *unakrsne validacije* (engl. *cross-validation*). Unakrsna validacija radi tako da dijeli skup za treniranje na deset podskupova te kroz deset različitih koraka uzima točno jedan od njih za validaciju, a ostale za treniranje.

U slučaju multiklasnog problema koristili smo metodu *jedan protiv svih* (engl. *One-vs.-rest*) koja koristi onoliko klasifikatora koliko ima klasa. Svaki klasifikator rješava binarni problem, u kojemu je promatrana klasa označena pozitivnom labelom, a preostale negativ-

nom. Za konačnu odluku uzima odluku onoga klasifikatora koji daje najveću vjerojatnost pripadanja toj klasi.

Uočimo, za svaki problem ćemo, ovisno o korištenom skupu za treniranje, konačno dobiti osam različitih modela. Tako ćemo ih opet nazivati model *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2* ili *Glove perm3*.

5.3 Predviđanje familije proteina

Problem predviđanja familije proteina je problem obrađen u radovima [2] i [16]. Familija proteina je klasa proteina povezanih na temelju evolucije te proteini unutar iste familije uglavnom imaju vrlo sličnu primarnu i tercijarnu strukturu (vidi potpoglavlje 1.3). Zbog te činjenice, smatramo da za ovaj problem veliku važnost ima upravo redoslijed aminokiselina u primarnoj strukturi. Trenutno je identificirano više od 60 000 proteinskih familija što je jedan od glavnih razloga za implementaciju automatskog raspoređivanja proteinskih sekvenci u familije (za detalje vidi [15]). Klasifikacija proteina na temelju primarne strukture, dakle sekvence aminokiselina, uglavnom se modelirala iz svojstava proteinske sekvence kao što su masa, volumen i naboj [2]. Budući da smatramo da su u našim reprezentacijama takve informacije već zakodirane, mi ćemo za klasifikaciju koristiti isključivo reprezentacije u obliku stodimenzionalnih realnih vektora.

Korištena baza proteinskih sekvenci poznata je pod nazivom *ProtVec* [1] koji se sastoji od 324 018 proteinskih sekvenci i 7021 klase familija. Iako su u radu [2] koristili sve sekvence i zabilježili rezultat uspješnosti za svaku od familija, u ovom radu smo odlučili gledati samo odabrani podskup tih familija. Naime, kako je već napomenuto, nije nam cilj bio usmjeriti se na točno određeni problem, već pokazati mogućnost generalizacije reprezentacija. Osim toga, budući da taj problem rješavamo metodom *jedan protiv svih* te unakrsnom validacijom s deset foldova, to bi značilo da bismo trebali trenirati više od $8 \cdot 70\,000$ modela (za svaku reprezentaciju, za svaku familiju je potrebno deset modela za svaki fold), što bi zahtijevalo veće hardverske resurse od onih koji su nam bili dostupni.

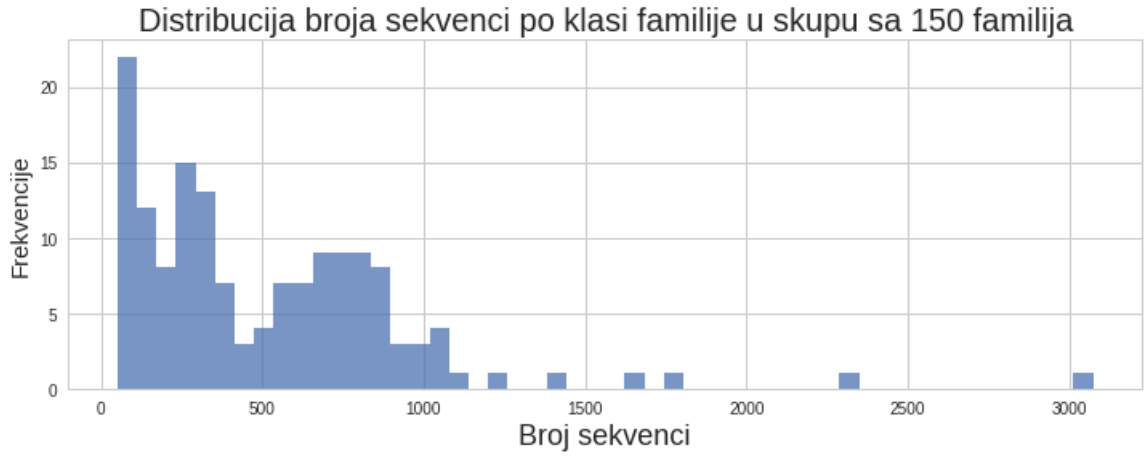
Prvi korak selekcije skupa podataka bio je promatrati samo one familije s minimalno 50 sekvenci po familiji čime smo dobili podskup od 260 856 proteinskih sekvenci i 1061 familiju. Distribucija broja sekvenci po klasi familije prikazana je histogramom na slici 5.1.

Možemo uočiti kako većina familija sadrži oko 100 proteina, dok najveći broj proteinskih sekvenci po familiji iznosi 3084. Budući da nam je 1061 familija i dalje predstavljala računalno prezahtjevan problem, odlučili smo uzeti reprezentativan uzorak tog podskupa

[1] Preuzeto s https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/JMFHTN, 04/2017

Slika 5.1: Histogram distribucije broja sekvenci za svaku familiju u filtriranom skupu podataka s minimalno 50 sekvenci u svakoj familiji

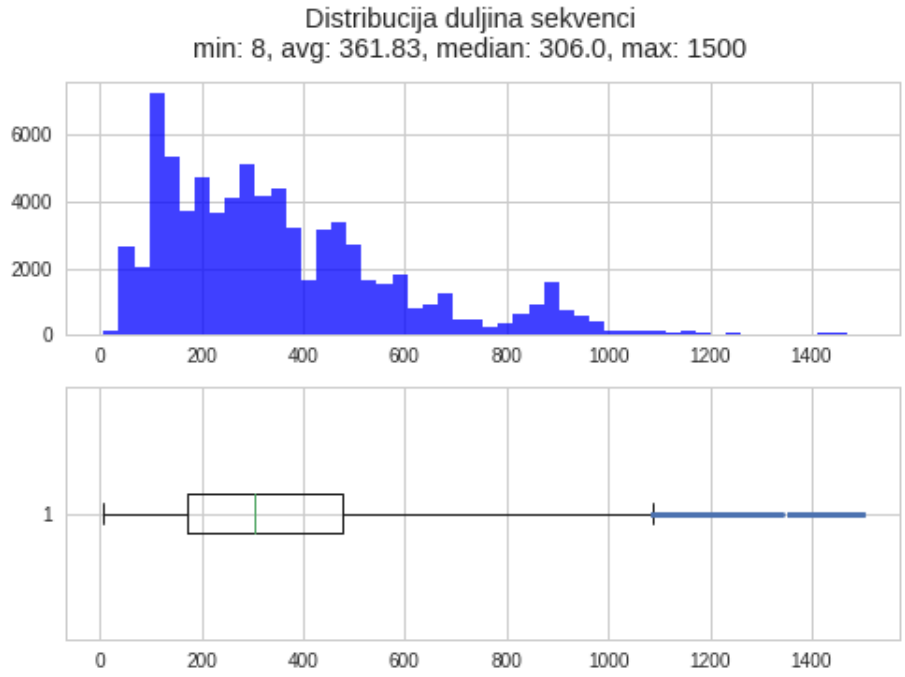
od 150 familija. Pod reprezentativnim uzorkom mislimo na uzorak koji bi sačuvao distribuciju broja sekvenci po familijama kako bismo bili sigurni da smo u rješavanje problema uključili klase svih veličina. Tako smo dobili konačan podskup od 77 160 proteinskih sekvenci i 150 familija, a distribucija broja sekvenci po familiji je prikazana histogramom na slici 5.2.



Slika 5.2: Histogram distribucije broja sekvenci za svaku familiju u filtriranom skupu podataka sa 150 familija i minimalno 50 sekvenci u svakoj familiji

Distribucija duljina sekvenci promatranoga filtriranog skupa prikazana je na slici 5.3 iz

kojeg možemo uočiti nešto drukčiju distribuciju u odnosu na skup koji se koristio za treniranje rekurentnih mreža (vidi sliku 3.4). Već samo na temelju duljina sekvenci možemo zaključiti da taj skup sadrži potpuno nove primjere čime smo sigurni da tim problemom provjeravamo mogućnost generalizacije reprezentacija dobivenih iz rekurentnih modela.



Slika 5.3: Distribucija duljina sekvenci u filtriranom skupu za učenje problema predviđanja familija

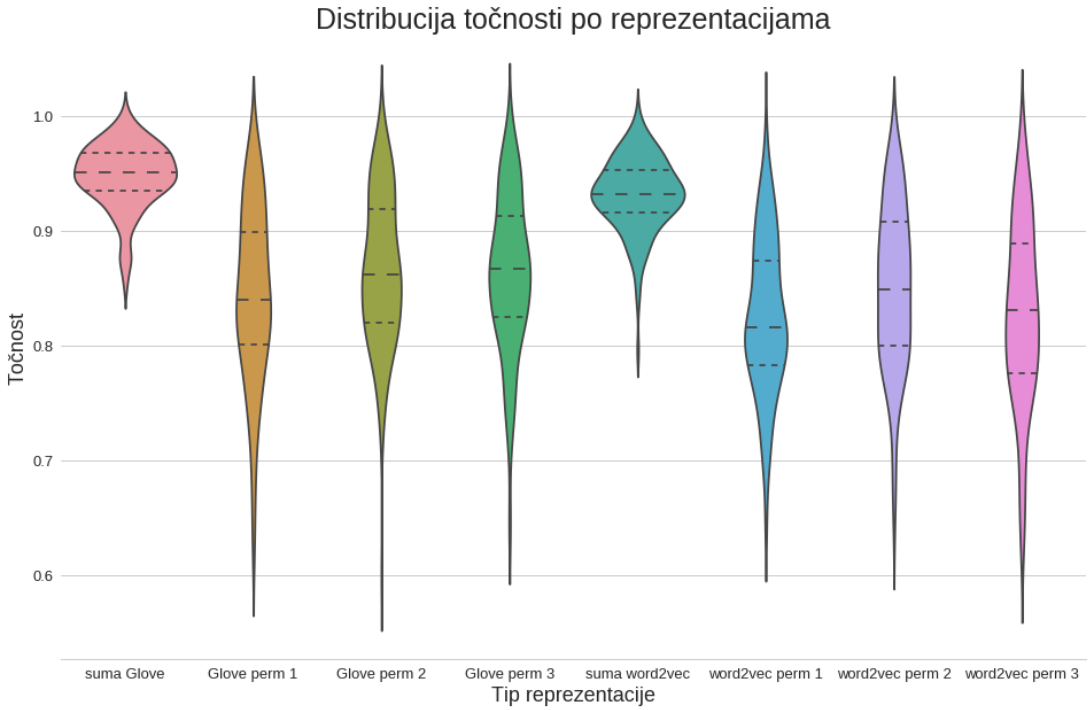
Rezultate točnosti modela svih osam reprezentacija prikazali smo grafom *violin plot* za svaku reprezentaciju. Svaki graf prikazuje distribuciju točnosti skupa od 1061 proteinske familije.

Iako obje reprezentacije *suma word2vec* i *suma Glove* daju bolju točnost na većem broju familija, postoje individualni primjeri familija na kojima reprezentacije dobivene dubokim neuronskim mrežama daju bolje rezultate, a prikazani su u tablici 5.1.

Nadalje, dobiveni rezultati s reprezentacijama *suma word2vec* nad filtriranim familijama odgovaraju rezultatima rada [2]. Time smo potvrdili korektnost naših metoda u dobivanju reprezentacija trigrama pomoću *word2vec*. Konačno, možemo uočiti kako *Glove* reprezentacije za sva četiri modela daju bolje rezultate od *word2vec* reprezentacija čime smo također potvrdili kako metoda *Glove* u praksi daje bolje rezultate od *word2vec* (vidi

OZNAKA FAMILIJE	BROJ SEKVENCI	REPREZENTACIJA	*suma*	*perm1*	*perm2*	*perm3*
Ribosom S12 S23	1016	*word2vec*	0.924	0.928	**0.947**	0.942
		Glove	0.952	0.946	**0.954**	0.949
Ribosomal S11	980	*word2vec*	0.937	0.949	**0.968**	0.962
		Glove	0.957	0.972	**0.981**	0.976
Ribosomal L32p	825	*word2vec*	0.938	0.933	**0.952**	0.938
		Glove	0.957	0.941	**0.965**	0.956
Ribosomal L35p	769	*word2vec*	0.943	0.931	**0.950**	0.929
		Glove	0.961	0.924	**0.965**	0.941
Ribosomal L29	757	*word2vec*	0.943	0.922	0.906	0.888
		Glove	0.952	0.929	**0.954**	0.948
SPOUT MTase	614	*word2vec*	0.945	0.927	**0.961**	0.942
		Glove	0.958	**0.967**	0.962	0.964
AICARFT IMPCHas	468	*word2vec*	0.941	0.951	**0.979**	0.966
		Glove	0.966	**0.974**	0.956	**0.974**
PsbM	137	*word2vec*	0.985	**0.989**	0.982	0.978
		Glove	0.985	0.978	**0.996**	**0.996**
RbcS	85	*word2vec*	0.958	0.894	0.910	0.922
		Glove	0.963	0.904	**0.976**	0.928
RtcB	54	*word2vec*	0.878	0.898	**0.893**	0.922
		Glove	0.860	0.925	0.928	**0.973**

Tablica 5.1: Rezultati točnosti osam različitih modela: *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2*, *Glove perm3* za koje je rezultat modela s reprezentacijama rekurentnih mreža postigao bolje rezultate od reprezentacije sumom. Podebljano su označeni rezultati permutacija posebno za *word2vec* i *Glove* koji su bolji od rezultata reprezentacija sumom



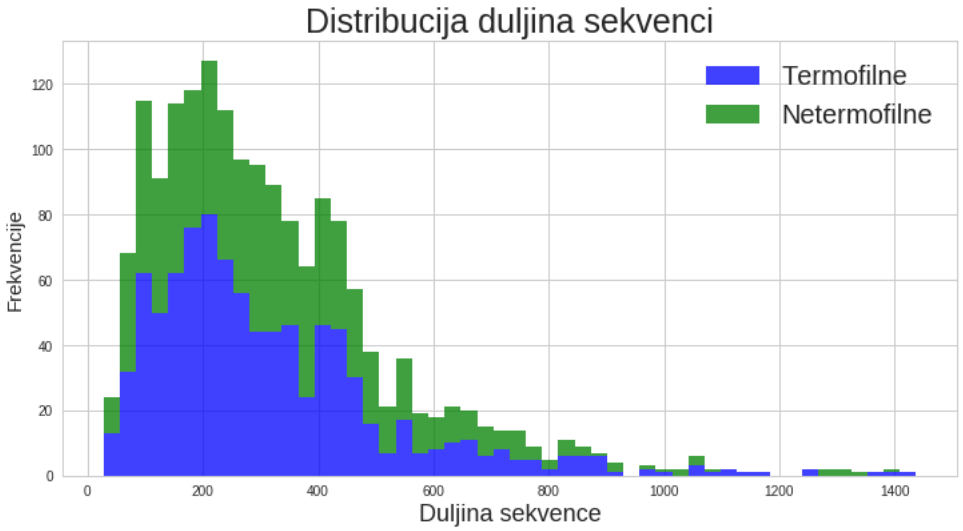
Slika 5.4: Graf *violin plot* koji prikazuje distribuciju točnosti problema predviđanja familija osam različitih modela *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2*

potpoglavlje 2.2). Iz grafa 5.4 ne možemo zaključiti koja od reprezentacija dobivenih iz rekurentnih modela daje najbolje rezultate za taj problem te postoji li, i u ovom slučaju, korelacija između težine problema rekurentne mreže i uspješnosti reprezentacije. Da bismo to odredili, koristili smo *Welchov t-test*, modifikaciju Studentovog t-testa, koji se koristi za testiranje hipoteze jednakosti aritmetičkih sredina bez pretpostavke o jednakosti varijanci. U svim testovima koristili smo varijantu jednostranog testa te na razini značajnosti od 0.05 određivali možemo li tvrditi je li određena reprezentacija bolja u smislu točnosti modela po familijama. Posebno smo uspoređivali rezultate za modele *word2vec* (*word2vec perm1*, *word2vec perm2* te *word2vec perm3*) i posebno za *Glove* (*Glove perm1*, *Glove perm2* te *Glove perm3*) te u oba slučaja možemo tvrditi da model *perm2* daje signifikantno bolje rezultate od *perm1* i od *perm3*. Konačno, kako bismo usporedili koja od reprezentacija *word2vec* i *Glove* je bolja, usporedili smo rezultate parova *word2vec perm2* i *Glove perm2*, kao primjer najboljih reprezentacija rekurentnih mreža za *word2vec* odnosno *Glove*, te *suma word2vec* i *suma Glove*. U oba slučaja, na razini značajnosti 0.05 možemo tvrditi da su rezultati *Glove* reprezentacija bolji.

5.4 Određivanje termofilnosti

Termofilnost proteina označava otpornost proteina na visoke temperature te su takvi proteini od posebnog interesa u biološkim istraživanjima. Posebno je zanimljivo pitanje kako takvi proteini izdržavaju visoke temperature te koji oblik strukture poprimaju kako bi to postigli [9]. Za razliku od problema predviđanja familija (vidi potpoglavlje 5.3), u kojemu važnu ulogu ima redoslijed aminokiselina u primarnoj strukturi proteinske sekvence, za taj problem veću ulogu imaju svojstva proteina. Osim toga, problem razlikovanja termofilnog od netermofilnog proteina je binarni problem zbog čega možemo reći da je ovaj problem dovoljno različit kako bismo testirali uspješnost reprezentacija.

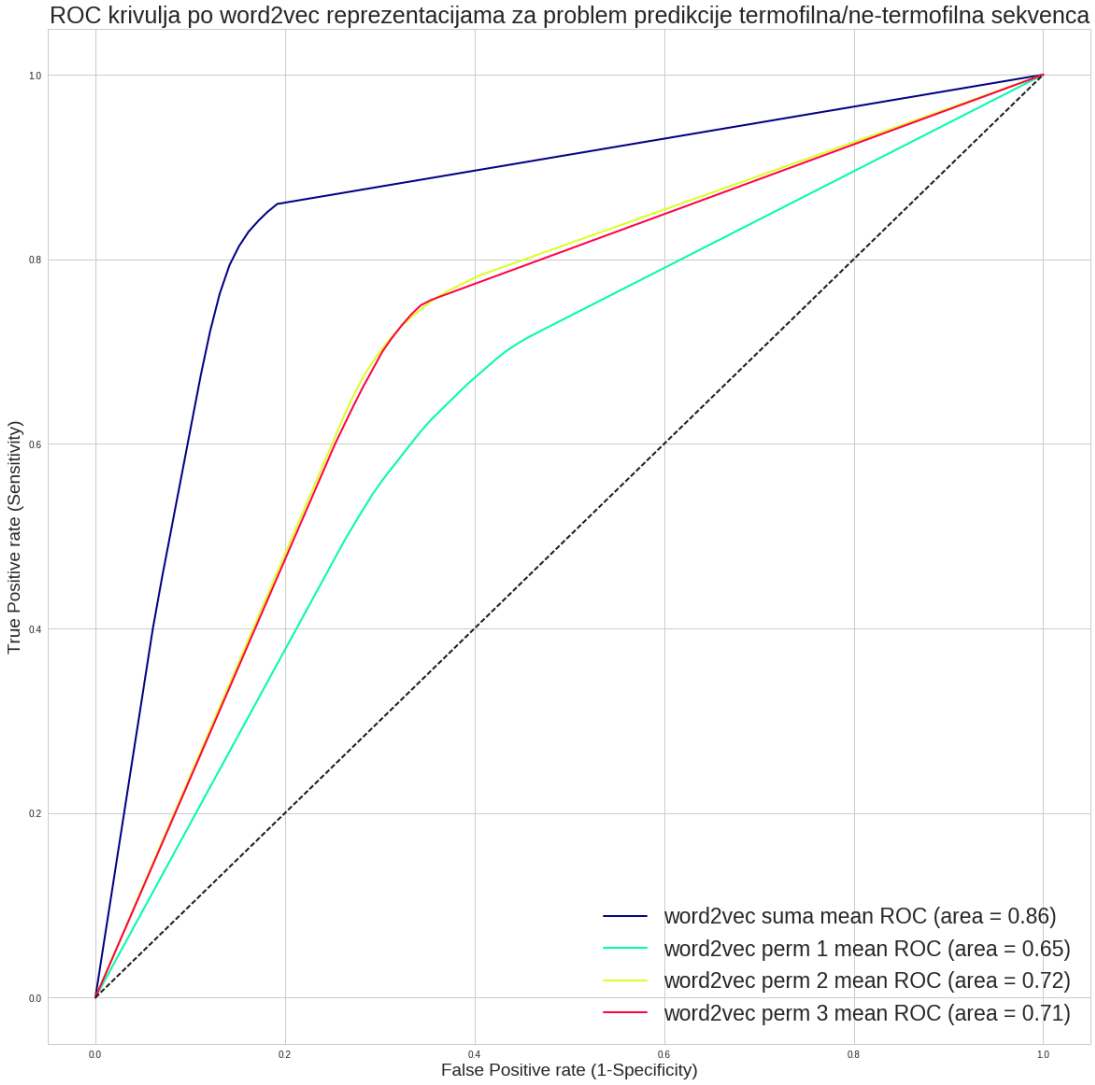
Skup podataka podskup je baze poznate pod nazivom *ProFET* [2] te se sastoji od 1703 proteinske sekvence od kojih je 913 termofilnih, a 790 netermofilnih. Distribucija duljina sekvenci prikazana je histogramom na slici 5.5.



Slika 5.5: Konkatenirani histogram distribucija duljina sekvenci u skupu za učenje problema predviđanja familija obojanih po klasi termofilnosti

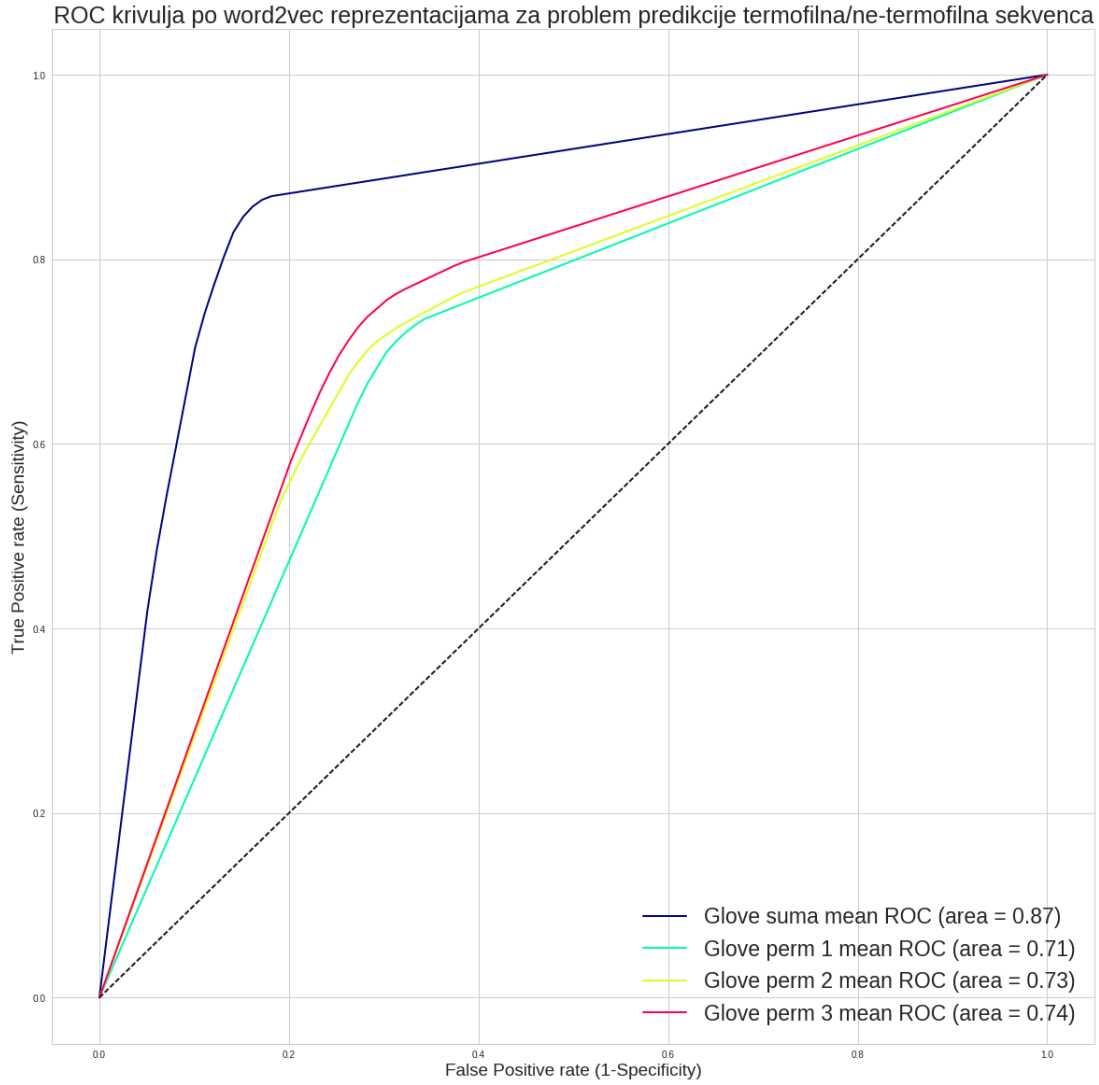
Za taj problem rezultate smo prikazali mjerama *točnosti*, *preciznosti*, *odziva* i *F1*. Točnost mjeri postotak točno klasificiranih primjera, preciznost mjeri postotak točno klasificiranih pozitivnih primjera u odnosu na sve pozitivne primjere, dok odziv (engl. *Recall, Sensitivity*) mjeri postotak točno klasificiranih pozitivnih primjera u odnosu na sve pozitivno klasificirane primjere. F1 mjera predstavlja harmonijsku sredinu preciznosti i odziva. Rezultate smo grafički prikazali *ROC* (engl. *Receiver Operating Characteristics*) krivu-

[2]Preuzeto s https://github.com/ddofer/ProFET/tree/master/fastas/Benchmarks/Thermophiles, 05/2017



Slika 5.6: *ROC* krivulje s *AUC* (*area*) mjerama koje prikazuju rezultate četiri modela: *suma word2vec*, *word2vec perm1*, *word2vec perm2* i *word2vec perm3*. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

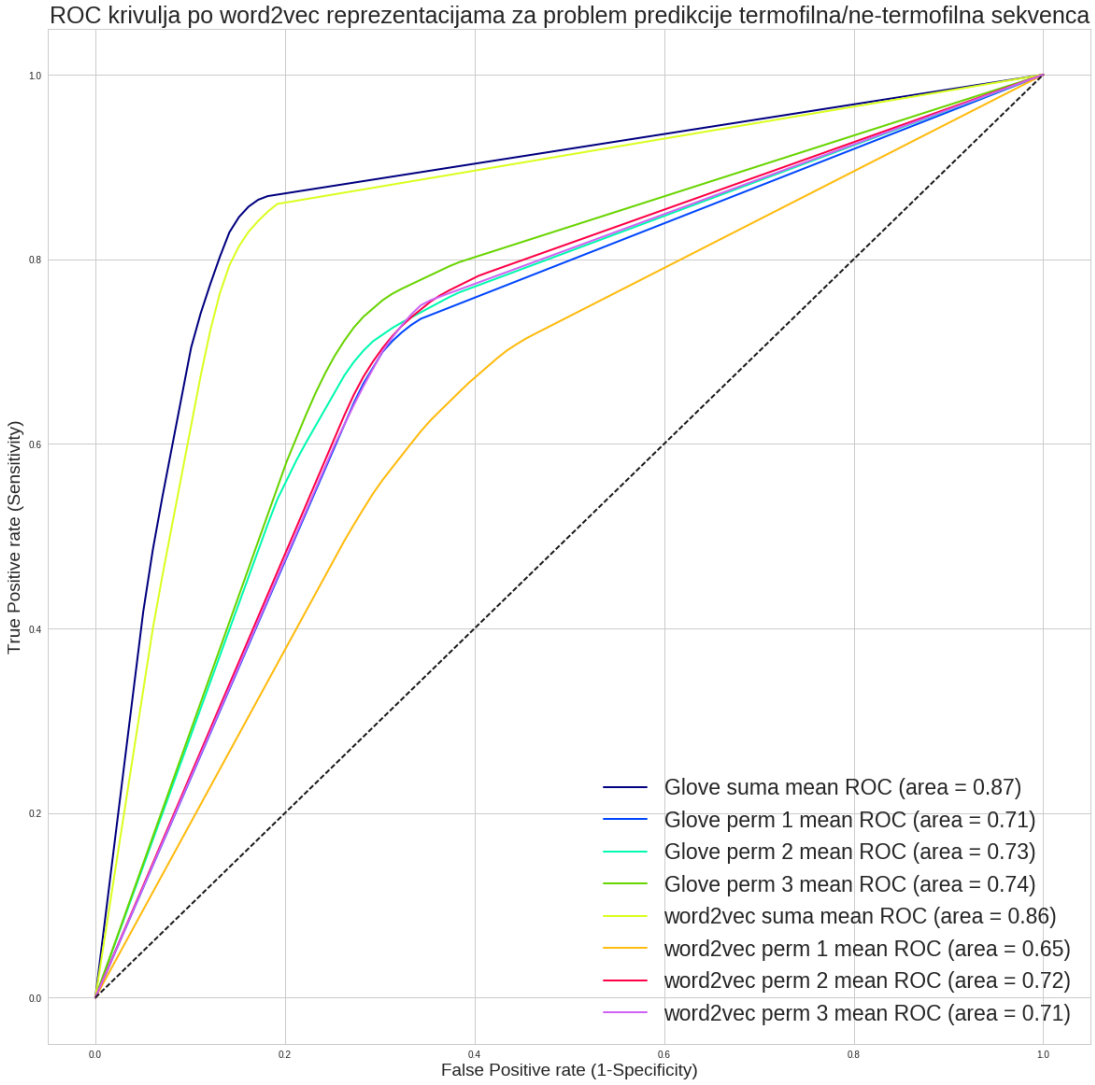
ljama koje pokazuju odziv (engl. *True Positive Rate, TPR*) kao funkciju *fall-outa* (engl. *False Positive Rate, FPR*) zajedno s mjerom površine ispod krivulje, *AUC* (engl. *Area Under Curve*). *FPR* odgovara mjeri 1 − *specifičnost*, gdje je specifičnost (engl. *Specificity*) omjer točno klasificiranih negativnih primjera u odnosu na sve negativne primjere. Što je model bolji, to se krivulja koja prikazuje taj model nalazi na većoj udaljenosti od krivulje



Slika 5.7: *ROC* krivulje s *AUC* (*area*) mjerama koje prikazuju rezultate četiri modela: *suma Glove*, *Glove perm1*, *Glove perm2* i *Glove perm3*. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

slučajnog modela te ima vrijednost *AUC* što bliže jedan.

Tablica 5.2 prikazuje rezultate mjera točnosti, preciznosti, odziva i F1 za svih osam različitih modela. Ponovno možemo uočiti kako oba modela *suma word2vec* i *suma Glove* daju bolje rezultate od modela s reprezentacijama dobivenih rekurentnim mrežama. Također, rezultati *Glove* modela daju bolje rezultate od *word2vec* modela dok se, u slučaju repre-



Slika 5.8: *ROC* krivulje s *AUC* (*area*) mjerama za rezultate svih osam modela: *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*,*suma Glove*, *Glove perm1*, *Glove perm2* i *Glove perm3*. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

zentacija rekurentnih mreža, uočava korelacija između uspješnosti i težine problema. Permutacije *perm2* i *perm3* u oba slučaja *word2vec* i *Glove* daju bolje rezultate od *perm1*. Isti zaključci mogu se potvrditi i iz grafičkih rezultata. Na slici 5.6 prikazane su *ROC* krivulje koje predstavljaju rezultate *word2vec* modela, na slici 5.7 su prikazani rezultati za *Glove* modele, dok su na slici 5.8 prikazani ukupni rezultati svih osam modela zajedno.

REPREZENTACIJA	točnost	F1	odziv	preciznost
suma word2vec	**0.855**	**0.863**	**0.851**	**0.876**
word2vec perm1	0.656	0.676	0.676	0.678
word2vec perm2	0.719	0.741	0.751	0.732
word2vec perm3	0.713	0.735	0.744	0.727
suma Glove	**0.868**	**0.875**	**0.859**	**0.893**
Glove perm1	0.709	0.727	0.723	0.731
Glove perm2	0.727	0.739	0.723	0.758
Glove perm3	0.747	0.764	0.763	0.764

Tablica 5.2: Rezultati točnosti osam različitih modela: *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2*, *Glove perm3* za problem određivanja termofilnosti proteinskih sekvenci. Podebljano su označena dva najbolja rezultata

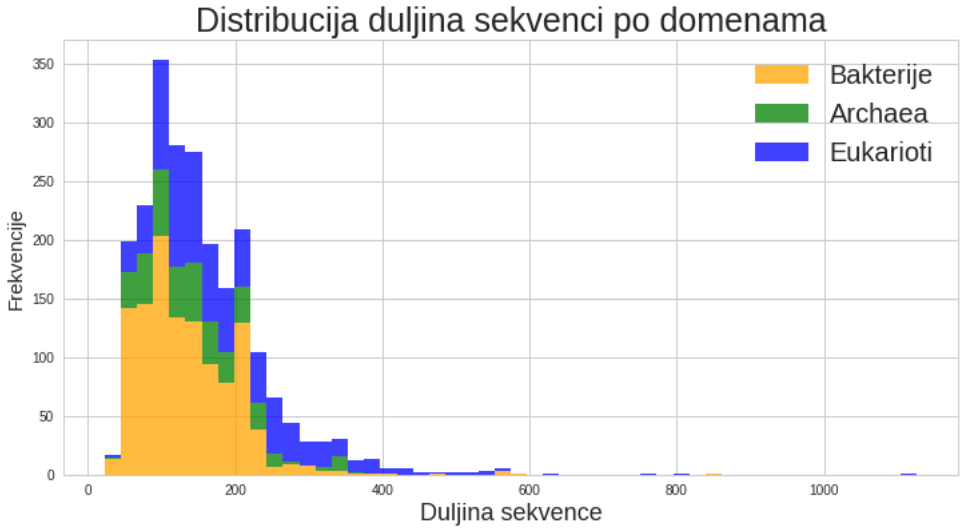
5.5 Određivanje klase ribosoma

Postoje tri glavne domene života, Bakterije, Arheje (*Archaea*) i Eukarioti, a glavni organel sve tri domene je ribosom. Ribosomi se sastoje od proteina i ribosomske *RNA*, a služe za prevođenje genetske upute (za detalje vidi [14]). Usporedba građe ribosoma iznimno je korisna u području biologije kako bi se odredila povezanost raznih organizama. Bilo bi korisno istražiti možemo li, samo na temelju informacije o proteinima u ribosomima, odrediti kojoj domeni života taj ribosom pripada. Ribosomi domena Bakterija i Archaea svrstani su u istu kategoriju prokariotskih ribosoma zbog čega se ribosomi dijele u samo dvije kategorije, prokariotske i eukariotske ribosome. Ipak, u svrhu tog problema, zanima nas možemo li dobiti finiju podjelu na sve tri domene života zbog čega ćemo govoriti o tri klase ribosoma, eukariote, archaea i bakterije. Određivanje tih klasa na temelju primarne strukture proteinske sekvence uzeli smo kao treći zadatak za mjerenje uspješnosti osam različitih reprezentacija. Taj problem, za razliku od prethodna dva problema, za odluku klase trebao bi koristiti i informaciju koja nije nužno vezana uz protein, a to je molekula ribosomske *RNA*. Rješavanjem ovog problema provjerit ćemo jesmo li uspjeli i tu informaciju implicitno pohraniti u našim reprezentacijama.

Skup podataka također je podskup baze *ProFET* [3] te se sastoji od 2286 proteinskih sekvenci od kojih je 1146 klase bakterija, a 760 eukariota te 380 archaea. Distribucija

[3]Preuzeto s https://github.com/ddofer/ProFET/tree/master/fastas/Ribosomes, 05/2017

duljina sekvenci prikazana je histogramom na slici 5.5.



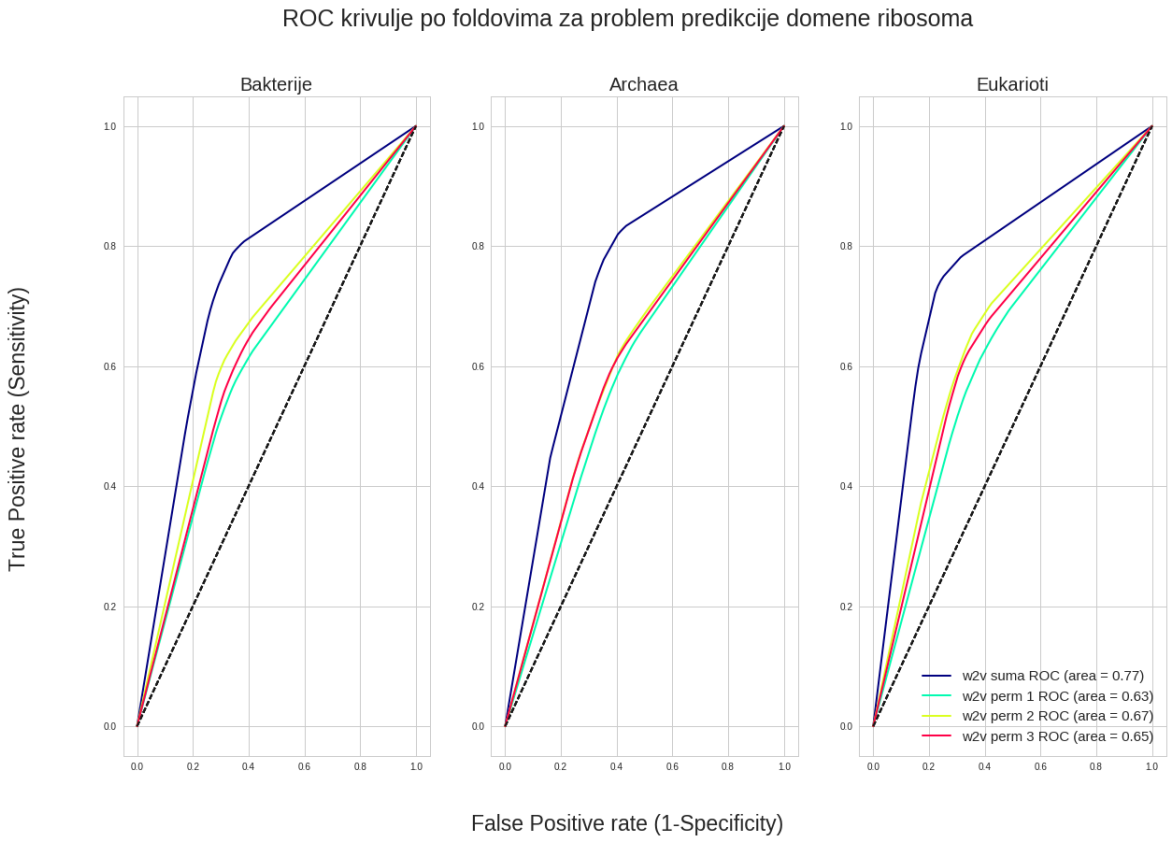
Slika 5.9: Konkatenirani histogram distribucija duljina sekvenci u skupu za učenje problema predviđanja klase ribosoma obojanih po klasi termofilnosti

Za taj problem rezultate smo prikazali analogno kao u prošlom problemu mjerama točnosti, preciznosti, odziva i F1 te grafički *ROC* krivuljama uz mjeru *AUC*. Sve rezultate smo prikazali zasebno za svaku klasu bakterija, archae i eukariote iz rezultata dobivenih metodom *jedan protiv svih* (vidi početak ovog poglavlja).

Tablica 5.3 prikazuje rezultate mjera točnosti, preciznosti, odziva i F1 za svih osam različitih modela i sve tri klase. Za razliku od prethodna dva problema, dobiveni rezultati su tek nešto bolji od slučajnog modela. U ovom slučaju čak niti reprezentacije dobivene sumom reprezentacija trigrama ne uspijevaju dati dobre rezultate. Možemo zaključiti da za klasifikaciju tog problema ipak nije dovoljna samo informacija o proteinskoj sekvenci, već bismo toj informaciji trebali dodati informaciju o ribosomskoj *RNA*. Ni reprezentacije dobivene rekurentnom mrežom nisu uspjele implicitno zakodirati tu informaciju. Ipak, i dalje uočavamo da reprezentacije dobivene sumom daju bolje rezultate od reprezentacija dobivenih rekurentnim mrežama. U slučaju *word2vec* i dalje se uočava pravilnost u kojoj reprezentacija *word2vec perm1* daje najlošije rezultate, dok u slučaju *Glove* to nije slučaj. Rezultati se bolje mogu usporediti na grafičkom prikazu pomoću *ROC* krivulja na slici 5.10 za *word2vec* reprezentacije, na 5.11 za *Glove* reprezentacije te na slici 5.12 za svih osam reprezentacija zajedno.

REPREZENTACIJA	KLASA	točnost	F1	odziv	preciznost
suma word2vec	bakterija	0.574	0.666	0.846	0.550
	archaea	0.619	0.688	0.838	0.584
	eukarioti	0.567	0.663	0.853	0.543
word2vec perm1	bakterija	0.571	0.609	0.666	0.563
	archaea	0.574	0.593	0.624	0.567
	eukarioti	0.525	0.607	0.734	0.518
word2vec perm2	bakterija	0.573	0.629	0.721	0.558
	archaea	0.584	0.615	0.665	0.575
	eukarioti	0.545	0.625	0.761	0.532
word2vec perm3	bakterija	0.577	0.626	0.703	0.564
	archaea	0.611	0.623	0.646	0.604
	eukarioti	0.555	0.614	0.711	0.541
suma Glove	bakterija	0.576	0.664	0.832	0.552
	archaea	0.624	0.683	0.805	0.593
	eukarioti	0.578	0.655	0.800	0.555
Glove perm1	bakterija	0.559	0.622	0.723	0.546
	archaea	0.569	0.611	0.678	0.556
	eukarioti	0.548	0.638	0.796	0.533
Glove perm2	bakterija	0.536	0.627	0.775	0.527
	archaea	0.578	0.686	0.603	0.575
	eukarioti	0.578	0.635	0.737	0.558
Glove perm3	bakterija	0.575	0.613	0.671	0.565
	archaea	0.588	0.600	0.622	0.583
	eukarioti	0.527	0.614	0.753	0.518

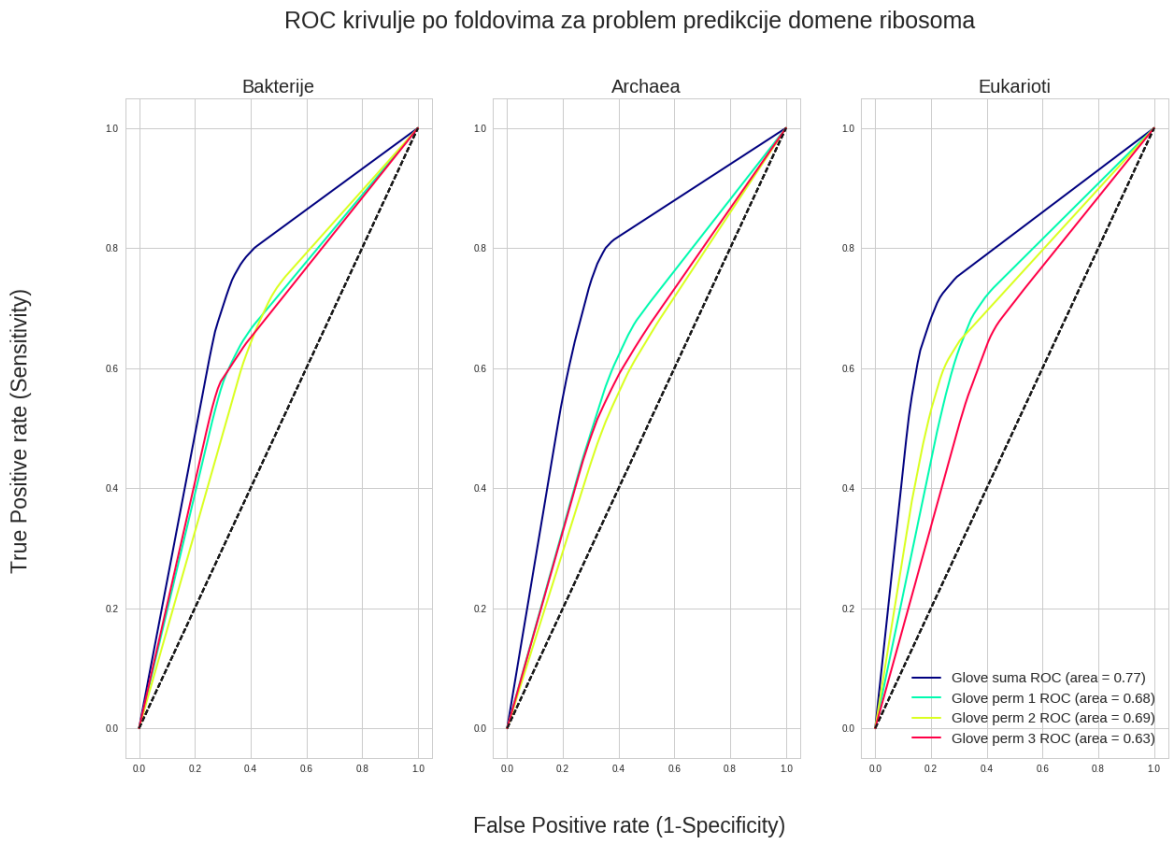
Tablica 5.3: Rezultati točnosti osam različitih modela: *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2*, *Glove perm3* za problem određivanja klase ribosoma proteinskih sekvenci. Rezultati su dobiveni zasebno po klasama metodom *jedan protiv svih*



Slika 5.10: *ROC* krivulje s *AUC* (*area*) mjerama za rezultate četiriju modela *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3* po klasama ribosoma bakterije, archaea i eukarioti. Za *word2vec* se koristi oznaka *w2w*. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

5.6 Analiza rezultata

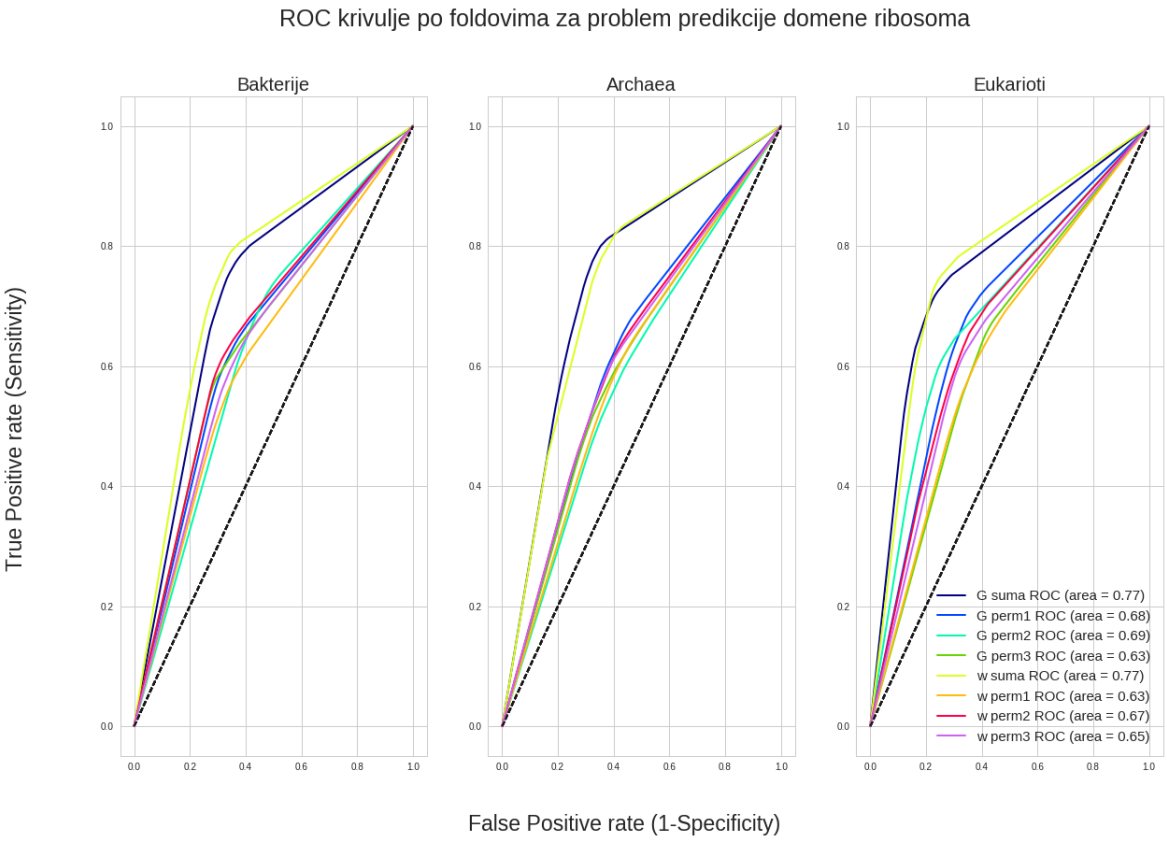
Cilj rada bio je modelirati različite reprezentacije realnim vektorima fiksnih dimenzija korištenjem dubokih neuronskih mreža. Dobivene reprezentacije bi trebale davati dobre rezultate na različitim klasifikacijskim problemima koji kao informaciju koriste primarnu strukturu proteinske sekvence. Uz pomoć rekurentnih mreža dobili smo šest različitih reprezentacija te smo, zbog usporedbe s radom [2], promatrali još dvije reprezentacije dobivene sumiranjem reprezentacija trigrama. Kako bismo mjerili uspješnost reprezentacija, rješavali smo tri različita problema, predviđanje klase familije, određivanje termofilnosti proteina te određivanje klase ribosoma. Rezultati za sva tri problema pokazuju gotovo iste zaključke. Reprezentacije dobivene sumom reprezentacija trigrama daju bolje rezultate



Slika 5.11: *ROC* krivulje s *AUC* (*area*) mjerama za rezultate četiri modela *suma Glove*, *Glove perm1*, *Glove perm2*, *Glove perm3* po klasama ribosoma bakterije, archaea i eukarioti. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

od reprezentacija dobivenih rekurentnim mrežama. Usporedbom reprezentacija dobivenih treniranjem rekurentne mreže različitim permutacijama, zaključujemo da reprezentacije dobivene prvom permutacijom pokazuju najlošije rezultate. Taj se rezultat podudara s uočenom korelacijom težine problema koju je rješavala rekurentna mreža i uspješnosti reprezentacije. U usporedbi istih parova s različitim reprezentacijama trigrama *word2vec* i *Glove* zaključujemo da *Glove* reprezentacije daju bolje rezultate što potvrđuje tvrdnju rada [23] da su *Glove* reprezentacije informativnije od *word2vec* reprezentacija.

Iako metodom rekurentnih mreža nismo uspjeli dobiti reprezentacije koje daju bolje rezultate od reprezentacija dobivenih sumiranjem trigrama, dobiveni rezultati pokazuju dobar smjer budućeg istraživanja. Naime, skup podataka s kojim smo trenirali rekurentne mreže je skup koji se sastojao od samo 20-ak tisuća labeliranih sekvenci. Zbog kompleksnosti ulaznog skupa i količine informacije koju želimo pohraniti u reprezentacijama,



Slika 5.12: *ROC* krivulje s *AUC* (*area*) mjerama za rezultate svih osam modela *suma word2vec*, *word2vec perm1*, *word2vec perm2*, *word2vec perm3*, *suma Glove*, *Glove perm1*, *Glove perm2*, *Glove perm3* po klasama ribosoma bakterije, archaea i eukarioti. Za *word2vec* se koristi oznaka *w* dok se za *Glove* koristi oznaka *G*. X-os prikazuje vrijednost *FPR*, dok Y-os prikazuje vrijednost *TPR*

bilo bi poželjno da je taj broj barem red veličine veći. Iako smo za učenje reprezentacije raspolagali skupom od 500 tisuća sekvenci, zbog nedostatka hardverskih resursa morali smo se ograničiti na mali broj sekvenci. Osim toga, definirani tip problema rekurentne mreže, predviđanje prave od permutirane sekvence, mogao bi biti prejednostavan problem za učenje kompleksne reprezentacije sekvence. Unatoč tome, već s jednostavnim tipom problema i malim skupom podataka uspjeli smo dobiti reprezentacije koje, u slučaju problema predviđanja familija i određivanja termofilnosti sekvence, daju zadovoljavajuće rezultate. Čak smo, za problem predviđanja familije proteina, s reprezentacijama rekurentnih mreža uspjeli određene klase raspoznati bolje od reprezentacije dobivene sumiranjem trigrama (vidi tablicu 5.1).

Kao konačno razmatranje, treba spomenuti i rezultate nedavnih istraživanja vezanih uz performanse *LSTM* rekurentnih mreža nad dugačkim sekvencama te nove pristupe koji omogućavaju njihova poboljšanja. U potpoglavlju 2.3 spomenuto je kako standardna arhitektura rekurentnih mreža ima poteškoća s učenjem strukture sekvence na velikim udaljenostima, dok bi *LSTM* arhitektura trebala omogućiti učenje udaljenih međuzavisnosti unutar sekvence. Ipak, u praksi se pokazalo da čak i *LSTM* arhitektura ima problema s modeliranjem udaljenih interakcija. Ti problemi posebno su uočeni u području obrade prirodnog jezika sa zadatkom prevođenja teksta između različitih jezika. U radu [3] kao problem navode činjenicu da rekurentna mreža pohranjuje informaciju o cijeloj sekvenci u fiksnom vektoru, iz kojega kasnije, kao izlaz, reproducira prijevod. Ono što se pokazuje problematičnim jest činjenica da ulazne sekvence mogu biti vrlo dugačke zbog čega nije potrebno pohraniti čitavu informaciju, nego omogućiti mreži da sama nauči koji dijelovi sekvence su *važniji*. U radu smatraju da se takvim pristupom, koji nazivaju *attention*, postižu bolji rezultati nad dugačkim sekvencama. Pristup *attention* su testirali na problemu prevođenja teksta s engleskog jezika na francuski te su postigli puno bolje rezultate, a posebno u slučaju dugačkih rečenica. Nadalje, u radu [12] za poboljšanje performansi nad dugačkim sekvencama navode pristup nevezane memorije (engl. *unbounded memory*) koja za implementaciju algoritama koristi strukture kao što su stog i red. Eksperimentalni rezultati na problemu učenja *sequence to sequence* arhitekture pokazuju poboljšanje u odnosu na standardne *LSTM* arhitekture. Budući da su proteinske sekvence vrlo dugačke, navedeni pristupi mogli bi biti od iznimne važnosti za učenje reprezentacija proteinskih sekvenci.

Iz svega navedenog, rezultate ovog rada smatramo dobrim temeljem za nastavak budućeg istraživanja. Uz treniranje rekurentne mreže kompleksnijim problemom uz dovoljno veliki skup podataka s poboljšanom arhitekturom modela uz pristupe *attention* i nevezane memorije, mogli bismo dobiti reprezentacije koje bi dale bolje rezultate od dosad postignutih. Time bismo mogli dobiti i generalnu reprezentaciju koja bi se mogla koristiti za razne biološke probleme.

Bibliografija

[1] N. H. Andersen, *Protein Structure, Stability, and Folding. Methods in Molecular Biology. Volume 168 Edited by Kenneth P. Murphy*, 2001.

[2] E. Asgari i M. Mofrad, *Continuous distributed representation of biological sequences for deep proteomics and genomics*, PloS one **10** (2015), br. 11, e0141287.

[3] D. Bahdanau, K. Cho i Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv preprint arXiv:1409.0473 (2014).

[4] Y. Bengio, *Deep learning of representations for unsupervised and transfer learning*, Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 2012, str. 17–36.

[5] Y. Bengio, A. Courville i P. Vincent, *Representation learning: A review and new perspectives*, IEEE transactions on pattern analysis and machine intelligence **35** (2013), br. 8, 1798–1828.

[6] A. Blum i R. Rivest, *Training a 3-node neural network is NP-complete*, Advances in neural information processing systems, 1989, str. 494–501.

[7] UniProt Consortium et al., *UniProt: a hub for protein information*, Nucleic acids research (2014), gku989.

[8] T. Dietterich, *Machine learning for sequential data: A review*, Structural, syntactic, and statistical pattern recognition (2002), 227–246.

[9] R. Donev, *Advances in Protein Chemistry and Structural Biology*, sv. 94, Academic Press, 2014.

[10] A. K. Dunker, I. Silman, V. N Uversky i J. L. Sussman, *Function and structure of inherently disordered proteins*, Current opinion in structural biology **18** (2008), br. 6, 756–764.

[11] I. Goodfellow, Y. Bengio i A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[12] E. Grefenstette, K. M. Hermann, M. Suleyman i P. Blunsom, *Learning to transduce with unbounded memory*, Advances in Neural Information Processing Systems, 2015, str. 1828–1836.

[13] S. Hochreiter i J. Schmidhuber, *Bridging long time lags by weight guessing and "Long Short-Term Memory"*, Spatiotemporal models in biological and artificial systems **37** (1996), 65–72.

[14] L. Hunter, *Molecular biology for computer scientists*, Artificial intelligence and molecular biology (1993), 1–46, `https://mitpress.mit.edu/sites/default/files/titles/content/9780262581158_sch_0001.pdf`.

[15] V. Kunin, I. Cases, A. J. Enright, V. de Lorenzo i C. A. Ouzounis, *Myriads of protein families, and still counting*, Genome biology **4** (2003), br. 2, 401.

[16] T. K. Lee i T. Nguyen, *Protein Family Classification with Neural Networks*, (2016).

[17] Z. C. Lipton, J. Berkowitz i C. Elkan, *A critical review of recurrent neural networks for sequence learning*, arXiv preprint arXiv:1506.00019 (2015).

[18] L. Maaten i G. Hinton, *Visualizing data using t-SNE*, Journal of Machine Learning Research **9** (2008), br. Nov, 2579–2605.

[19] A. Markotić, *Intracellular Traffic & Sorting of Proteins*, Harperova ilustrirana biokemija, Medicinska naklada, 2011.

[20] C. P. McKay, *What is life—and how do we search for it in other worlds?*, PLoS Biol (2004), br. 9, e302.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado i J. Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems, 2013, str. 3111–3119.

[22] C. Olah, *Deep Learning, NLP, and Representations*, `http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/`, posjećena 11.04.2017.

[23] J. Pennington, R. Socher i C. Manning, *Glove: Global vectors for word representation*, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, str. 1532–1543.

[24] M. Schuster i K. K. Paliwal, *Bidirectional recurrent neural networks*, IEEE Transactions on Signal Processing **45** (1997), br. 11, 2673–2681.

[25] T. Zhang, *An introduction to support vector machines and other kernel-based learning methods*, AI Magazine **22** (2001), br. 2, 103.

Sažetak

Informacije o primarnoj strukturi proteina koriste se kao ulazni podaci u mnogim klasifikacijskim problemima iz područja biologije. Pri tome se značajke standardno konstruiraju uprosječivanjem svojstava aminokiselina u sekvenci. Međutim, u takvoj reprezentaciji gube se informacije o slijedu aminokiselina u sekvenci. Umjesto ručnog konstruiranja značajki koje zahtijeva dodatno vrijeme te poznavanje specifičnog problema koji se rješava, duboke neuronske mreže omogućavaju automatsko učenje reprezentacije u obliku realnog vektora fiksne dimenzije. Takve reprezentacije mogu se koristiti za rješavanje različitih klasifikacijskih problema. Budući da danas raspolažemo velikim brojem sekvenciranih genoma, taj pristup je iznimno prikladan za proteinske sekvence. U ovom radu bavimo se modeliranjem realnih vektorskih reprezentacija proteinskih sekvenci kroz nekoliko faza. U prvoj fazi proteinske sekvence dijelimo u sekvence trigrama aminokiselina i povezujemo problem traženja reprezentacija trigrama s problemom traženja reprezentacija riječi koristeći metode *word2vec* i *Glove*. Naučene reprezentacije trigrama koristimo u drugoj fazi za treniranje dvosmjerne *LSTM* rekurentne mreže na problemu prepoznavanja originalne od permutirane sekvence. U tu svrhu definiramo tri različita načina permutiranja koji odgovaraju različitoj težini problema. Na problemu razlikovanja stvarne od permutirane sekvence, rekurentna mreža s *Glove* reprezentacijama trigrama postigla je visoku točnost (> 93 %) već nakon 30 epoha. Za razliku od *Glove* reprezentacija, *word2vec* je postigla nešto lošije rezultate (> 72 %). Skriveni sloj *LSTM* mreže odgovara reprezentaciji sekvence koju smo zatim usporedili s reprezentacijama dobivenih sumom reprezentacija trigrama. U zadnjoj fazi, testirali smo kvalitete dobivenih reprezentacija na tri različita klasifikacijska problema: problemu predviđanja familije proteina, razlikovanja termofilnih od netermofilnih proteina te predviđanju klase ribosoma. Rezultati pokazuju da reprezentacije dobivene sumiranjem daju najbolje rezultate, dok *Glove* daje bolje rezultate od *word2vec*. Nadalje, uočava se korelacija između težine problema rekurentne mreže i kvalitete reprezentacija. Iako je zbog hardverskih ograničenja mreža trenirana na malom skupu podataka, rezultati pokazuju da postoje primjeri reprezentacija dobivenih rekurentnim mrežama koji postižu veću točnost od reprezentacija dobivenih sumom.

Summary

Protein sequences are used as features in many biological classification problems. Those sequences are usually represented using biophysical properties of amino acids the sequence is built from. However, that kind of representation does not include information about the order of amino acids in the original sequence. Deep neural networks enable learning dense vector representations of sequences automatically, instead of building features by hand which is time-consuming and requires domain knowledge. These representations can be used to solve different classification tasks. In this work, we propose different methods of extracting dense vector representations from protein sequences through couple of phases. In the first phase, we divide each protein sequence into trigrams of amino acids. In order to map trigrams to distributed vectors, we relate trigrams to words and use natural language processing models *word2vec* and *Glove*. In the second phase, learned representations of trigrams are used as an input to bidirectional *LSTM* recurrent network to differentiate between a real protein sequence and a permuted sequence. For this purpose, we define three different permutation methods corresponding to distinct levels of complexity. On the hardest permutation problem, recurrent networks in case of *Glove* trigram representations achieve high accuracy (> 93 %) after only 30 epochs. On the other hand, recurrent networks with *word2vec* trigrams as inputs reach lower accuracy (> 72 %). Hidden layer of trained *LSTM* network corresponds to sequence representation which we have compared to representations obtained by sum of trigram representations. The final phase was used to test quality of all eight representations by solving three different classification tasks: protein family classification, distinguishing between thermophilic and non-thermophilic protein and predicting class of ribosomes. Results suggest that representations obtained from trigram summation outperform those from recurrent network and that *Glove* recurrent representations exceed those from *word2vec*. Furthermore, we have noticed a correlation between complexity of permutation task and achieved results. Although hardware limitations allow us to train the network only on small subset of the original dataset, there are individual cases where representations from recurrent networks perform better than summation representation.

Životopis

Rođena sam i živim u Zagrebu gdje sam završila srednju školu V. gimnazija s odličnim uspjehom. Završila sam preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta s prosjekom ocjena 4.9, na kojem sam, 2015. godine upisala diplomski sveučilišni studij Računarstvo i matematika. Do sada sam na diplomskom studiju ostvarila prosjek ocjena 5.0. Kroz studij sam držala demonstrature iz kolegija Matematička logika na diplomskom, te Elementarna matematika 1 i 2 na preddiplomskom studiju. Za vrijeme studija bila sam nagrađena stipendijama, 2013. godine stipendijom Sveučilišta u Zagrebu za 10 % najboljih studenata, dok sam od 2014. godine nagrađena stipendijom Grada Zagreba. Osim toga, 2017. godine sam dobila nagradu za najuspješnije studente završnih godina diplomskih studija te 2015. godine za najuspješnije studente završnih godina preddiplomskih studija.

Tijekom studija radila sam na mnogo zanimljivih projekata. Zajedno s kolegom Filipom Srnecom, pod vodstvom dr. sc. Tomislava Šmuca i doc. dr. sc. Goranke Nogo, radila sam na projektu *Prepoznavanje glazbenih akorda koristeći tehnike strojnog učenja*, za koji smo, akademske godine 2016./2017., nagrađeni Rektorovom nagradom. S kolegama Filipom Srnecom i Tomislavom Bujanovićem radila sam na raznim projektima među kojima bih istaknula projekt *Problem bojanja grafova i Sudoku* u sklopu kolegija Meta-heuristike. U navedenom projektu smo, koristeći evolucijske algoritme pokušali riješiti problem bojanja grafova uz primjenu na zagonetku Sudoku. Osim grupnih projekata, radila sam i na mnogim samostalnim projektima kao što je aplikacija *Uzmi broj* u sklopu kolegija Programiranje za suvremene procesore te na projektima koji su uključivali implementaciju paralelnih algoritama u sklopu kolegija Uvod u paralelno računanje i Primjena paralelnih računala.

U studenom 2017. godine zaposlila sam se u tvrtki HashCode kao softverski inženjer. U toj tvrtki radila sam kao student na projektu od travnja do srpnja 2017. godine. Od kolovoza do studenog 2017. godine radila sam kao softverski inženjer u tvrtki Google u Londonu kao dio tima Android Search and Infrastructure, dok sam ljeto 2016. godine, od srpnja do listopada, provela u tvrtki Google u Zurichu kao dio tima Hijacking and Risk analysis. Na preddiplomskom studiju sam zajedno s kolegom Filipom Srnecom radila za tvrtke PJR (Projekt jednako razvoj) d.o.o. i IDE 3 d.o.o. na održavanju i administraciji

baze podataka te na savjetovanju i kooperaciji za razvoj CRM softvera.

Osim angažmana u poslovnom i studentskom svijetu, u slobodno vrijeme se bavim plesanjem i raznim sportskim aktivnostima kao što su trčanje i planinarenje.