Označavanje vrsta riječi pomoću neuronskih mreža

Juričić, Josipa

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, Faculty of Science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:035081

Rights / Prava: Attribution-NoDerivatives 4.0 International/Imenovanje-Bez prerada 4.0 međunarodna

Download date / Datum preuzimanja: **2024-06-26**

Repository / Repozitorij:

Repository of Faculty of Science

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

OZNAČAVANJE VRSTA RIJEČI POMOĆU NEURONSKIH MREŽA

Josipa Juričić

Split, rujan 2022.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za Informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

OZNAČAVANJE VRSTA RIJEČI POMOĆU NEURONSKIH MREŽA

Josipa Juričić

SAŽETAK

Označavanje vrsta riječi jedan je od temeljnih zadataka u području obrade prirodnog jezika. Ono samo po sebi ne rješava određeni problem u obradi prirodnog jezika, ali je preduvjet za mnoge druge procese kao što su analiza sentimenta, prevođenje teksta ili prepoznavanje govora. Označavanja temeljena na pravilima, transformacijama ili stohastičkim metodama opisana su i uspoređena sa korištenjem neuronskih mreža u svrhu označavanja vrsta riječi. Implementirane su RNN i LSTM arhitekture neuronske mreže čiji modeli mogu provoditi automatsko označavanje vrsta riječi za dani tekst pisan hrvatskim jezikom. Kako bi se utvrdila efikasnost uporabe modela neuronske mreža za ovakav zadatak, modeli su testirani i evaluirani. Cilj je prikazati primjenu neuronskih mreža za razvoj označivača vrsta riječi hrvatskog jezika.

Ključne riječi: obrada prirodnog jezika, označavanje vrsta riječi, POS, hrvatski jezik, neuronske mreže, povratne neuronske mreže, RNN, LSTM

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 70 stranica, 33 grafička prikaza, 2 tablice i 48 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Branko Žitko,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Branko Žitko,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Dr. sc. Ani Grubišić, *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Ines Šarić-Grgić, *viši asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **rujan 2022.**

Basic documentation card

Thesis

University of Split
Faculty of Science
Department of Informatics
Ruđera Boškovića 33, 21000 Split, Croatia

PART-OF-SPEECH TAGGING OF THE CROATIAN LANGUAGE WITH NEURAL NETWORKS

Josipa Juričić

ABSTRACT

Part-of-speech tagging is one of the main tasks of natural language processing. On its own, it does not solve a specific problem of natural language processing but it is a part of preprocessing in sentiment analysis, text translation or speech recognition. In the first part of this thesis the rule-based, transformation based and stochastic methods of part-of-speech tagging are described and compared to neural network methods. As a practical part of the thesis, RNN and LSTM neural network models are implemented for an automatic part-of-speech tagger of the Croatian language. To check the efficiency of using the neural network approach for this task, models are tested and evaluated. The aim of this thesis is to implement neural networks for an automatic part-of-speech tagger of the Croatian language.

Key words: natural language processing, part-of-speech tagging, POS, Croatian language, neural networks, recurrent neural network, RNN, LSTM

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 70 pages, 33 figures, 2 tables and 48 references

Original language: Croatian

Mentor: **Branko Žitko, Ph.D.** *Associate Professor of Faculty of Science, University of Split*

Reviewers: **Branko Žitko, Ph.D.** *Associate Professor of Faculty of Science, University of Split*
Ani Grubišić, Ph.D. *Associate Professor of Faculty of Science, University of Split*
Ines Šarić-Grgić *Assistant Professor of Faculty of Science, University of Split*

Thesis accepted: **September 2022.**

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam diplomski rad s naslovom OZNAČAVANJE VRSTA RIJEČI POMOĆU NEURONSKIH MREŽA izradila samostalno pod voditeljstvom izv. prof. dr. sc. Branka Žitka. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u diplomskom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Josipa Juričić

Zahvaljujem se svojim roditeljima na podršci tijekom studija te na svemu što su mi kroz život omogućili, a bez čega danas ne bih bila to što jesam. Također, zahvaljujem i svom mentoru izv. prof. dr. sc. Branku Žitku na pomoći i vodstvu prilikom izrade ovog rada.

Sadržaj

Uvod 1

1 Označavanje vrsta riječi u obradi prirodnog jezika 2

1.1. Označavanje pravilima i transformacijama 4

1.2. Označavanje stohastičkim metodama 6

2 Označavanje vrsta riječi neuronskim mrežama 12

2.1. Reccurent Neural Network (RNN) 14

2.2. Long Short Term Memory (LSTM) 18

3 Označavanje vrsta riječi hrvatskog jezika 23

3.1. Vrste riječi u hrvatskom jeziku 25

3.2. Pregled istraživačkih radova 26

4 Primjena RNN i LSTM modela za označavanje vrsta riječi 31

4.1. Korpus 33

4.2. Vektorizacija 35

4.3. Treniranje 39

5 Evaluacija RNN i LSTM modela 44

5.1. Povijest treniranja 44

5.2. Preciznost, točnost i F1 47

Zaključak 49

Literatura 50

Sažetak 53

Summary 54

Skraćenice 55

Privitak 56

Uvod

Umjetna inteligencija može se definirati kao znanstveno područje koje istražuje načine kako postići da se računalo inteligentno ponaša [45], a uključuje koncepte kao što su strojno učenje, obrada prirodnog jezika, računalni vid i drugi. Obrada prirodnog jezika (engl. *natural language processing*, skraćeno NLP) je dio umjetne inteligencije, a koristi računalne tehnike u svrhu učenja, razumijevanja i generiranja ljudskog jezika. Rani računalni pristupi istraživanju jezika bili su usmjereni na automatizaciju analize jezične strukture jezika i razvoj temeljnih tehnologija kao što su strojno prevođenje, prepoznavanje govora i sinteza govora. [19] Današnji istraživači usavršavaju i koriste ovakve alate u stvarnim aplikacijama, stvarajući alate za pretvaranje teksta u govor, tražeći informacije o zdravlju ili financijama na društvenim mrežama i identificirajući osjećaje i emocije prema proizvodima i uslugama. [19] Označavanje vrsta riječi jedan je od temeljnih zadataka u obradi prirodnog jezika. U procesu obrade prirodnog jezika svakoj se riječi dodjeljuje oznaka koja označuje kojoj vrsti riječi ili dijelu govora ta riječ pripada te taj postupak nazivamo označavanje vrsta riječi (engl. *part-of-speech tagging*, skraćeno POS označavanje).

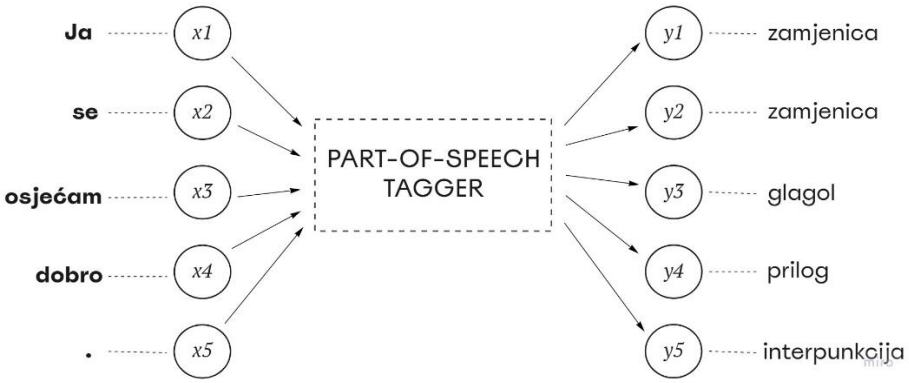
U sljedećim poglavljima napravit će se pregled poznatih metoda u obradi prirodnog jezika koje se uobičajeno koriste u svrhu označavanja vrsta riječi te će se pokazati potreba za korištenjem neuronskih mreža i metoda dubokog učenja. Nakon što se objasni što su neuronske mreže te koje se arhitekture neuronskih mreža koriste u svrhu označavanja vrsta riječi, napravit će se pregled dostupnih istraživanja za njihovu primjenu.

S obzirom na to da se znanstveni članci i istraživanja o primjeni neuronskih mreža u POS označavanju u velikom broju odnose na engleski jezik, zanimljivom se pokazala ideja istraživanja označavanja vrsta riječi hrvatskog jezika. Stoga se treće poglavlje prvo bavi vrstama riječi u hrvatskom jeziku, a potom i pregledom literature o dostupnim korpusima hrvatskog jezika te označivačima vrsta riječi za hrvatski jezik.

Na kraju rada implementirat će se i modeli dviju arhitektura dubokih neuronskih mreža – RNN i LSTM. Potom će se i usporediti točnosti ovih modela za POS označavanje, a sve kako bi se odredilo koji je pristup najuspješniji. Cilj ovog rada je prikazati primjenu neuronskih mreža za implementaciju označivača vrsta riječi hrvatskog jezika.

1 Označavanje vrsta riječi u obradi prirodnog jezika

Obrada prirodnog jezika strojevima ne daje samo mogućnost čitanja, već i razumijevanja i interpretacije pisanog ili izgovorenog teksta. [10] Potreba za obradom prirodnog jezika javlja se iz činjenice da računalo ne razumije prirodni ljudski jezik, već koristi strojno razumljiv format (brojke, a ne slova!). [45] Strojno prevođenje, automatsko sažimanje teksta, automatsko prepoznavanje govora i analiza sentimenta samo su neki od područja obrade prirodnog jezika. Označavanje vrsta riječi postupak je u obradi prirodnog jezika koji svakoj riječi u tekstu pridjeljuje pripadajuću joj oznaku vrste riječi. Oznake se pridjeljuju s obzirom na značenje riječi i njezinu vezu sa susjednim riječima u rečenici. Označivač vrsta riječi ili POS označivač (engl. *part-of-speech tagger*) kao ulaz prima niz riječi: x1, x2, x3, …, xn i označeni skup podataka (ili korpus) iz kojeg "uči" kako prepoznati različite vrste riječi. Kao izlaz daje se niz oznaka: y1, y2, …, yn, gdje yi odgovara točno jednom ulazu xi (Slika 1). [24]



Slika 1 Zadatak POS označivača: *nizu riječi pridružiti niz POS oznaka*

POS označavanje samo po sebi ne rješava određeni problem u obradi prirodnog jezika, ali je preduvjet za mnoge druge procese. Ovaj je zadatak neophodan prije drugih procesa kao što su analiza sentimenta, prevođenje teksta, provjera gramatike, prepoznavanje govora i mnogih drugih. Ako je POS označavanje loše izvedeno to negativno utječe na sve zadatke koji slijede nakon njega te iz tog razloga mnogi autori navode važnost pronalaska modela koji će najuspješnije riješiti ovaj zadatak.

Dodjeljivanje POS oznake riječi pruža dodatne informacije o samoj riječi, ili o susjednim joj riječima, što može pridonijeti određivanju konteksta riječi u rečenici. [24] Značenje riječi u rečenici ovisi o kontekstu u kojem se nalazi, a kontekst se može odrediti pomoću POS oznaka susjednih riječi. Uz pomoć dobro definiranih algoritama ili dobro uvježbanih neuronskih

mreža, riječi često nisu definirane općenito kao zamjenice, već primjerice, kao osobne (npr. ja, ti, ona, …) ili posvojne (npr. moj, tvoj, naš, …) zamjenice. Razumijevanje ovakvih detaljnijih razlika ključno je za određivanje POS oznaka susjednih riječi gdje, primjerice, osobne zamjenice često slijede glagoli, a poslije posvojnih zamjenica slijede imenice. Pomoću poznavanja POS oznaka susjednih riječi, lako je riješiti problem višeznačnica (riječi koje imaju više značenja). Uzmimo za primjer riječ *vodi* u rečenicama: „*On vodi tim do pobjede.*“ i „*On ide prema onoj vodi.*“ (Slika 2). S obzirom na to da u hrvatskom jeziku osobne zamjenice često slijede glagoli, a poslije posvojnih zamjenica slijede imenice, sustav lako može riješiti problem višeznačnosti:

a. u prvoj rečenici riječ *vodi* je glagol (prezent glagola voditi);
b. u drugoj rečenici riječ *vodi* je imenica (voda - nominativ jednine, vodi - dativ jednine).

On	vodi	tim	do	pobjede	.
zamjenica	glagol	prijedlog	prijedlog	imenica	interpunkcija

On	ide	prema	onoj	vodi	.
zamjenica	glagol	prijedlog	zamjenica	imenica	interpunkcija

Slika 2 Primjer različitog značenja riječi u ovisnosti o kontekstu

Sve navedeno pomaže u stvaranju jezičnih modela koji se koriste u sustavima za sintezu govora (engl. *speech synthesis*) ili u sustavima za prepoznavanje govora. (engl. *speech recognition*). [24] Kod stvaranja sustava za sintezu govora hrvatskog jezika problem mogu stvarati homografi ili istopisnice (riječi koje se isto pišu, a različito izgovaraju). Jedan od primjera su riječi *bît* (glavni smisao) i *bȉt* (binarna znamenka), gdje različit naglasak na slovu *i* ima razlikovnu ulogu u značenju riječi u govoru. Ako se riječ *bît* označi POS oznakom koja je označuje kao imenicu ženskog roda, a riječ *bȉt* oznakom koja je označuje kao imenicu muškog roda, sustav će za prvu riječ koristiti dugosilazni naglasak na slovu *i*, a kratkosilazni naglasak za drugu riječ. U ovakvim, i sličnim, slučajevima, poznavanje POS oznake omogućuje stvaranje prirodnijih izgovora u sustavima za sintezu govora i veću točnost u sustavima za prepoznavanje govora. [24]

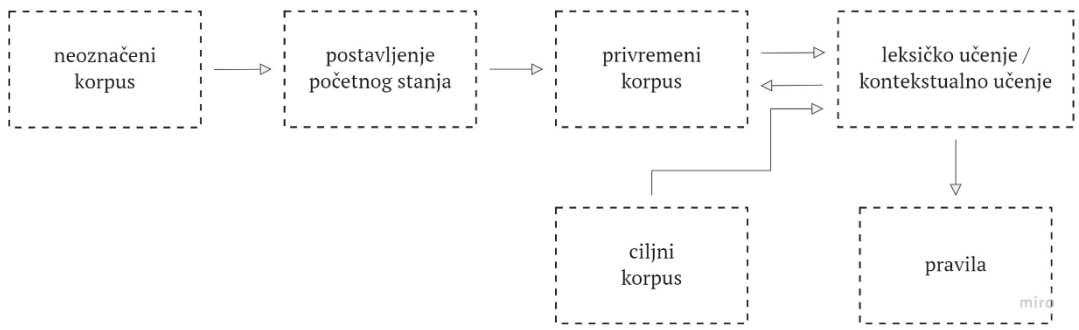
U ovom će se poglavlju opisati klasične metode označavanja vrsta riječi temeljene na pravilima i na transformacijama te stohastičke metode. POS označivači temeljeni na

pravilima dodjeljuju POS oznaku riječi na temelju ručno stvorenih ili naučenih jezičnih pravila, dok stohastički označivači određuju POS oznaku riječi u danom kontekstu na temelju vjerojatnosti izračunatih iz označenog korpusa. [18] S druge strane, kombinacija probabilističkih metoda i metoda temeljenih na pravilima je označavanje vrsta riječi temeljeno na transformacijama.

1.1. Označavanje pravilima i transformacijama

Najstariji sustav označavanja vrsta riječi je sustav koji je koristio pristup temeljen na pravilima (engl. *rule-based*), a kako bi se riječima dodijelile oznake vrste riječi primjenjivao se skup pisanih pravila te kontekstualne informacije o riječi. [28] U najvećem broju slučajeva označavanja pomoću pravila izrađuje se rječnik s mogućim oznakama za svaku riječ, a pravila su ručno pisana, naučena ili oboje. [13]

Eric Brill je 1992. godine predstavio označivač vrsta riječi koji se temelji na pravilima ili, kako ih on naziva, transformacijama. [36] Za rad njegovog sustava za POS označavanje bio je potreban korpus riječi s pravilno označenim vrstama iz kojeg se uzimaju informacije. Na osnovu tih informacija sustav se trenira kako zaključiti koja je najvjerojatnija POS oznaka riječi. [36] Nakon što se istrenira, POS označivač se može koristiti za označavanje novih, neoznačenih korpusa. [36] Najvažnija stvar koju treba napomenuti o Brillovom označivaču je da pravila nisu ručno izrađena, već se umjesto toga pronalaze pomoću priloženog korpusa označenih riječi. Jedino potrebno, uz korpus riječi označenih POS oznakama, je skup predložaka pravila koje model može koristiti za osmišljavanje novih značajki. Brillov se POS označivač može definirati kao označivač temeljen na transformacijama (oznaka se dodjeljuje svakoj riječi i mijenja se korištenjem unaprijed definiranih pravila) koji uči na temelju pronalaska i minimiziranja pogrešaka. [36]



Slika 3 Brillov označivač

Slika 3 prikazuje slijed Brillovog sustava za označavanje. Prvi je korak slanje neoznačenog korpusa ili teksta označivaču koji svakoj riječi postavlja početnu oznaku. Algoritam svakoj riječi prvo dodjeljuje najvjerojatniju oznaku vrste riječi, tj. najčešću oznaku za tu riječ u danom tekstu. Tim se postupkom stvara privremeni korpus koji se uspoređuje s ciljnim korpusom (popis riječi koje sadrže informacije o učestalostima oznaka u ručno označenom korpusu). [36] Brillovo označavanje koristi dva modula: leksički modul i kontekstualni modul. [36] Prvo, leksički modul koristi učenje temeljeno na pogreškama za izradu pravila leksičkog označavanja. Zatim kontekstualni modul, također koristeći učenje temeljeno na pogreškama, proizvodi pravila kontekstualnog označavanja. [36] Za svaki put kada privremeni korpus prolazi kroz proces leksičkog ili kontekstualnog učenja, sustav proizvodi jedno novo pravilo koje najviše smanjuje grešku označavanja. Svako pravilo se sastoji od dva dijela: uvjeta (koji se sastoji od okidača (*trigger*) i moguće oznake) i rezultirajuće oznake. Pravila se instanciraju iz skupa unaprijed definiranih transformacijskih predložaka:

```
If Trigger, then change the tag X to the tag Y       a)
If Trigger, then change the tag to the tag Y         b)
```

Prvi tip transformacijskog predloška (a) označuje da ako se pravilo aktivira na riječi s trenutnom oznakom *X*, tada pravilo zamjenjuje trenutnu oznaku s oznakom *Y*. [36] Drugi predložak (b) znači da ako se pravilo aktivira na riječi, a bez obzira na trenutnu oznaku tada pravilo označava ovu riječ oznakom *Y*. Skup svih pravila generira se iz svih mogućih instanci ovakvih unaprijed definiranih predložaka. [36] Prilikom korištenja pravila za označavanje neoznačenog teksta prvo označivač početnog stanja na nepoznate riječi (tj. riječi koje nisu u leksikonu) stavlja početne oznake, potom primjenjuje uređena leksička pravila, a zatim i uređena kontekstualna pravila na cijeli korpus čime se dobije označeni tekst. [36] Ovakvo označavanje temeljeno na transformacijama kombinacija je metode temeljene na vjerojatnostima (u početku se svakoj riječi pridodaje najvjerojatnija oznaka) te metode temeljene na pravilima (algoritam za učenje pravila smanjuje greške uvedene metodom temeljenom na vjerojatnostima). [36]

Acedański (2010) predstavlja Brillov označivač za označavanje vrsta riječi koji je prilagođen za morfološki bogate jezike, a posebice za poljski jezik. [1] U radu se navode značajna poboljšanja u usporedbi s prethodnim radovima, a za poljski jezik njegov označivač postiže točnost od 92,44%. Ovakvo je poboljšanje postignuto nadograđivanjem originalnog Brillovog označivača transformacijskim predlošcima koji uzimaju u obzir prefikse i sufikse kod leksičkog označavanja te uvođenjem paralelnog izvršavanja. [1]

Hjouj et al. (2016) izrađuju POS označivač za arapski jezik u kojem koriste označavanje pomoću pravila. Njihov POS označivač radi u dvije faze: prva faza se sastoji od leksičkog analizatora koji sadrži sve arapske čestice (uključujući i prijedloge, priloge, veznike, upitne čestice, iznimke i usklike), a potom morfološka faza koristi informacije kao što su uzorci riječ i njezini prefiksi i sufiksi za određivanje klase riječi. Testiranjem POS označivača na uzorku od 793 riječi rezultati su pokazali uspješno označivanje 679 riječi. Autori za daljnji rad navode potrebu povećanja skupa oznaka za označavanje kao i skupa pravila označavanja. [20]

Li et al. (2021) u svom radu predlažu novi pristup poboljšanju točnosti POS označivača temeljenih na pravilima i transformacijama. Njihov pristup se sastoji od toga da se za svaku riječ najprije odrede sve moguće POS oznake (bez razmatranja konteksta), a zatim se primjenom pravila uklanjaju neprihvatljive oznake. Ovim načinom se broj mogućih POS oznaka svake riječi može smanjiti na samo jednu oznaku koja se smatra točno pridijeljenom oznakom. Autori navode da rezultati njihovog pristupa daju čak i bolje rezultate nego metode kao što je dvosmjerna LSTM neuronska mreža (potpoglavlje 2.2) za koju prethodna istraživanja smatraju za jednim od najboljih pristupa. [33]

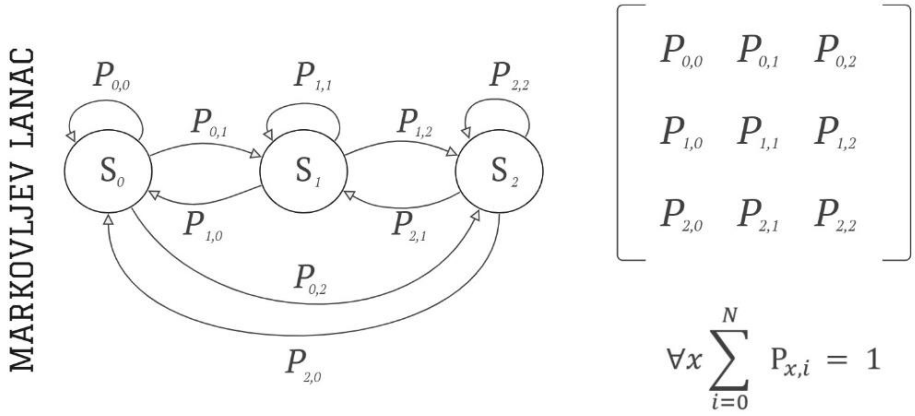
1.2. Označavanje stohastičkim metodama

Skriveni Markovljev model (engl. *Hidden Markov Model*, skraćeno HMM) najpoznatija je metoda stohastičkog označavanja. S obzirom na niz (riječi, slova ili rečenica) HMM izračunava distribuciju vjerojatnosti preko niza oznaka i predviđa najbolji niz oznaka. [24] Skriveni Markovljevi modeli imaju različite primjene kao što su prepoznavanje govora, obrada signala te neki zadaci niske razine obrade prirodnog jezika kao što su POS označavanje, grupiranje fraza i izvlačenje informacija iz dokumenata.

HMM se temelji na Markovljevim lancima. Markovljev lanac je model koji opisuje slijed potencijalnih događaja u kojima je vjerojatnost događaja ovisna samo o stanju koje je postignuto u trenutnom događaju. [25] Markovljevi lanci, nazvani po Andreju Markovu, mogu se zamisliti kao sustav koji ide iz jednog stanja u drugo tvoreći lanac. Markovljevi lanci imaju Markovljevo svojstvo koje kaže da vjerojatnost prelaska u bilo koje sljedeće stanje ovisi samo o trenutnom stanju, a ne o prethodnim stanjima. [25]

Slika 4 prikazuje Markovljev lanac u kojem su stanja S_0, S_1, i S_2 prikazana kao čvorovi usmjerenog grafa, a prijelazne vjerojatnosti $P_{0,0}$, $P_{0,1}$, …, $P_{2,2}$ su označene bridovima. U

ovakvom grafu sve prijelazne vjerojatnosti koje izlaze iz jednog čvora moraju u zbroju biti jednake 1 (jednadžba sa slike 4).

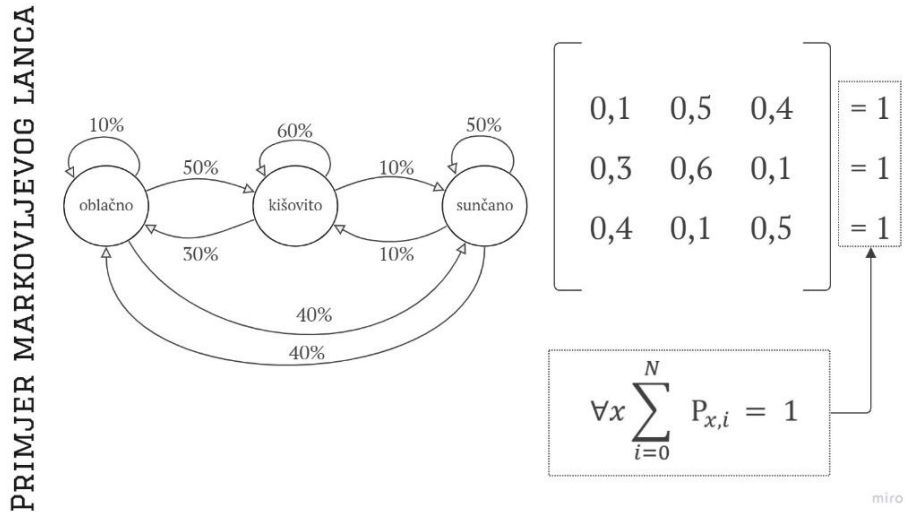


Slika 4 Model Markovljevog lanca

Slika 5 prikazuje primjer Markovljevog lanca gdje postoje tri stanja koja predstavljaju vrijeme dana (*oblačno*, *kišovito* i *sunčano*) te prijelazne vjerojatnosti koje pretpostavljaju vrijeme sljedećeg dana s obzirom na vrijeme tekućeg dana. [25] Primjerice, ako je danas sunčano vrijeme, vjerojatnost da će sutra biti sunčano je 50%, kišovito 10%, a oblačno 40%. Vjerojatnost da će u srijedu biti oblačno, ako je danas (ponedjeljak) sunčano, računa se [25]:

- *sunčano* (danas) – *sunčano* (utorak) – *oblačno* (srijeda): 0.5 x 0.4 = 0.2
- *sunčano* (danas) – *kišovito* (utorak) – *oblačno* (srijeda): 0.1 x 0.3 = 0.03
- *sunčano* (danas) – *oblačno* (utorak) – *oblačno* (srijeda): 0.4 x 0.1 = 0.04

Iz čega slijedi da je ukupna vjerojatnost oblačne srijede 27% (0.2 + 0.03 + 0.04 = 0.27). [25]

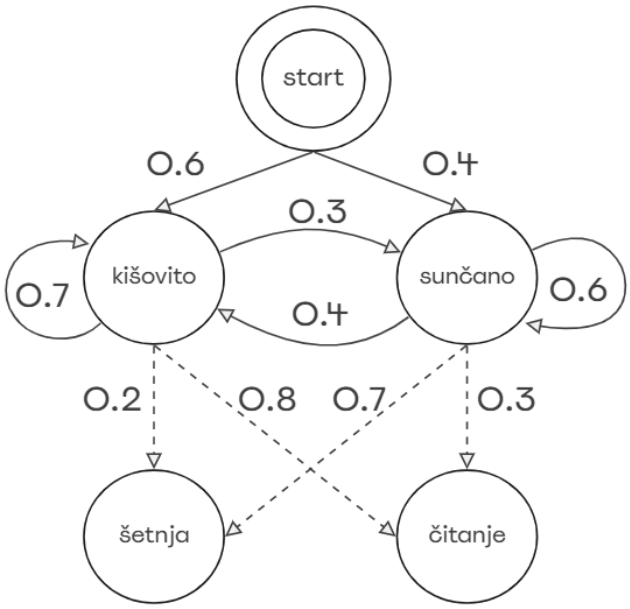


Slika 5 Primjer Markovljevog lanca

Markovljev lanac prikazuje stanja i njima pridružene prijelazne vjerojatnosti. Međutim, u skrivenom Markovljevom modelu, Markovljev lanac je skriven, ali možemo zaključiti o njegovim svojstvima kroz zadana promatrana stanja. Skriveno stanje je termin koji se daje sljedećoj mogućoj varijabli koja se ne može izravno promatrati, ali se može zaključiti promatranjem jednog ili više stanja prema Markovljevoj pretpostavci. [25] Markovljeva pretpostavka je pretpostavka da skrivena varijabla ovisi samo o prethodnom skrivenom stanju.

Slika 6 prikazuje jedan primjer skrivenog Markovljevog modela koji predstavlja proces predviđanja da će osoba određenog dana otići u *šetnju* ili ostati kod kuće *čitajući*, a ovisno o tome je li dan *kišovit* ili *sunčan*. Komponente koje ga čine su [25]:

- dva skrivena stanja: *kišovito* i *sunčano* (skrivena su jer ono što se promatra kao rezultat procesa je *šetnja* ili *čitanje*),
- redoslijed promatranja je *šetnja* pa *čitanje*,
- početna distribucija vjerojatnosti predstavljena je startnom vjerojatnošću od 60% da će vrijeme biti *kišovito* i 40% da će biti *sunčano*,
- vjerojatnost prijelaza predstavlja prijelaz jednog stanja (*kišno* ili *sunčano*) u drugo stanje s obzirom na trenutno stanje,
- vjerojatnost emisije predstavlja vjerojatnost promatranja *šetnje* ili *čitanja* s obzirom na stanja (*kišovito* ili *sunčano*).



Slika 6 Skriveni Markovljev model

Skriveni se Markovljev model sastoji od pet komponenti [25]:

$Q = q_1 q_2 \ldots q_N$	skup N stanja
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	A – matrica prijelaznih vjerojatnosti u kojoj svaki a_{ij} predstavlja vjerojatnost prelaska iz stanja *i* u stanje *j*
$O = o_1 o_2 \ldots o_T$	niz od T promatranih stanja gdje se svaki uzima iz rječnika $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	niz vjerojatnosti promatranih stanja od kojih svaki predstavlja da je vjerojatnost promatranog stanja o_t generirana iz stanja q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	početna distribucija vjerojatnosti stanja gdje je π_i vjerojatnost da će Markovljev lanac započeti u stanju *i*

Skriveni Markovljev model najčešće je korišten model za POS označavanja u kojem su promatrano stanje riječi, a skrivena stanja oznake vrsta riječi. Primjer prijelazne vjerojatnosti u ovom slučaju bio bi: *P* (*VERB* | *NOUN*) – vjerojatnost da je trenutna riječ glagol (*VERB*) ako se zna da je prethodna riječ bila imenica (*NOUN*).

Označivač vrsta riječi temeljen na HMM algoritmu sadrži dvije matrice: matricu vjerojatnosti *A* i matricu emisije *B*. [24] Matrica *A* sadrži vjerojatnosti prijelaza $P(t_i|t_{i-1})$ koje predstavljaju vjerojatnost pojavljivanja oznake (t_i) s obzirom na prethodnu oznaku (t_{i-1}). Procijenjena vjerojatnost se izračunava na način da se prebrojava koliko često iza prve POS oznake slijedi druga [24]:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \quad (1)$$

Niz vjerojatnosti promatranih stanja $P(w_i|t_i)$, predstavljaju vjerojatnost da će se određena POS oznaka (t_i) povezati s danom riječi (w_i) [24]:

$$P(w_i|\, t_i) = \frac{C(t_i, w_i)}{C(t_i)} \quad (2)$$

Zadatak određivanja niza skrivenih stanja koji odgovara nizu promatranih stanja naziva se dekodiranje. Za označavanje vrsta riječi cilj HMM dekodiranja je odabrati niz oznaka $t_1, t_2, \ldots, t_n$ koji je najvjerojatniji s obzirom na niz od *n* promatranih riječi $w_1, w_2, \ldots, w_n$ [24]:

$$\hat{t}_{1:n} = argmax_{\, t_1 \ldots t_n} P(t_1 \ldots t_n | w_1 \ldots w_n) \quad (3)$$

Primjenom Bayesovog[1] pravila, dobije se sljedeća jednadžba:

$$\hat{t}_{1:n} = argmax_{t_1 \ldots t_n} \frac{P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n)}{P(w_1 \ldots w_n)} \quad (4)$$

Konačno, pojednostavljivanjem se dobije sljedeća jednadžba:

$$\hat{t}_{1:n} = argmax_{t_1 \ldots t_n} P(w_1 \ldots w_n | t_1 \ldots t_n) P(t_1 \ldots t_n) \quad (5)$$

Algoritam koji se koristi za dekodiranje u HMM modelu naziva se Viterbijev algoritam. [24] Ideja iza Viterbijevog algoritma je korištenje dinamičkog programiranja za smanjenje broja izračuna pohranjivanjem izračuna koji se ponavljaju. Algoritam prvo postavlja matricu vjerojatnosti gdje su stupci promatrana stanja (riječi rečenice u istom poretku kao u rečenici), a redci su skrivena stanja (sve moguće POS oznake). Unutar matrice na mjestu *t,j* (gdje *t* predstavlja stupac, a *j* redak matrice) nalazi se vjerojatnost $V_{t,j}$ koja predstavlja vjerojatnost da je HMM u stanju *j* (trenutna POS oznaka) ako je poznat prethodni niz od *t* stanja (prethodne riječi za koje je već vrijednost $V_{t,j}$ izračunata). [17] $V_{t,j}$ se računa kao:

$$V_{t,j} = max\,(V_{t-1} \cdot A_{i,j} \cdot B_j\,(o_t)) \quad (6)$$

gdje je A matrica prijelaza i B matrica emisija iz HMM algoritma. [17]

Iako je HMM koristan model, pokazalo se da HMM modeli trebaju znatan broj augmentacija za postizanje visoke točnosti. [24]

Farizki Wicaksono i Purwarianti (2010) u svom radu kombiniraju nekoliko metoda za poboljšanje točnosti POS označavanja pomoću HMM algoritma. Prva metoda je korištenje stabla afiksa za uzimanje u obzir sufiksa i prefiksa riječi, druga je korištenje sljedeće POS oznake kao jedne od značajki HMM algoritma te treći način je korištenje dodatnog leksikona kako bi se ograničio broj mogućih POS oznaka koje odredi stablo afiksa. Njihove metode su postigle točnost modela od 96,50% za korpus koji više od 15 000 riječi. [16]

Qiao et al. (2015) predlažu novo proširenje HMM modela koje nazivaju raznoliki HMM model, a koji omogućuje dinamičnije sekvencijalno označavanje. Empirijske procjene referentnim skupom podataka za nenadzirano POS označavanje potvrdile su učinkovitost njihovog predloženog modela u odnosu na primjenu "običnog" HMM algoritma. [42]

[1] Bayesov teorem opisuje vjerojatnost događaja *A* ako se zna prethodni događaj *B*: $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$.

Mohamed et al. (2017) istražuju problem HMM modela koji se odnosi na to da rječnik ograničava treniranje modela jer ne može sadržavati sve riječi nekog jezika. Autori ispituju prefikse i sufikse malajskog jezika kako bi nepoznatim riječima dodijelili POS oznake na temelju lingvistički smislenih afiksa. Svoj algoritam za točnije označavanje nepoznatih riječi su integrirali u Viterbijev algoritam koji koristi HMM model: prvo se koristi predviđanje oznake temeljeno na znakovima, potom predviđanje temeljeno na morfemu te na kraju kombinacija prethodna dva koraka. Na kraju rada zaključuju da je označavanje nepoznatih riječi koje su identificirane kao nepostojeće u rječniku bolje uz pomoć POS predviđanja na temelju morfema. [37]

Lam Cing i Soe (2020) istražuju stohastički pristup POS označavanju za burmanski jezik. Zbog složene morfološke strukture koju sadrži njihov jezik navode potrebu za poboljšanje točnosti postojećih metoda POS označavanja. Posebno se usredotočuju na korištenje morfološke strukture riječi, eliminiranje nebitnih oznaka, identifikaciju slogova riječi te segmentaciju riječi u rečenici. U daljnjem radu navode potrebu za rješavanjem višeznačnosti riječi korištenjem morfoloških pravila te upotrebu većeg korpusa za poboljšanje točnosti POS označavanja. [29]

Ananda et al. (2021) u POS označavanju indonezijskog jezika koriste hibridni model koji je kombinacija skrivenog Markovljevog modela i označavanja koristeći pravila kako bi riješili problem višeznačnosti. Eksperimenti pokazuju da je njihov predstavljeni hibridni model bolji od "običnog" HMM modela s povećanjem točnosti modela od 2,03%. [8]

Kumar et al. (2022) kao glavni fokus svog istraživačkog rada navode poboljšanje osnovnih koraka predprocesiranja teksta kao što su tokenizacija teksta, normalizacija i označavanje vrsta riječi. U njihovom se istraživanju navodi važnost standardizacije neformalnih tekstova (kao što su Twitter objave) kako bi POS označavanje bilo što točnije i efikasnije. Autori koriste skriveni Markovljev model za izdvajanje ključnih leksičkih značajki iz skupa podataka o Twitter objavama kako bi identificirali kontekst objave. [27]

Pattnaik i Nayak (2022) predlažu modificirani model za POS označavanje koji se temelji na Markovljevom modelu. U izgradnji svog POS označivača koriste kombinaciju metoda strojnog učenja i lingvističkih pravila kako bi riješili problem morfološke kompleksnosti jezika i nedostupnosti korpusa dovoljne veličine. Predloženi algoritam u eksperimentiranju s Odia jezikom pokazuje veću točnost nego postojeće metode kao što su HMM model, metode temeljene na pravilima, maksimalna entropija i metoda uvjetnog slučajnog polja. [39]

2 Označavanje vrsta riječi neuronskim mrežama

Neuronska mreža niz je algoritama koji nastoje prepoznati temeljne odnose u skupu podataka kroz proces koji oponaša način na koji funkcionira ljudski mozak. U tom smislu, neuronske mreže se odnose na sustave neurona koji čine umjetnu neuronsku mrežu. [22] Neuron u umjetnoj neuronskoj mreži je matematička funkcija koja prikuplja i klasificira informacije prema specifičnoj arhitekturi. [22] Neuronska mreža sadrži slojeve međusobno povezanih čvorova, a svaki čvor je poznat kao perceptron. S obzirom na broj slojeva, neuronske mreže se dijele na jednoslojne i višeslojne.

Jednoslojni perceptron se smatra najjednostavnijim modelom umjetne neuronske mreže, a obavlja funkciju binarne klasifikacije. Jednoslojni perceptron sadrži četiri glavna parametra (Slika 7):

a. **ulazni sloj** (engl. *input layer*): sloj u kojem svaki ulaz predstavlja realnu brojčanu vrijednost $x_1, x_2, x_3, \ldots x_n$;
b. **težine i pristranost** (engl. *weight and bias*): težine $w_1, w_2, w_3, \ldots, w_n$ predstavljaju važnost svakog ulaza, a pristranost (*bias*) se može smatrati kao vrijednost za pomak aktivacijske funkcije da bi bolje uskladila predviđanje s podacima;
c. **funkcija zbroja** (engl. *net sum*): funkcija koja zbraja produkte svakog ulaza x_i sa pripadajućom mu težinom w_i :

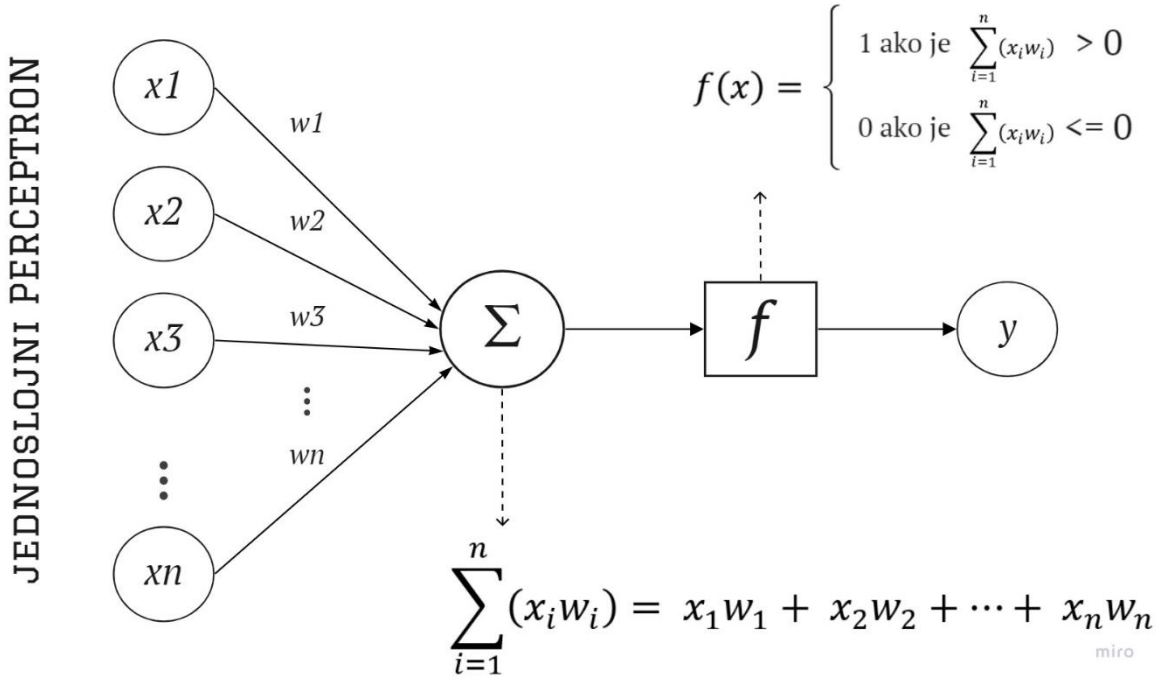
$$\sum_{i=1}^{n}(x_i w_i) = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \tag{7}$$

Težine se upotrebljavaju kao multiplikatori za ulazne vrijednosti neurona koje se zbrajaju, a zbroj težina pomnožen s ulaznim varijablama naziva se linearna kombinacija ulaznih varijabli [15];

d. **aktivacijska funkcija** (engl. *activation function*): funkcija koja uspoređuje rezultat funkcije zbroja sa definiranim pragom te određuje jedan od dva moguća izlaza, a naziva se i funkcija praga [25]:

$$f(x) = \begin{cases} 1 \; ako \; je \; \sum_{i=1}^{n}(x_i w_i) > 0 \\ 0 \; ako \; je \; \sum_{i=1}^{n}(x_i w_i) \leq 0 \end{cases} \tag{8}$$

Nakon što perceptron primi ulazne vrijednosti x_1, x_2, x_3, ... x_n zbraja umnoške svake vrijednosti sa pripadajućom težinom w_1, w_2, w_3, ..., w_n (7) i prosljeđuje izračun aktivacijskoj funkciji f (8) koja izračunava izlaznu vrijednost y uspoređujući izračun sa definiranim pragom (Slika 7).

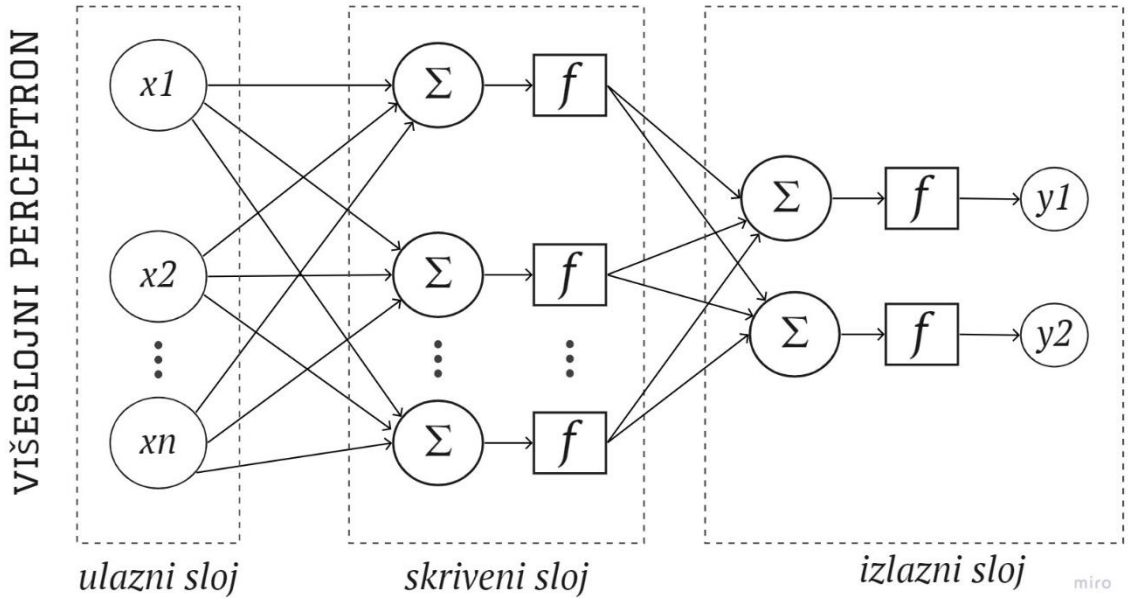


Slika 7 Jednoslojni perceptron

Jednostavno rečeno, perceptron je čvor ili mjesto gdje se događa računanje, a kombinira ulazni podatak sa skupom koeficijenata, ili težina, koji ili pojačavaju ili prigušuju taj unos, pridajući time značaj ulazima s obzirom na zadatak koji algoritam pokušava naučiti. [38] Ovi ulazi i težine se zbrajaju, a zatim taj zbroj prolazi kroz takozvanu aktivacijsku funkciju, kako bi se utvrdilo treba li i u kojoj mjeri taj signal dalje napredovati kroz mrežu kako bi utjecao na krajnji ishod. Ako signali prođu, neuron je "aktiviran". [38]

Višeslojni perceptron je neuronska mreža koja se sastoji od tri dijela: ulaznog sloja, jednog ili više skrivenih slojeva i izlaznog sloja (Slika 8). [12] Za ulazne podatke, neuronska mreža u izlaznom sloju daje klasifikacije ili izlazne signale na koje se ulazni podaci mogu mapirati. Skriveni slojevi fino podešavaju ulazne težine sve dok granica pogreške neuronske mreže ne bude minimalna. Učenje ili prilagodba odvija se kad se težine prilagođavaju kako bi mreža mogla generirati točne izlaze, baš kao kod metode linearne ili metode logističke regresije. [15] Pretpostavlja se da skriveni slojevi ekstrapoliraju istaknute značajke u ulaznim podacima koji imaju moć predviđanja u pogledu izlaza. Ovo opisuje ekstrakciju

značajki, koja postiže korisnost sličnu statističkim tehnikama kao što je analiza glavnih komponenti. [22]



Slika 8 Višeslojni perceptron

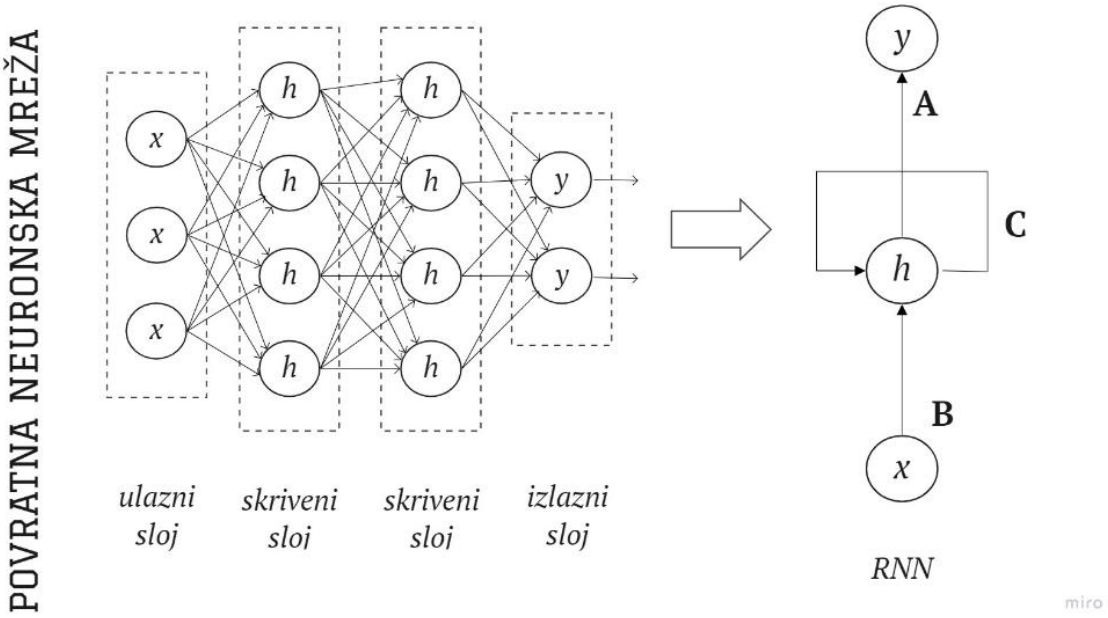
Duboko učenje je naziv koji koristimo za složene neuronske mreže; odnosno umjetne neuronske mreže sa tri ili više slojeva. [38] Mreže dubokog učenja razlikuju se od uobičajenih jednoslojnih neuronskih mreža svojom dubinom. Dubina se određuje brojem slojeva kroz koje podaci moraju proći u više stupanjskom procesu prepoznavanja uzoraka. [38] U mrežama dubokog učenja, svaki sloj čvorova trenira se na posebnom skupu značajki na temelju izlaza prethodnog sloja. Sposobnost dubokog učenja da obrađuje i uči iz ogromnih količina neoznačenih podataka je prednost u odnosu na prethodne algoritme.

Većina neuronskih mreža ne pamti korake iz prethodnih situacija i stoga ne uči donositi odluke na temelju konteksta. No, rješenje ovog problema je povratna neuronska mreža koja pohranjuje informacije iz prošlih stanja i sve svoje odluke donosi na temelju onoga što je naučila iz prošlosti.

2.1. Reccurent Neural Network (RNN)

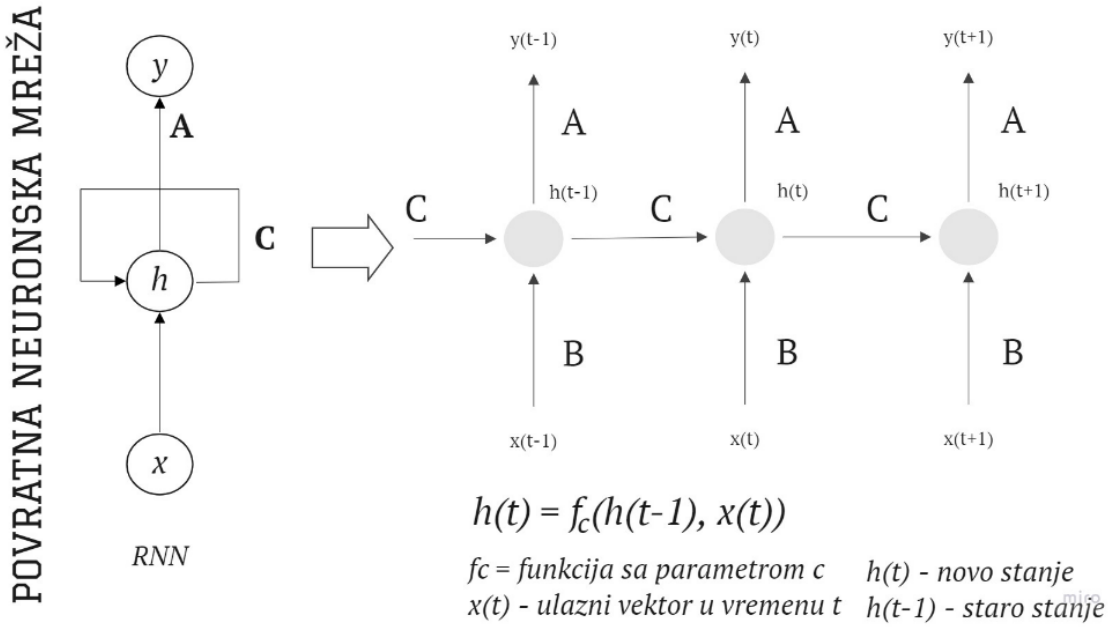
Klasična neuronska mreža (Feed-Forward Neural Network) radi na način da ulazni sloj prima ulazne podatke, obrađuje ih te prosljeđuje na srednji sloj. Srednji se sloj može sastojati od više skrivenih slojeva, svaki sa svojim aktivacijskim funkcijama i težinama. Povratna neuronska mreža (engl. *reccurent neural network*, skraćeno RNN) će standardizirati različite

aktivacijske funkcije, težine i pristranosti tako da svaki skriveni sloj ima iste parametre. [43] Tada će, umjesto stvaranja više skrivenih slojeva, stvoriti jedan i prelaziti preko njega onoliko puta koliko je potrebno (Slika 9). [43]



Slika 9 Feed-Forward Neural Network → Recurrent Neural Network

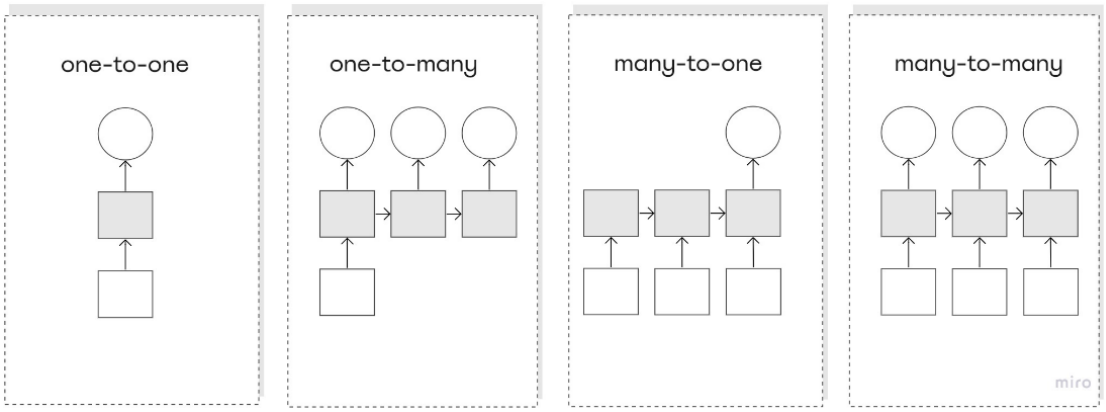
Slika 10 prikazuje primjer RNN modela, gdje je *x* ulazni sloj, *h* je skriveni sloj, a *y* je izlazni sloj. *A*, *B* i C su parametri mreže koji se koriste za poboljšanje izlaza modela. U bilo kojem trenutku *t*, trenutni ulaz je kombinacija ulaza na $x_{(t)}$ i $x_{(t-1)}$. Izlaz se u bilo kojem trenutku dohvaća natrag u mrežu radi poboljšanja izlaza. [43]



Slika 10 RNN model

Povratna neuronska mreža, jednostavno rečeno, radi na principu spremanja izlaza određenog sloja i vraćanja ga natrag na ulaz kako bi se predvidio izlaz. Kada se donosi odluka u RNN modelu, u obzir se uzima trenutni ulaz te ono što je naučeno iz prethodnog ulaza, a što je spremljeno u RNN memoriju. [26] Unutarnja memorija RNN modelima omogućuje veliku preciznost u predviđanju onoga što slijedi te su zbog toga često korišteni za predviđanje sekvencijalnih podataka kao što su govor, tekst, financijski podaci i slično.

Četiri su vrste RNN mreža (Slika 11): one-to-one označava jednostavne RNN mreže koje za svaki ulazni podatak daju jedan izlazni podatak, one-to-many mreže za svaki ulazni podatak daju niz podataka (primjerice generiranje opisa slike gdje za ulaznu sliku na izlazu daje niz riječi), many-to-one mreže za niz podataka na ulazu daju jedan izlaz (primjerice primjena u analizi sentimenta gdje za niz riječi (rečenicu) daje pozitivnu ili negativnu klasifikaciju) i many-to-many mreže na ulazu dobiju niz podataka, a na izlazu također daju niz podataka (primjer je strojno prevođenje gdje za niz riječi jednog jezika generira niz riječi drugog jezika). [43]



Slika 11 Vrste RNN mreža

RNN mreže predstavljaju poboljšanje klasičnih statističkih metoda pri obradi jezika. [26] Povratne su neuronske mreže korištene na tri razine u svojoj primjeni u obradi prirodnog jezika: klasifikacija riječi, klasifikacija rečenica i generiranje jezika [26].

Marques i Lopes (2008) navode problem potrebe za velikim količinama ručno označenih tekstova potrebnih za treniranje RNN modela te predlažu model neuronske mreže koja je sposobna prevladati navedeni problem. [34] U svom istraživanju uspoređuju tri različite arhitekture neuronske mreže: prvi model čine samo ulazni i izlazni sloj, u drugom modelu uz ulazni i izlazni sloj dodan je i jedan skriveni sloj, a treći model čini Elmanov RNN model. Ove različite modele neuronske mreže primjenjuju na problem učenja parametara vrsta riječi

iz vrlo malog portugalskog korpusa za obuku i iz podskupa Susanne Corpusa. Prema dobivenim rezultatima korpus za treniranje koji se sastoji od 3362 ručno označene riječi omogućuje preciznost predviđanja iznad 95%, a najbolji su rezultati postignuti jednoslojnom neuronskom mrežom. Dobiveni rezultati ukazuju na stopu ispravka točnosti iznad 97% za korpus treniranja s približno 15 000 riječi. [34]

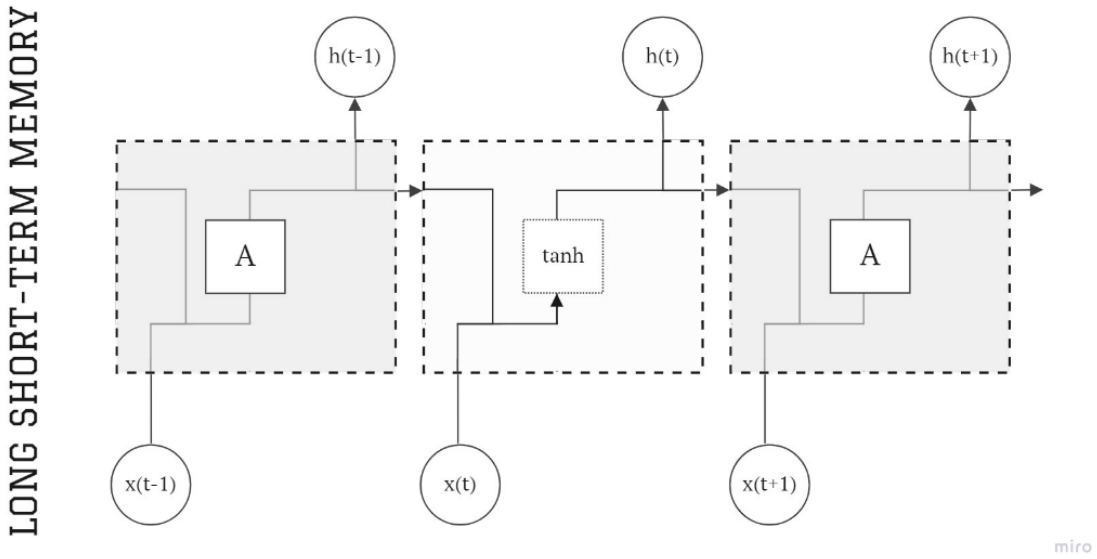
Dinarelli i Tellier (2016) proučavaju različite arhitekture povratnih neuronskih mreža za označavanje nizova. Navode da se u literaturi o povratnim neuronskim mrežama u obradi prirodnog jezika koriste uglavnom dvije arhitekture: Elmanov RNN i Jordanov RNN. [14] Razlika između ova dva modela leži u povratnoj vezi unutar RNN modela. U Elmanovom modelu veza je petlja u skrivenom sloju koja omogućuje RNN mrežama korištenje, u trenutnom vremenskom koraku, stanja skrivenog sloja prethodnih vremenskih koraka. [14] Jordanov RNN ima povratnu vezu između izlaznih i skrivenih slojeva. U ovom slučaju kontekstualne informacije se sastoje od jedne ili više oznaka predviđenih u prethodnim vremenskim koracima. S druge strane, Dinarelli i Tellier predlažu dva nova RNN modela: u prvom modelu povratna veza je između izlaznog i ulaznog sloja (daju se oznake predviđene na prethodnim pozicijama u niz kao ulaze u mrežu), a drugi je kombinacija Elmanovog RNN-a i prvog modela (mogu se iskoristiti pružene kontekstualne informacije iz prethodnih stanja skrivenog sloja i oznake predviđene na prethodnim pozicijama niza). Prema rezultatima usporedbe njihovih predloženih modela sa Elmanovim i Jordanovim, njihov model uvijek nadmašuje Elmanov i Jordanov RNN model. [14]

Problem povratnih neuronskih mreža je kratkoročno pamćenje: ako je niz ulaznih podataka dovoljno dugačak, teško će se prenijeti informacije iz ranijih vremenskih koraka u kasnije. [41] Povratne neuronske mreže pate od problema nestajanja gradijenta. Gradijenti su vrijednosti koje se koriste za ažuriranje težina neuronske mreže. [41] Jednostavno rečeno, gradijent mjeri koliko se izlaz promijeni ako se napravi mala promjena na ulazu, a problem nastaje kada se gradijent toliko smanjuje dok se vraća na prethodne korake da mu vrijednost postane iznimno mala što ne pridonosi previše učenju. [41] Dakle, slojevi koji dobiju malo ažuriranje gradijenta prestaju učiti pa RNN-ovi mogu zaboraviti što su vidjeli u dužim nizovima podataka, stoga kažemo da imaju kratkoročno pamćenje. Poboljšanje u odnosu na RNN je LSTM koji rješava problematično pitanje nestajanja gradijenata jer održava gradijente dovoljno strmima, što čini treniranje modela relativno kratkim, a preciznost visokom.

2.2. Long Short Term Memory (LSTM)

LSTM (engl. *Long Short Term Memory*, skraćeno LSTM) je posebna vrsta povratne neuronske mreže koja ima mogućnost pamćenja podataka tijekom dužeg vremenskog razdoblja. Memorija LSTM mreže slična je memoriji računala, a iz nje se informacije mogu čitati, pisati i brisati. Ova se memorija može promatrati kao zatvorena jedinica koja odlučuje hoće li ili ne pohraniti ili izbrisati informacije na temelju važnosti koju pridaje informaciji.

LSTM sloj sastoji se od skupa povezanih ćelija poznatih kao memorijski blokovi (Slika 12).



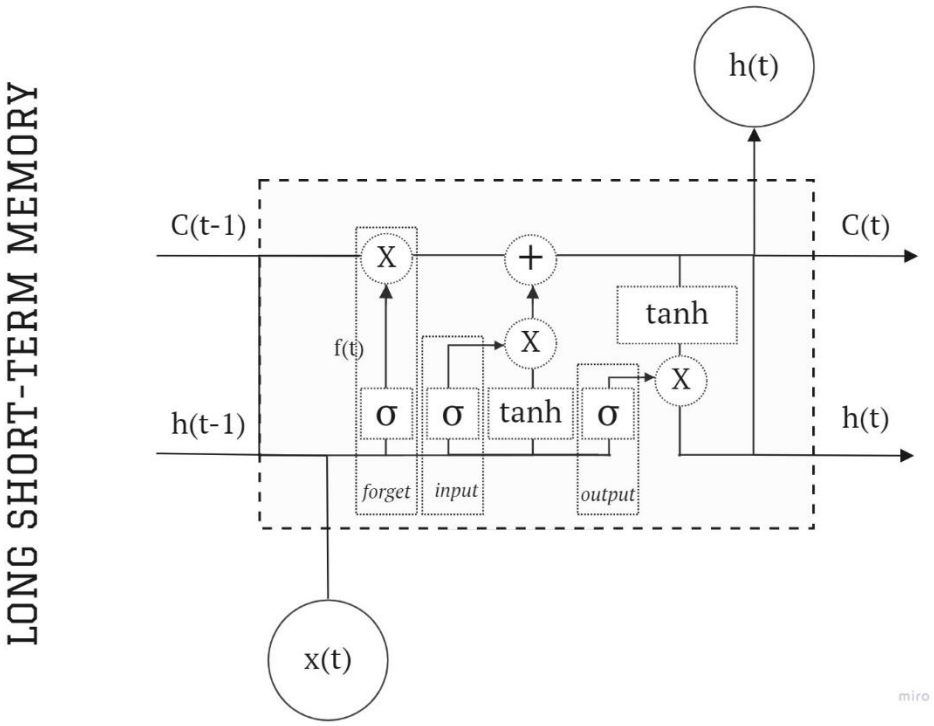
Slika 12 LSTM mreže

LSTM sadrži troja vrata: *input* (određuju hoće li se dopustiti novi unos), *forget* (određuju hoće li se izbrisati informacije) i *output* (izlazna) vrata. *Forget* vrata odlučuju koje informacije treba odbaciti ili zadržati na način da informacije iz prethodnog skrivenog stanja i informacije iz trenutnog ulaza prolaze kroz sigmoidnu funkciju. [41] Sigmoidna[2] funkcija vraća vrijednosti koje se pojavljuju između 0 i 1: što je vrijednost bliže nuli znači da je treba zaboraviti, a što je bliže jedinici znači da je važna i da je treba zadržati. [41] Za ažuriranje stanja ćelije postoje *input* vrata. Ona prvo prosljeđuju prethodno skriveno stanje i trenutni ulaz u sigmoidnu funkciju. Nakon toga prosljeđuje skriveno stanje i trenutni ulaz u tanh[3]

[2] Sigmoidna funkcija pretvara poslane joj ulazne vrijednosti u vrijednosti raspona [0, 1]: $s(x) = \frac{1}{1+e^{-x}}$

[3] Tanh (engl.*tangent hyperbolic*) funkcija pretvara ulazne vrijednosti u izlaze raspona [-1, 1]: $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

funkciju da bi se vrijednosti podesile između -1 i 1 kako bi se lakše regulirala mreža. [41] Zatim se množi izlaz tanh funkcije sa sigmoidnom funkcijom da bi se odlučilo koje izlaze tanh funkcije je potrebno zadržati. *Output* vrata odlučuju koje bi trebalo biti sljedeće skriveno stanje. Prvo se prosljeđuju prethodno skriveno stanje i trenutni ulaz u sigmoidnu funkciju, a zatim se prosljeđuje novo modificirano stanje ćelije funkciji tanh. [41] Potom se množi tanh izlaz sa sigmoidnim izlazom, a rezultat je novo skriveno stanje. Novo stanje ćelije (C_t) i novo skriveno stanje ćelije (h_t) zatim se prenose na sljedeći vremenski korak (Slika 13). [41]



Slika 13 LSTM model

Rad LSTM mreže može se sažeti u tri koraka [43] (Slika 13) :

a. odlučivanje koliko prošlih podataka treba zapamtiti: prvi korak je odlučiti koje informacije treba izostaviti iz ćelije u određenom vremenskom koraku, a to određuje sigmoidna funkcija. Gleda prethodno stanje h_{t-1} zajedno s trenutnim ulazom x_t i izračunava funkciju:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (9)$$

b. odlučivanje koliko trenutna ćelija pridodaje trenutnom stanju: ovaj se korak sastoji od dva dijela sigmoidne funkcije (10) koja odlučuje koje vrijednosti propustiti (0 ili 1) i tanh funkcije (11) koja dodaje težinu vrijednostima koje se prosljeđuju te tako odlučuje o njihovoj važnosti:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \quad (10)$$

$$\tilde{C}_t = tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \quad (11)$$

c. odlučivanje koji dio trenutnog stanja ćelije ide na izlaz: u trećem koraku prvo se pokreće sigmoidni sloj (12), koji odlučuje koji će dijelovi stanja ćelije doći do izlaza, a zatim stanje ćelije prolazi kroz tanh funkciju kako bi se dobivene vrijednosti "gurnule" između -1 i 1 i to se množi s izlazom sigmoidnog sloja (13):

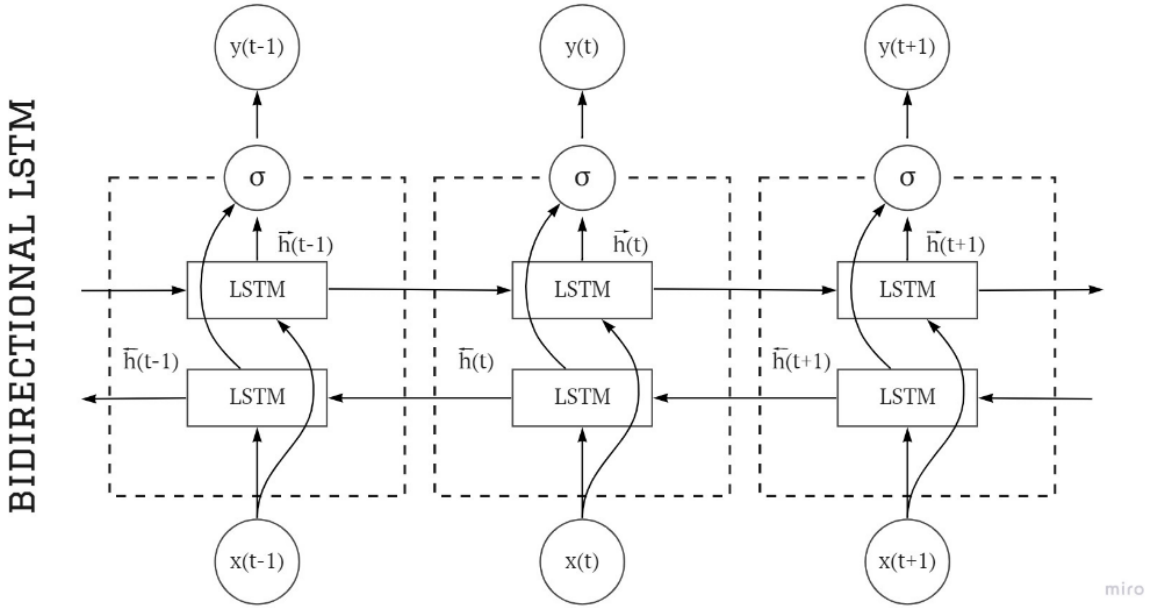
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \quad (12)$$

$$h_t = o_t \cdot tanh(C_t) \quad (13)$$

LSTM model bilježi samo prošla stanja, no u nekim slučajevima, poput označavanja nizova podataka, korisno je znati i prethodna i buduća stanja u određenom trenutku. Rješenje ovog problema je posebna arhitektura LSTM modela koja se sastoji od dva LSTM sloja, a koju nazivamo dvosmjerni LSTM model (engl. *Bidirectional Long Short Term Memory*, skraćeno BiLSTM). [33] BiLSTM s obzirom na rečenicu $w_1, w_2, \ldots, w_3$ sa oznakama $y_1, y_2, \ldots, y_3$ se koristi za predviđanje distribucije vjerojatnosti oznake svake riječi. [33] Dobivena distribucija vjerojatnosti svake oznake ne ovisi jedna o drugoj jer kontekstualne informacije dolaze samo iz BiLSTM sloja. Stoga, najvjerojatnija oznaka y_i ulazne riječi w_i može se odabrati pomoću funkcije:

$$y_i = arg\ max_{\ t\in 1,\ldots,m} P_i\ (\mathrm{t}|\mathrm{w}_1, w_2, \ldots, w_n) \qquad \text{za } i \in 1, \ldots, n \quad (14)$$

gdje je *m* ukupan broj oznaka vrsta riječi, a *n* ukupan broj distribucija vjerojatnosti. [33]



Slika 14 BiLSTM model

Ukratko, BiLSTM dodaje još jedan LSTM sloj, koji mijenja smjer protoka informacija (Slika 14). Prethodne značajke izdvaja prvi LSTM sloj, a buduće značajke hvata drugi LSTM sloj. To znači da niz ulaznih podataka teče unatrag u dodatnom LSTM sloju. [47] Naposlijetku kombiniraju se izlaze iz oba LSTM sloja na nekoliko načina, npr. prosjek, zbroj, množenje ili ulančavanje. [47] U odnosu na LSTM model, BiLSTM je znatno sporiji te je potrebno više vremena za njegovo treniranje. Stoga se preporuča njegovo korištenje samo u potrebnim situacijama. [47]

Lorincz et al. (2019) istražuju označavanje vrsta riječi rumunjskog jezika te predlažu LSTM model u kombinaciji s potpuno povezanim izlaznim slojem za predviđanje POS i MSD (poglavlje 3) oznaka. Prema rezultatima njihov LSTM model postiže najveću točnost od 99,18%. Izvješćuju da dodavanje sloja za *letter embeddings* i korištenjem slojeva za *word embeddings* umjesto *one-hot* kodiranja za predstavljanje ulaznih podataka nije rezultiralo većom preciznošću modela. Autori zaključuju da je njihova predložena metoda neovisna o jeziku jer se u ulaznim značajkama modela ne primjenjuje nikakvo lingvističko znanje. [31]

Kod istraživanja koja se tiču dvosmjernih LSTM modela (BiLSTM) često se BiLSTM model koristi u kombinaciji s nekom drugom metodom za POS označavanje [7] [21] ili drugim modelom neuronske mreže [9] [26] [21].

Hoesen i Purwarianti (2018) primjenjuju duboko učenje za razvoj indonezijskog označivača imenovanih entiteta (NER) gdje koriste LSTM model. U ovom se istraživanju prikazuje uporaba POS označavanja kao jednog od slojeva neuronske mreže za NER označavanje. Rezultati pokazuju da korištenje POS oznaka u *embedding* sloju kao dodatnog unosa poboljšava izvedbu NER označivača. [21]

AlKhwiter i Al-Twairesh (2021) uspoređuju rezultate POS označavanja koristeći BiLSTM model s rezultatima primjenjujući CRF[4] metodu. Rezultati njihovog istraživanja za POS označavanje arapskog jezika pokazuju točnost BiLSTM modela od 96,5% čime prednjači nad CRF metodom. [7]

Balwant (2019) istražuje klasificiranje lažnih vijesti na društvenim mrežama. Predložena arhitektura uključuje BiLSTM model te konvolucijsku neuronsku mrežu (CNN). LSTM

[4] CRF (engl. *Conditional Random Field)* još je jedna od metoda koja se može koristiti za POS označavanje. CRF je algoritam koji se koristi za identifikaciju entiteta ili uzoraka u tekstu (kao što su POS oznake). Više na: https://towardsdatascience.com/pos-tagging-using-crfs-ea430c5fb78b

model se koristi za POS označavanje tekstova lažnih objava dok se CNN model koristi za dohvaćanje informacija o korisniku koji ih objavljuje. Rezultati pokazuju da dobivena hibridna arhitektura značajno poboljšava učinkovitost otkrivanja lažnih vijesti. [9]

Kumar (2020) također kombinira BiLSTM i CNN model u analizi sentimenta. Prema rezultatima predloženim modelom se povećava točnost (98,6%) i smanjuje vrijeme izvršavanja u usporedbi s tradicionalnim metodama. [26]

3 Označavanje vrsta riječi hrvatskog jezika

Hrvatski jezik se smatra morfološki veoma bogatim jezikom što za računalnu obradu teksta hrvatskog jezika predstavlja veći izazov nego za obradu teksta na engleskom jeziku. Prema hrvatskoj školskoj gramatici morfologija se definira kao grana gramatike koja proučava oblike i vrste riječi. Morfološka analiza uključuje proučavanje oblika riječi kako bi se utvrdilo o kojoj se vrsti riječi radi. Svaka se riječ sastoji od morfema, najmanjih jezičnih jedinica koje imaju svoje značenje. [48]

Ovdje je važno navesti razliku između POS i MSD označavanja. Kao što je prethodno navedeno, POS označavanjem se riječima pridjeljuju oznake vrste riječi. Univerzalni skup POS oznaka (engl. *universal part-of-speech tags*, skraćeno UPOS[5]) koje su zajedničke mnogim jezicima čine sljedeće oznake:

ADJ	*pridjev*	INTJ	*usklik*	PUNCT	*interpunkcija*
ADP	*prijedlog*	NOUN	*imenica*	SCONJ	*zavisni veznik*
ADV	*prilog*	NUM	*broj*	SYM	*simbol*
AUX	*pomoćni glagol*	PART	*čestica*	VERB	*glagol*
CCONJ	*nezavisni veznik*	PRON	*zamjenica*	X	*ostalo*
DET	*determinant*	PROPN	*vlastita imenica*		

MSD (engl. *morphosyntactic descriptors*) je kratica za morfosintaksne oznake koje se koriste kod visoko flektivnih[6] jezika kao što su južnoslavenski jezici (slovenski, hrvatski, srpski i sl.). Ovakve oznake sadrže dodatne značajke flektivnih jezika kao što su gramatičke kategorije roda, broja i padeža ili glagolska vremena. Ako se ponovno za primjer uzme značenje riječi *vodi* u rečenicama: „*On vodi tim do pobjede.*“ i „*On ide prema onoj vodi*, pripadajuće MSD oznake za riječ *vodi* su:

[5] https://universaldependencies.org/u/pos/

[6] Flektivni jezici (prema lat. *flectere*: svijati; sklanjati po padežima) su tip jezika u kojem se gramatičke kategorije izražavaju pretežito fleksijom, promjenom oblika riječi u morfološkim paradigmama.

a. u prvoj rečenici riječ *vodi* ima oznaku Vmr3s (*Verb Type=main VForm=present Person=third Number=singular*) –prezent glagola *voditi* u trećem licu jednine;
b. u drugoj rečenici riječ *vodi* je ima oznaku Ncfsl (*Noun Type=common Gender=feminine Number=singular Case=locative*) – opća imenica *voda* u lokativu ženskog roda jednine.

U POS označavanju hrvatskog jezika najčešće se koriste MSD oznake koje su u skladu sa MULTEXT-East skupom podataka. MULTEXT-East kao skup podataka za istraživanje i razvoj jezičnog inženjerstva je dostupan i s univerzalnim i s morfosintaksnim oznakama hrvatskog jezika. Skupovi oznaka za morfološki bogate jezike, kao što je hrvatski jezik, su nizovi morfoloških oznaka, a ne jedna oznaka. Korištenje niza kao što je *Ncnpg* (*Noun, common, neuter, plural, genitive*) kao POS oznake uvelike povećava ukupan broj POS oznaka pa korpusi mogu biti do deset puta veći od 50 do 100 oznaka koje se uobičajeno koriste za engleski jezik. [24] S takvim velikim korpusima, svaku riječ treba morfološki analizirati da bi se stvorio popis mogućih nizova morfoloških oznaka za riječ.

ReLDI je mreža istraživača stručnjaka za računalnu obradu teksta i lingvističku analizu prije svega za južnoslavenske jezike. Kroz suradnju sa CLARIN.SI infrastrukturom pružaju niz dostupnih NLP alata i resursa. Između ostalog, preko njihovog sučelja za pretragu korpusa lako je doći i do dostupnih korpusa za hrvatski jezik: CLASSLAWiki-hr (Croatian Wiikipedia), ENGRI (Croatian News Portals), hrWaC (Croatian Web), Tweet-hr (Croatian Tweets), ParlaMint-HR (Croatian parliament) ili hr500k. Mrežna aplikacija *ReLDIanno* (dostupno na poveznici: http://clarin.si/services/web/query), omogućuje MSD označavanje, lematizaciju, označavanje imenovanih entiteta (NER) i parsiranje za hrvatski jezik. Moguće je i korištenje odgovora mrežne aplikacije *ReLDIanno* u programskom kôdu, a najbolji način je učitavanje Python biblioteke *reldi*.

Za označavanje tekstova hrvatskog jezika može se koristiti i HunPos, označivač otvorenog kôda koji se temelji na skrivenom Markovljevom modelu. HunPos je posebno razvijen za jezike sa složenijom morfologijom te za preciznije označavanje nepoznatih riječi (riječi koje nisu sadržane u skupu za treniranje). No, instalacija i korištenje ovog alata na Linux i MacOSX sustavima je jednostavna (https://github.com/mivoq/hunpos), dok je za Windows sam proces dosta složeniji.

U ovom poglavlju riječ je o vrstama riječi u hrvatskom jeziku, dostupnim alatima i bibliotekama za POS i MSD označavanje tekstova na hrvatskom jeziku te dosadašnjim istraživanjima na tu temu.

3.1. Vrste riječi u hrvatskom jeziku

S obzirom na oblik riječi, riječi se nekog jezika dijele na vrste riječi. U hrvatskome jeziku ima deset vrsta riječi, a to su imenice, zamjenice, pridjevi, brojevi, glagoli, prilozi, prijedlozi, veznici, čestice i usklici. One se dijele na promjenjive (imenice, zamjenice, pridjevi, neki brojevi i glagoli) i nepromjenjive (prilozi, prijedlozi, veznici, čestice, usklici i neki brojevi).

Promjenjive vrste riječi u hrvatskom jeziku su riječi koje se mogu mijenjati u skladu s gramatičkom kategorijom kojoj pripadaju, tj. prema rodu, broju i padežu. Padež je gramatička kategorija riječi kojom se izriču odnosi imenskih riječi (imenice, zamjenice, pridjevi i brojevi) s ostalim riječima u rečenici.[7] Riječi se sklonidbom ili deklinacijom mijenjaju kroz sedam padeža koji odgovaraju na padežna pitanja: nominativ (*tko? što?*), genitiv (*koga? čega?*), dativ (*komu? čemu?*), akuzativ (*koga? što?*), vokativ (*oj! ej!*), lokativ (*o kome? o čemu?*) i instrumental (*s kim? s čim?*). Broj je gramatička kategorija riječi kojom se označuje govori li se o jednome primjerku onoga što označuje imenica ili o više njih.[1] Kada govorimo o broju, riječi mogu biti u jednini i u množini. Rod je gramatička kategorija riječi koja se očituje u slaganju imenice s pridjevom, a može biti muški, ženski i srednji.[1]

U hrvatskom jeziku primjenjive vrste riječi su [48]:

- **imenice**: riječi kojima se označuju bića, stvari i pojave, mijenjaju se po padežima i brojevima, a imaju i svoj rod (imenice mogu biti muškoga, ženskoga ili srednjega roda, a po broju mogu biti u jednini ili množini);
- **pridjevi**: riječi koje se dodaju imenicama kako bi ih preciznije označili, označuju svojstvo (kakvo je što?), pripadnost (čije je što?) i građu (od čega je što?), a dijele se na opisne pridjeve, posvojne i gradivne pridjeve. Pridjevi svoj oblik mijenjaju prema padežu, rodu, broju i vidu (određeni i neodređeni);
- **zamjenice**: riječi kojima se zamjenjuju druge riječi (najčešće imenice), a sklanjaju se po padežima;
- **brojevi**: riječi kojima označujemo koliko čega ima (glavni brojevi) ili koje je što po redu (redni brojevi).

[7] http://pravopis.hr

- **glagoli**: riječi kojima se izriče radnja, stanje i zbivanje, a mijenjaju se po licima (taj se proces naziva sprezanje ili konjugacija). Ostale gramatičke kategorije po kojima se mijenjaju su: rod, broj, vrijeme, vid, način, prijelaznost.

Nepromjenjive riječi u hrvatskom jeziku su one koje ne mijenjaju svoj oblik te na njihov oblik ne utječu ni ostale riječi u rečenici ni mjesto na kojemu se nalaze. Nepromjenjive vrste riječi u hrvatskom jeziku su [48]:

- **prilozi**: riječi koje se najčešće dodaju glagolima kako bi izrazile različite okolnosti vršenja glagolske radnje;
- **prijedlozi**: riječi koje označavaju različite odnose između imenica (ili zamjenica koje upućuju na imenice);
- **veznici**: riječi koje povezuju dvije riječi, skupove riječi ili rečenice;
- **usklici**: riječi koje služe za izricanje nekih osjećaja ili raspoloženja, za dozivanje ili poticanje nekoga, odnosno za oponašanje zvukova iz prirode;
- **čestice**: riječi koje same nemaju (ili samo dijelom imaju) značenja, a koje služe za oblikovanje i preoblikovanje rečenica te na taj način mogu promijeniti značenje rečenice.

3.2. Pregled istraživačkih radova

Uz praktičnu implementaciju RNN i LSTM modela za označavanje riječi hrvatskog jezika ideja ovog rada bila je i istražiti dosadašnje radove s temom POS označavanja hrvatskog jezika. Označavanje vrsta riječi engleskog jezika u obradi je prirodnog jezika već dobro istražena tema te su izgrađeni različiti modeli s preciznostima i do 98%. [12] U slučaju flektivnih jezika (kao što je hrvatski) evaluirani modeli postižu manje točnosti i preciznosti. [1] Da bi se provjerila situacija s označivačima za hrvatski jezik pretražene su baze znanstvenih članaka *Web Of Science* i *Scopus*. Tablica 1 prikazuje relevantne pronađene radove.

Agić i Tadić (2006) opisuju rezultate primjene stohastičke metode za morfosintaksno označavanje hrvatskih tekstova. Korpus koji su koristili za treniranje modela bio je 100Kw Croatia Weekly, korpus novinskih tekstova koji je sadržavao oko 1000 različitih MULTEXT-East MSD oznaka. Alat koji su koristili bio je TnT označivač koji koristi implementaciju HMM modela za POS ili MSD označavanje. POS označivač je postigao najveću točnost od 98,63% te tako dostigao razinu ljudske pogreške. Prema rezultatima

njihov je označivač u prosjeku ispravno označio više od 98% testnih skupova, a većina MSD pogrešaka pojavila se na tipovima s najvećim brojem mogućih oznaka po riječi dok su pogreške za POS označavanje bile gotovo beznačajne. Autori navode da bi pokazana trigram metoda bila dobar izbor za označavanje hrvatskih tekstova u daljnjim istraživanjima. [2]

Agić, Dovedan i Tadić (2008) istražuju nekoliko metoda kombiniranja izlaza HMM modela za POS ili MSD označavanje sa fleksijskim leksikonom za hrvatski jezik kako bi poboljšali ukupnu točnost označavanja hrvatskih tekstova. Navode da njihov model (kojeg nazivaju CPT), kada je kombiniran s lematizatorom temeljenim na hrvatskom morfološkom leksikonu, na takav način da lematizator daje morfološke značajke označivaču kada susretne nepoznate riječi, nadmašuje TnT na zadanim MSD testnim slučajevima. No, iz testiranja su izostavili POS rezultate jer su i TnT i CPT u mogućnosti postići točnost preko 95% bez dodatnih modula pa su bili usredotočeni na poboljšanje točnosti MSD označavanja. Svoje rezultate uspoređuju s prethodnim istraživanjem (Agić i Tadić, 2006) te navode da iako ovaj CPT pokazuje bolje rezultate od TnT označivača u označavanju nepoznatih riječi ipak se on bavio s manje nepoznanica u testnom slučaju. [4]

Agić, Dovedan i Tadić (2008) u radu *Improving Part-of-Speech Tagging Accuracy for Croatian by Morphological Analysis* predstavljaju CroTag - stohastički POS/MSD označivač kojim su željeli poboljšati označavanje tekstova na hrvatskom jeziku. U svom radu predlažu njegovu kombinaciju s velikim leksikonom hrvatskog jezika sa ciljem stvaranja hibridnog sustava za precizno označavanja hrvatskog korpusa. [5] Njihovi rezultati pokazuju da CroTag funkcionira na visokom nivou glede označavanja pomoću skrivenog Markovljevog modela.

Agić, Dovedan i Tadić (2010) predstavljaju rezultate njihovog eksperimentiranja s kombiniranjem izlaza POS označivača putem *tagger voting* pristupa kako bi poboljšali točnost MSD označivača za hrvatski jezik. Njihov model koristi pet označivača, a njihovi pojedinačni izlazi se procjenjuju, kombiniraju i zatim razjašnjavaju pomoću jednostavnih pravila odlučivanja (*tagger voting*). Tri od pet korištenih označivača temeljeno je na HMM modelu (CroTag, HunPoS i TnT), četvrti se temelji na SVM (engl. *support vector machines*) metodi (SVMTool), a peti koristi stablo odluke (TreeTager). Sva tri označivača temeljena na HMM modelu su postigla najbolje rezultate. Rezultati su pokazali da je njihova *tagger voting* ideja kombinacije više označivača nadmašila pojedinačne označivače u većini testnih slučajeva. Međutim, važnim ističu da se pokazalo kako razlike između najboljih

pojedinačnih označivača i kombiniranog označivača nisu statistički značajne te da to treba uzeti u obzir u budućim istraživanjima. [6]

Peradin i Šnajder (2012) implementiraju gramatiku ograničavanja (CG) koja koristi ručno izrađena pravila ovisna o kontekstu za razrješenje gramatičkog označavanja riječi. Autori opisuju razvoj morfološkog označivača za hrvatski jezik koji koristi morfološki analizator temeljen na flektivnom leksikonu te MULTEXT-East korpusu. Njihova gramatika se sastoji od 290 pravila organiziranih u pravila čišćenja i mapiranja, pravila razjašnjenja i heuristička pravila. Njihova metoda rezultira preciznošću od 96,1% za POS označavanje i 88,2% za MSD označavanje. [40]

Hladek, Staš i Juhar (2012) predlažu klasifikator temeljen na skrivenom Markovljevom modelu, a koji se može koristiti za rješavanje problema POS označavanja slavenskih jezika, poput slovačkog, češkog ili poljskog. Ovi su jezici vrlo flektivni i morfološki bogati te imaju vrlo velik vokabular (što je slično sa slučajem hrvatskog jezika te je zato ovaj rad relevantan). Eksperimenti su pokazali da kombinacija nekoliko izvora informacija, kao što su vjerojatnosti opažanja temeljene na sufiksu, vjerojatnosti prijelaza pojednostavljenih oznaka i morfološki rječnik za morfološku analizu, donose važna poboljšanja točnosti. [23]

Agić, Ljubešić i Merkler (2013) istražuju tada najsuvremeniji pristup morfosintaksnom označavanju – statističke modele, koje primjenjuju u lematizaciji i MSD označavanju hrvatskih i srpskih tekstova. Modeli koje uspoređuju su HunPoS, CST, PurePoS, SVMTool i TreeTagger, a alati s najboljom izvedbom su se pokazali CST lematizator i HunPos označivač. U procjeni ovih modela postignuta je točnost lematizacije od 97,87% i 96,30% za hrvatski i srpski te točnost MSD označavanja od 87,72% i 85,56%, uz točnost POS označavanja od 97,13% i 96,46%. Autori navode kako planiraju predstavljene modele iskoristiti za označavanje korpusa za hrvatski i srpski jezik – hrWaC i srWaC. [3]

Meftah et al. (2018) u svom radu opisuju morfosintaksno označavanje Twitter objava koristeći *CEA List DeepLIMA,* alat koji je višejezična platforma za analizu teksta temeljena na dubokom učenju. Njihov je pristup usredotočen na morfosintaksno označavanje Twitter objava na trima južnoslavenskim jezicima: slovenskom, hrvatskom i srpskom. Autori predlažu korištenje modela neuronske mreže te model prijenosnog učenja s obzirom na nedostatak označenih skupova podataka za navedeni problem. Predložena arhitektura neuronske mreže još je jedan oblik dvosmjerne neuronske mreže - BiGRU (engl. *bidirectional Gated Recurrent Unit*). Prema rezultatima POS označavanje bez primjene

prijenosnog učenja daje (za slovenski jezik) točnost od 89,55%, a s primjenom prijenosnog učenja 91,71%. [35]

Ljubešić i Dobrovoljc (2019) uspoređuju poboljšanja prelaska s tradicionalnih pristupa POS označavanju na neuronske i to za južnoslavenske jezike (slovenski, hrvatski i srpski). [32] U svom su istraživanju za tradicionalni pristup koristili *reldi-tagger* (označivač koji koristi CRF algoritam), a za pristup s neuronskim mrežama *stanfordnlp* (najsuvremeniji alat u neuronskoj morfosintaksnoj analizi). Rezultati su pokazali značajne razlike između *reldi-tagger* i *stanfordnlp* alata. Rezultati pokazuju da su neke od najčešćih pogrešaka za *reldi-tagger* znatno smanjene *stanfordnlp* alatom, poput zbunjenosti između imenica muškog roda u akuzativu jednine i nominativu. [32]

Robnik-Sikonja i Ulčar (2020) eksperimentirali su s višejezičnim BERT modelom za NER i POS označavanje hrvatskog, slovenskog i engleskog jezika. BERT (engl. *Bidirectional Encoder Representations from Transformers*) je okvir strojnog učenja za obradu prirodnog jezika, a dizajniran je da pomogne računalima razumjeti značenje dvosmislenosti u tekstu korištenjem okolnog teksta za određivanje konteksta. [46] Autori su implementirali CroSloEngual BERT model koji je treniran za hrvatski, slovenski i engleski jezik. Rezultati su pokazali da CroSloEngual i FinEst BERT imaju znatno bolje rezultate od višejezičnog BERT modela (mBERT) za NER zadatke. Rezultati POS označavanja pokazuju značajna poboljšanja predloženih modela koji nisu ni u jednom testnom slučaju lošija od mBERT-a. [46]

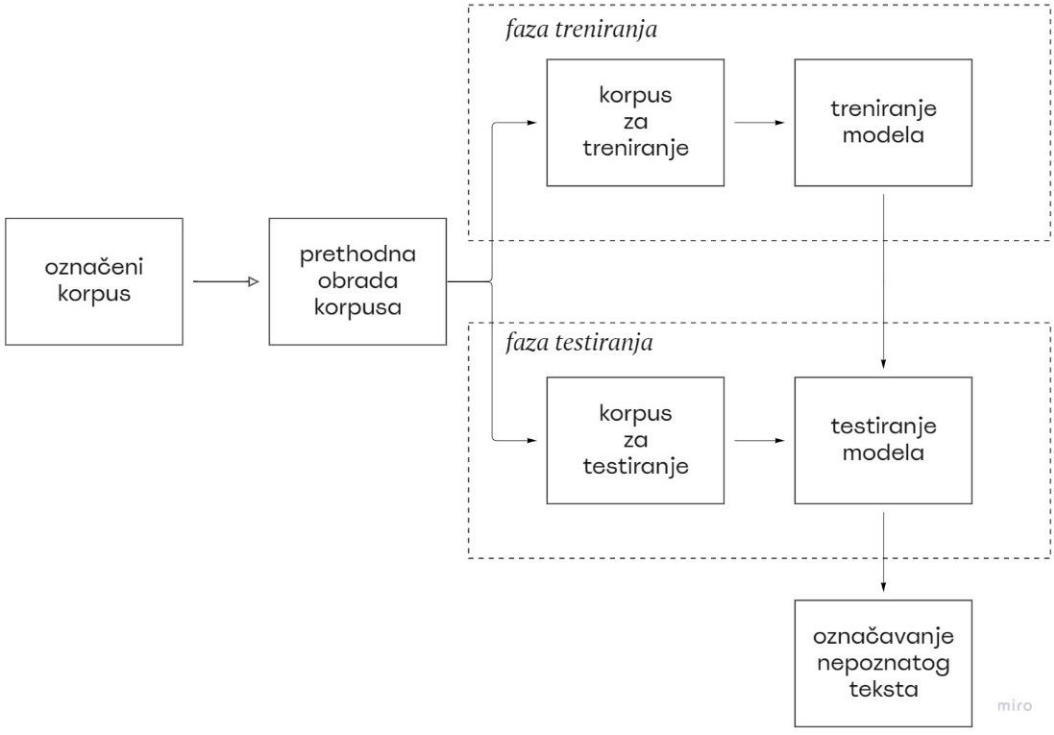
Tablica 1 Istraživanja o POS i MSD označavanju hrvatskog jezika

Godina	**Autori**	**Tema**
2006.	Agić, Željko; Tadić, Marko	Evaluating Morphosyntactic Tagging of Croatian Texts
2008.	Agić, Željko; Dovedan, Zdravko; Tadić, Marko	Combining Part-of-Speech Tagger and Inflectional Lexicon for Croatian
2008.	Agić, Željko; Dovedan, Zdravko; Tadić, Marko	Improving Part-of-Speech Tagging Accuracy for Croatian by Morphological Analysis
2009.	Agić, Željko; Dovedan, Zdravko; Tadić, Marko	Evaluating Full Lemmatization of Croatian Texts
2010.	Agić, Željko; Dovedan, Zdravko; Tadić, Marko	Tagger Voting Improves Morphosyntactic Tagging Accuracy on Croatian Texts
2012.	Peradin, Hrvoje; Šnajder, Jan	Towards a Constraint Grammar Based Morphological Tagger for Croatian
2012.	Dabelo Bašić, Bojana; Karan, Mladen; Šnajder, Jan	Evaluation of Classification Algorithms and Features for Collocation Extraction in Croatian
2012.	Justin, Tadej; Pobar, Miran; Ipšić, Ivo; Mihelič, France; Žibert, Janez	A Bilingual HMM-Based Speech Synthesis System for Closely Related Languages
2012.	Hladek, Daniel; Staš, Jan; Juhar, Jozef	Dagger: The Slovak Morphological Classifier
2013.	Agić, Željko; Ljubešić, Nikola; Merkler, Danijela	Lemmatization and Morphosyntactic Tagging of Croatian and Serbian
2016.	Orešković, Marko; Čubrilo, Mirko; Essert, Mario	The Development of a Network Thesaurus with Morpho-semantic Word Markups
2018.	Meftah, Sara; Semmar, Nasredine; Sadat, Fatiha; Raaijmakers, Stephan	Using Neural Transfer Learning for Morpho-syntactic Tagging of South-Slavic Languages Tweets
2019.	Dobrovoljc, Kaja; Ljubešić Nikola	What does Neural Bring? Analysing Improvements in Morphosyntactic Annotation and Lemmatisation of Slovenian, Croatian and Serbian
2020.	Stanković, Ranka; Sandrih, Branislava; Krstev, Cvetana; Utvić, Miloš; Škorić, Mihailo	Machine Learning and Deep Neural Network-Based Lemmatization and Morphosyntactic Tagging for Serbian
2020.	Ulčar, Matej; Robnik-Šikonja, Marko	FinEst BERT and CroSloEngual BERT: less is more in multilingual models

4 Primjena RNN i LSTM modela za označavanje vrsta riječi

Za izradu praktičnog dijela ovog rada potrebno je bilo provesti sljedeće korake:

- pronaći označeni korpus hrvatskog jezika,
- analizirati odabrani korpus,
- nad tim korpusom provesti predprocesiranje:
    - tokenizacija,
    - numerička reprezentacija,
    - vektorizacija,
    - word embeddings,
- osigurati skup za treniranje, skup za validaciju i skup za testiranje,
- implementirati RNN i LSTM arhitekture
    - treniranje
    - evaluacija.



Slika 15 Praktični dio rada - koraci

Za usporedbu efikasnosti različitih modela neuronske mreže u POS označavanju hrvatskog jezika korištene su dvije arhitekture koje su implementirane pomoću biblioteka za duboko učenje Keras[8] i Tensorflow[9]:

- RNN modeli:
    - jednostavni RNN model s proizvoljno postavljenim težinama koje se ne ažuriraju tijekom treniranja,
    - jednostavni RNN model s proizvoljno postavljenim težinama koje se ažuriraju tijekom treniranja,
    - jednostavni RNN model sa težinama koje su generirane Word2vec modelom koje se ažuriraju tijekom treniranja,
- LSTM modeli:
    - jednostavni LSTM model s proizvoljno postavljenim težinama koje se ažuriraju tijekom treniranja i
    - jednostavni LSTM model sa težinama generiranim Word2vec modelom koje se ažuriraju tijekom treniranja,
    - dvosmjerni LSTM model sa težinama generiranim Word2vec modelom koje se ažuriraju tijekom treniranja.

Izrada dubokih neuronskih mreža pomoću sekvencijalnog modela u Kerasu može se opisati kroz sljedeće korake:

- **definiranje modela:** instanciranje sekvencijalnog modela i dodavanje potrebnih slojeva,
- **kompajliranje modela:** pozivanje funkcije *.compile(),*
- **treniranje modela:** model "uči" nad poslanim podacima pozivanjem funkcije *.fit(),*
- **predviđanja pomoću modela:** korištenje modela za predviđanja nad novim podacima pozivanjem funkcije *.predict(),*
- **evaluacija modela**: računanje točnosti i gubitka pozivanjem metode *.evaluate(),*
- **testiranje modela**: usporedba rezultata predviđanja s testnim skupom podataka, a računa se točnost, opoziv i f1-score.

[8] https://keras.io/

[9] https://www.tensorflow.org/

4.1. Korpus

S obzirom na to da se tehnike obrade prirodnog jezika danas prvenstveno temelje na nadziranom strojnom učenju referentni korpusi za treniranje su ključni za razvoj NLP alata. Korpus je jednostavno rečeno veliki i strukturirani skup tekstova.

Hr500k referentni je korpus hrvatskih tekstova koji ukupno sadrži 900 dokumenata podijeljenih u 24 794 rečenice, ili 506 457 pojavnice. Cijeli korpus je označen na razinama leme i morfosintaksnih oznaka. Skup morfosintaksnih oznaka korištenih u korpusu u skladu je sa smjernicama MULTEXT-East skupa oznaka. Hr500k javno je dostupan na sljedećoj poveznici: https://huggingface.co/datasets/classla/hr500k/tree/main.

Za treniranje modela potrebno je preko poveznice preuzeti mapu *data_ner.zip* te iz nje učitati datoteke *train_ner.conllu* i *test_ner.conllu*. Preuzete datoteke mogu se učitati sa lokalnog računala pokretanjem programskog kôda koji se nalazi u privitku (Kôd 2). Korpus je sadržan u datoteci tipa *.conllu*, a izgled dostupnih informacija o jednoj rečenici iz korpusa prikazan je na slici (Slika 16).

```
# sent_id = train-s3
# text = Barem na papiru, izgleda kao odlična ideja.
1   Barem   barem   PART    Qo  _   _   _   _   O
2   na  na  ADP Sl  Case=Loc    _   _   _   O
3   papiru  papir   NOUN    Ncmsl   Case=Loc|Gender=Masc|Number=Sing    _   _   _   O
4   ,   ,   PUNCT   Z   _   _   _   _   O
5   izgleda izgledati   VERB    Vmr3s   Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   _   _   _   O
6   kao kao SCONJ   Cs  _   _   _   _   O
7   odlična odličan ADJ Agpfsny Case=Nom|Definite=Def|Degree=Pos|Gender=Fem|Number=Sing _   _   _   O
8   ideja   ideja   NOUN    Ncfsn   Case=Nom|Gender=Fem|Number=Sing _   _   _   O
9   .   .   PUNCT   Z   _   _   _   _   O
```

Slika 16 Struktura informacija o jednoj rečenici iz korpusa

U privitku (Kôd 3) je dostupna metoda (*read_data()*) kojom se podaci mogu iščitati iz *.conllu* datoteke. Navedenom se metodom podaci spremaju u rječnik, a za svaku rečenicu dostupne su sljedeće informacije:

- *sent_id*: jedinstveni identifikator rečenice,
- *text*: tekst rečenice,
- *tokens*: lista riječi,
- *xpos_tags*: lista MSD oznaka (MULTEXT-East oznake),
- *upos_tags:* lista UPOS oznaka (univerzalne POS oznake).

Statističkom analizom korpusa saznaje se da skup za treniranje ukupno sadrži 20 159 označenih rečenica koje ukupno sadrže 415 328 označenih riječi, a koje su označene sa 17 UPOS oznaka i 755 jedinstvenih MSD oznaka (Slika 17). S druge strane, skup za testiranje ima 2 672 rečenice sa ukupno 51 364 označene riječi. Riječi su u testnom skupu također označene sa 17 jedinstvenih UPOS oznaka te 564 MSD oznake.

Skup za treniranje sadrži 20159 rečenica.

	words	UPOS	MSD
count	415328	415328	415328
unique	64998	17	755
top	,	NOUN	Z
freq	21840	93576	50356

Skup za testiranje sadrži 2672 rečenice.

	words	UPOS	MSD
count	51364	51364	51364
unique	15108	17	564
top	,	NOUN	Z
freq	2874	10840	6762

Slika 17 Statistika označenog korpusa

----- treniranje ------

	tag	count	%
0	NOUN	93576	22.53
1	PUNCT	50187	12.08
2	ADJ	45916	11.06
3	VERB	40282	9.70
4	ADP	37127	8.94
5	AUX	25886	6.23
6	ADV	21580	5.20
7	PROPN	18931	4.56
8	CCONJ	18220	4.39
9	DET	17433	4.20
10	PRON	16013	3.86
11	SCONJ	11900	2.87
12	PART	7333	1.77
13	NUM	6256	1.51
14	X	4250	1.02
15	INTJ	256	0.06
16	SYM	182	0.04

----- testiranje ------

	tag	count	%
0	NOUN	10840	21.10
1	PUNCT	6742	13.13
2	ADJ	5669	11.04
3	VERB	4881	9.50
4	ADP	4540	8.84
5	AUX	3284	6.39
6	ADV	2633	5.13
7	PROPN	2415	4.70
8	CCONJ	2281	4.44
9	PRON	2044	3.98
10	DET	1968	3.83
11	SCONJ	1518	2.96
12	PART	970	1.89
13	NUM	852	1.66
14	X	644	1.25
15	INTJ	49	0.10
16	SYM	34	0.07

------ treniranje ------

	tag	count	%
0	Z	50356	12.12
1	Rgp	19035	4.58
2	Cc	18220	4.39
3	Var3s	14482	3.49
4	Sl	14117	3.40
5	Cs	11900	2.87
6	Vmr3s	10084	2.43
7	Sa	10033	2.42
8	Sg	9446	2.27
9	Ncmsn	8507	2.05
10	Ncfsg	8408	2.02
11	Ncmsg	7997	1.93
12	Vmn	7715	1.86
13	Npmsn	6982	1.68
14	Ncfsn	6831	1.64
15	Px--sa	6503	1.57
16	Ncfsa	6348	1.53
17	Ncmsan	5930	1.43

------ testiranje ------

	tag	count	%
0	Z	6762	13.16
1	Rgp	2366	4.61
2	Cc	2281	4.44
3	Var3s	1790	3.48
4	Sl	1749	3.41
5	Cs	1518	2.96
6	Sg	1233	2.40
7	Vmr3s	1231	2.40
8	Sa	1151	2.24
9	Ncmsn	1132	2.20
10	Ncfsg	942	1.83
11	Ncmsg	931	1.81
12	Ncfsn	870	1.69
13	Npmsn	840	1.64
14	Vmn	824	1.60
15	Ncmsan	750	1.46
16	Px--sa	749	1.46
17	Ncfsa	723	1.41

Slika 18 Zastupljenosti UPOS i MSD oznaka u skupu za treniranje i testiranje

Korpus ukupno sadrži 17 jedinstvenih UPOS oznaka: ADJ (pridjev), ADP (prijedlog), ADV (prilog), AUX (pomoćni glagol), CCONJ (nezavisni veznik), DET (determinator), INTJ (usklik), NOUN (imenica), NUM (broj), PART (čestica), PRON (zamjenica), PROPN (vlastita imenica), PUNCT (interpunkcija), SCONJ (zavisni veznik), SYM (simbol), VERB (glagol) i X (nepoznato). Usporedbom zastupljenosti pojedine UPOS oznake u skupu za treniranje i skupu za testiranje može se zaključiti da su gotovo podjednako zastupljene (u oba skupa najviše ima imenica, potom interpunkcijskih znakova, pa pridjeva itd.) (Slika 18).

Uvidom u postotak zastupljenosti najčešćih 17 MSD oznaka (Slika 18) vidljivo je da su podjednako zastupljene i u skupu za treniranje i u skupu za testiranje. Od ukupnog broja MSD oznaka najviše je oznaka Z (interpunkcija), potom Rgp (prilozi), Cc (zavisni veznici), Var3s (prezent pomoćnog glagola u trećem licu jednine), Sl (prijedlozi lokativa), Cs (nezavisni veznici) i tako dalje.

4.2. Vektorizacija

Neuronska mreža na ulazu očekuje brojčane podatke, odnosno numeričku reprezentaciju ulaznog teksta. Iz tog razloga prvi neophodan korak je numerička reprezentacija riječ i vektorizacija. Vektorizacija je postupak pretvaranja ulaznih podataka iz njihova originalnog formata u vektore realnih brojeva koji su potrebni za algoritme strojnog učenja. Vektorizacijom se iz teksta "izvlače" različite značajke na kojima će se model trenirati.

Za ovaj korak potrebna je funkcija *tf.keras.preprocessing.text.Tokenizer()*[10] iz Keras biblioteke koja omogućuje numeričku reprezentaciju korpusa tako da pretvara tekst korpusa u niz cijelih brojeva. Najprije se inicijalizira *Tokenizer()* objekt koji postavlja rječnike za korpus:

```
X_tokenizer = Tokenizer() # lista riječi X
Y_tokenizer = Tokenizer() # lista UPOS oznaka Y
Z_tokenizer = Tokenizer() # lista MSD oznaka Z
```

Potom se poziva metoda *fit_on_texts()* koja ažurira stvorene rječnike na temelju tekstova:

```
X_tokenizer.fit_on_texts(X)
Y_tokenizer.fit_on_texts(Y)
Z_tokenizer.fit_on_texts(Z)
```

Pozivom metode *texts_to_sequences()* tekst iz korpusa se pretvara u niz cijelih brojeva (svaki cijeli broj je indeks riječi u stvorenom rječniku):

```
X_encoded = X_tokenizer.texts_to_sequences(X)
Y_encoded = Y_tokenizer.texts_to_sequences(Y)
Z_encoded = Z_tokenizer.texts_to_sequences(Z)
```

[10] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

Prethodne je korake potrebno provesti i na skupu za testiranje, a rezultati prethodnih koraka mogu se vidjeti na primjeru jedne rečenice iz korpusa. U početnoj listi *X* na indeksu 6567 nalazi se sljedeća rečenica:

```
X[6567]):  ['Volim', 'pjevati', '.']
```

koja je označena sljedećim oznakama:

```
Y[6567]:  ['VERB', 'VERB', 'PUNCT'].
```

Nakon numerizacije ta ista rečenica u listi *X_encoded* ima oblik:

```
X_encoded:  [472, 6540, 2],
```

a njezine pripadajuće oznake u listi *Y_encoded* su:

```
Y_encoded:  [4, 4, 2].
```

S obzirom na to da je za efikasan rad neuronskih mreža neophodno da su ulazni podaci koji im se šalju jednakog ili sličnog oblika (ili u ovom slučaju jednakih duljina) sljedeći korak je definiranje jednakih duljina svih rečenica u korpusu jer on trenutno sadrži rečenice raznih duljina (može se provjeriti na način: `duljine = [len(seq) for seq in X_encoded])`:

- duljina najdulje rečenice: 255 `max(duljine)`
- duljina najkraće rečenice: 1 `min(duljine)`
- prosječna duljina rečenica: 20.60 `numpy.average(duljine)`
- medijan duljina rečenica: 18.0 `numpy.median(duljine)`

U prosjeku rečenice u korpusu imaju oko 20 riječi, dok najdulja rečenica ima 255 riječi, najkraća 1 riječ, a medijan duljina rečenica pokazuje da 50% rečenica ima manje od 18 riječi. Da bi se sve rečenice postavile na jednaku duljinu potrebno je ili produljiti kratke rečenice ili skratiti duge rečenice na fiksnu duljinu. Za fiksnu duljinu rečenica odabrana je duljina od 50 riječi jer u tom slučaju 97% rečenica ima manje od 50 riječi (`numpy.quantile(duljine, 0.97)` → `50`). Dakle, samo 3% rečenica korpusa (555 od ukupno 20 159 rečenica) ima više od 50 riječi i samo one će biti skraćene, tj. uklonit će se višak riječi, čime se ne gubi značajan broj podataka iz korpusa.

Za transformaciju svih rečenica u rečenice jednakih duljina u se ovom koraku koristi metoda *keras.preprocessing.sequence.pad_sequence()*[11]:

[11] https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences

```
MAX_LEN = 50
X_padded = pad_sequences(X_encoded, maxlen=MAX_LEN,
padding="pre", truncating="post")
Y_padded = pad_sequences(Y_encoded, maxlen=MAX_LEN,
padding="pre", truncating="post")
Z_padded = pad_sequences(Z_encoded, maxlen=MAX_LEN,
padding="pre", truncating="post")
```

Parametar *padding="pre"* metode *pad_sequences()* označava da će se rečenice koje imaju manje od 50 riječi s lijeve strane nadopuniti nulama, a parametar *truncating="post"* označava da će se za rečenice koje imaju više od 50 riječi ukloniti višak riječi s desne strane. Ponovno, prethodne korake postavljanja jednakih duljina rečenica potrebno je ponoviti i za skup testiranja.

Ako sada i nakon ovog koraka ponovno pogledamo rečenicu indeksa 6567 u listi *X_padded* ona izgleda ovako:

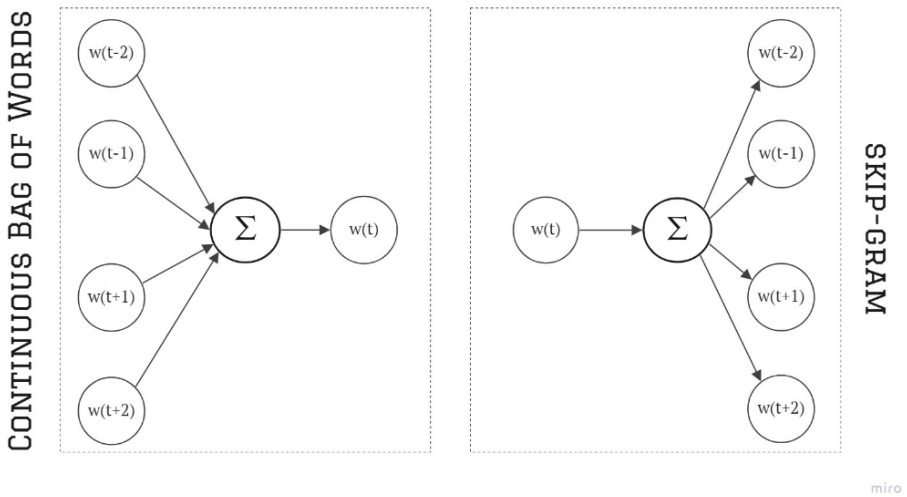
```
X_padded:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 472 6540 2],
```

a njezine oznake u listi *Y_padded* su:

```
Y_padded:  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 4 4 2].
```

Najsuvremenija metoda vektorizacije tekstova je učenje numeričkih reprezentacija riječi pomoću metoda dubokog učenja. Ovakve numeričke reprezentacije jezičnih jedinica temeljene na dubokom učenju se nazivaju *word embeddings*.

Word embedding vrsta je numeričke reprezentacije riječi koje omogućuje riječima slična značenja slične numeričke reprezentacije. *Word embedding* je zapravo tehnika gdje su pojedinačne riječi predstavljene kao vektori stvarnih vrijednosti u unaprijed definiranom vektorskom prostoru. [11] Svaka se riječ preslikava na jedan vektor, a vektorske vrijednosti se uče na način koji podsjeća na neuronsku mrežu, pa se stoga tehnika često stavlja u područje dubokog učenja. [11] *Word embedding* se može učiti ili zajedno s ciljnim NLP zadatkom (primjerice *embedding* sloj u RNN modelu) ili putem nenadzirane metode temeljene na velikom broju tekstova (primjerice *Word2vec*). Standard za razvoj unaprijed treniranih *word embeddinga* postao je Word2vec model.



Slika 19 CBOW i Skip-gram modeli

Word2vec je tehnika za procesiranje prirodnog jezika koja koristi *Continuous Bag of Words* (skraćeno CBOW) ili *Skip-gram* plitku neuronsku mrežu da bi se stvorio Word2vec model koji razumije veze između riječi u danom tekstu [44]:

- CBOW model: predviđa riječ na temelju konteksta koji se sastoji od nekoliko riječi ispred i iza trenutne riječi, a poredak riječi u kontekstu ne utječe na rezultat,
- Skip-gram model: predviđa kontekst trenutne riječi koji se temelji na samoj riječi, a cilj mu je pronaći uzorke riječi koji su korisni za predviđanje okolnih riječi unutar određenog raspona u rečenici.

Rezultat učenja modela je tzv. *embedding* matrica koja sadrži vektorske reprezentacije riječi.

Word2vec implementira se kôdom koji je dostupan u prilogu rada (Kôd 7). Pomoću Word2vec modela računaju se težine koje se u sljedećem koraku šalju neuronskoj mreži.

Prije definiranja RNN i LSTM modela potrebno je skup za treniranje podijeliti na dva dijela: skup za treniranje i skup za validaciju. Skup za treniranje modela je skup za koji se računaju težine i pristranosti te skup iz kojeg model "uči". Skup za validaciju je skup za provjeru valjanosti koji se koristi za procjenu modela tijekom treniranja te podešavanje parametara modela. U ovom slučaju koristi se metoda *train_test_split()* te se određuje parametar *test_size=0.15* koji označuje da će se početni skup za treniranje podijeliti na dva dijela tako da 75% slučajno odabranih podataka ide u skup za treniranje, a 15% u skup za validaciju:

```
VALID_SIZE = 0.15
X_train, X_validation, Y_train, Y_validation=
train_test_split(X_padded, Y, test_size=VALID_SIZE,
random_state=4)
```

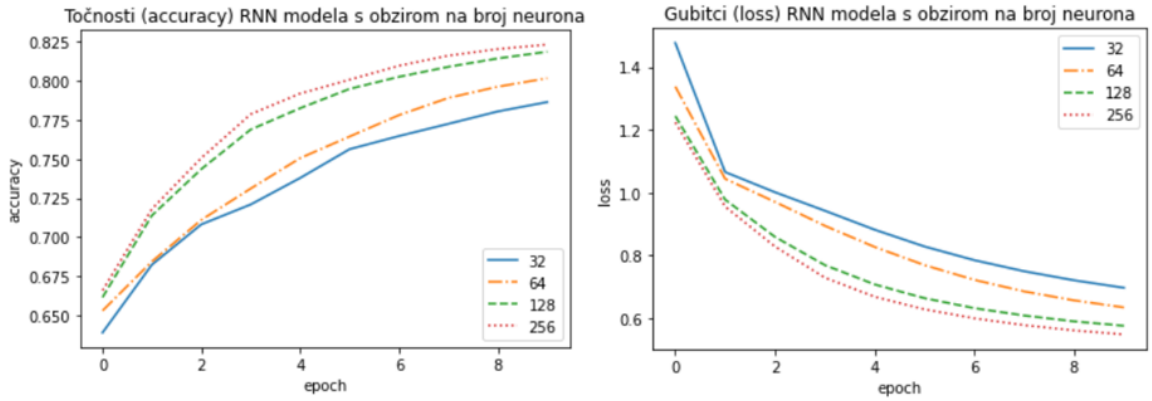
4.3. Treniranje

Nakon prethodnih koraka koji su provedeni za predprocesiranje korpusa, podaci iz skupa za treniranje imaju sljedeće dimenzije:

- skup riječi: 17135x50 (ukupno ima 17135 rečenica, a svaka je predstavljena vektorom duljine 50),
- skup UPOS oznaka: 17135x50x18 (ukupno ima 17135 rečenica, svaka je predstavljena vektorom duljine 50 (ima 50 UPOS oznaka), a svaka je oznaka predstavljena vektorom duljine 18 (odgovara broju UPOS oznaka → *one-hot encoding*),
- skup MSD oznaka: 17135x50x756 (ukupno ima 17135 rečenica, svaka je predstavljena vektorom duljine 50 (ima 50 MSD oznaka), a svaka je oznaka predstavljena vektorom duljine 756 (odgovara broju MSD oznaka → *one-hot encoding*).

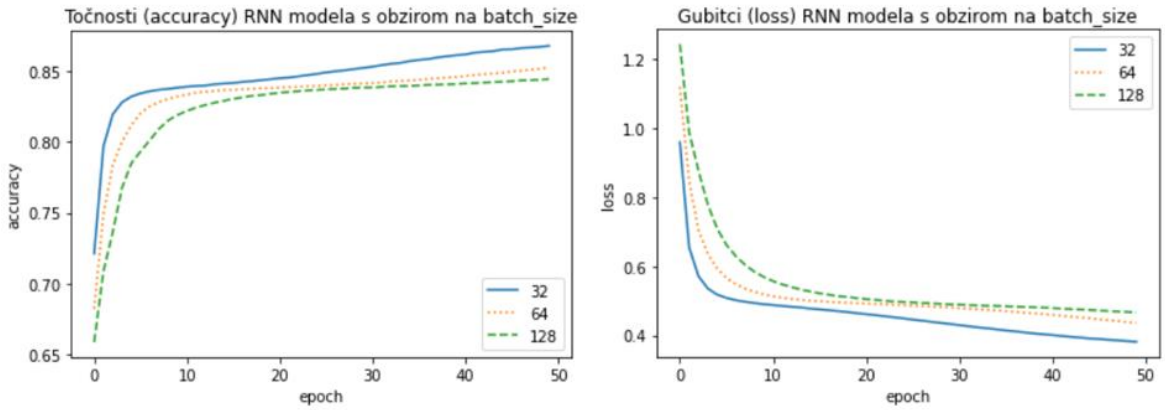
Kada neuronska mreža započne sa radom, težine se (ako nije drugačije definirano) nasumično inicijaliziraju. S obzirom na to da su težine važan dio neuronske mreže te da one odlučuju o važnosti ulaznih podataka, ovakav početak ne bi dao dobre rezultate. No, tijekom treniranja ove se težine mijenjaju, smanjuje se pogreška te na kraju mreža daje rezultate visoke točnosti.

Ukratko, treniranje se provodi tako da se težine nasumično postave na neke početne vrijednosti te se pomoću tih težina predvidi određeni broj oznaka. Ovo se predviđanje uspoređuje sa ispravnim oznakama te se pomoću funkcije gubitka pronalazi način kako smanjiti pogrešku. Nakon što se pogreška predviđanja smanji, težine se ponovno postavljaju, ponovno se predviđa određeni broj oznaka, uspoređuju se sa ispravnim oznakama, pronalazi način kako smanjiti pogrešku te tako sve ispočetka. Epohe su pojam koji označuje koliko puta će se na ovakav način prolaziti kroz model, a *batch_size* označuje broj uzoraka za koji se predviđaju oznake u svakom koraku. Primjerice, ako se definira epochs = 10 i batch_size = 32, onda će model kroz cijeli skup za treniranje proći 10 puta (svaki put podešava težine), a u svakoj epohi ili prolazu će ažurirati težine na osnovu prosječne vrijednosti gubitka na uzorku od 32 podatka.



Slika 20 Treniranje RNN modela s različitim brojem neurona u RNN sloju

Tijekom treniranja RNN modela točnost se sve više povećava što model ima više neurona u RNN sloju. Uspoređeni su modeli koji imaju RNN sloj sa 32 neurona, sa 64 neurona, sa 128 neurona i sa 256 neurona. Slika 20 prikazuje da je najveću točnost kroz sve epohe treniranja postigao model sa 256 neurona u RNN sloju, a najmanju model sa 32 neurona. S druge strane, prikazan je i gubitak, odnosno *loss* funkcija koja označuje minimiziranje pogreške. Ovdje je također model sa 256 neurona u RNN sloju postizao najmanje pogreške tijekom treniranja, a model sa 32 neurona je bio najlošiji u smanjivanju pogreške tijekom treniranja.



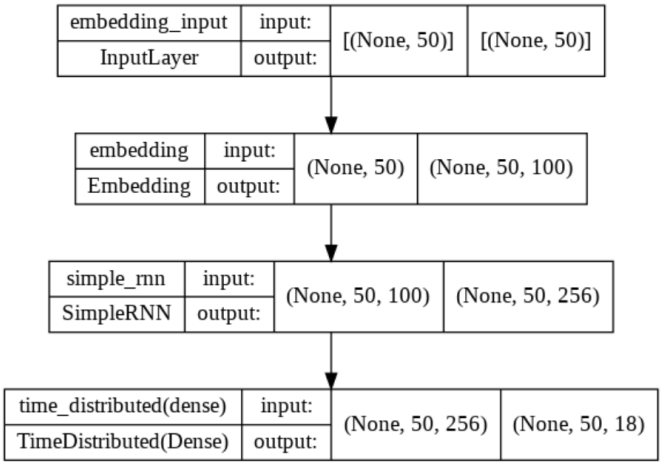
Slika 21 Treniranje RNN modela s različitim *batch_size* parametrom

Za treniranje modela važan parametar je i batch_size. Tijekom treniranja ulazi se uzimaju u grupama veličine koja odgovara parametru batch_size te se iz njih uči i predviđa. Nakon što model završi sa određenom grupom, predviđanja se uspoređuju sa točnim oznakama te se računa pogreška. Slika 21 prikazuje točnosti modela tijekom treniranja, a s obzirom na različite vrijednosti parametra batch_size. Može se uočiti da je tijekom svih epoha model sa batch_size parametrom 32 imao najveću točnost, a i pravio je najmanje pogrešaka.

S obzirom na prethodne grafičke analize, za implementaciju svih RNN modela u sljedećim koracima koristit će se RNN sloj s 256 neurona, parametar batch_size će se postaviti na 32, a modeli će se trenirati kroz 10 epoha (iako bi 50 ili 100 epoha možda dalo i bolje rješenje).

Svaka od korištenih RNN arhitektura je sekvencijalni model (*Sequential()*) koji sadrži sljedeće komponente tj. slojeve (Slika 22):

- *Embedding sloj*: Ovaj sloj izračunava vektorski model riječi (vektorizacija). Ovaj sloj je ujedno i ulazni sloj mreže, a podaci koje očekuje na ulazu trebaju imati duljinu 50 (u prethodnim koracima sve su duljine rečenica postavljene na 50 riječi). Nakon što rečenice prođu kroz ovaj sloj na izlazu i dalje imaju 50 riječi, no svaka je riječ predstavljena vektorom dimenzije 100. Također, sadrži parametar trainable – ako se postavi na True početne težine modela se ažuriraju tijekom treniranja, a ako se postavi na False težine se ne mijenjaju tijekom treniranja.;
- *RNN sloj:* Ovaj sloj na ulazu prima izlaz prethodnog sloja, tj. listu rečenica dimenzija 50x100. Nakon što podaci prođu kroz ovaj sloj na izlazu su oblika 50x256 (256 je broj neurona u sloju). Ovdje je važno postaviti parametar return_sequences=True da bi model na izlazu dao niz oznaka, a ne samo jednu oznaku.;
- *Dense sloj:* Ovo je potpuno povezani sloj koji odabire odgovarajuću POS oznaku, a budući da treba raditi na svakom elementu niza potrebno je i dodati modifikator *TimeDistributed*. U ovom se sloju dodaje i aktivacijska funkcija koja je u ovom slučaju *softmax* funkcija. Ovaj sloj služi kao izlazni sloj modela, a podaci na izlazu su oblika 50x18: svaka rečenica se sastoji od 50 riječi, a svaka riječ je predstavljena vektorom čija duljina odgovara ukupnom broju UPOS oznaka (plus 0 korištena za postavljanje jednakih duljina rečenica).



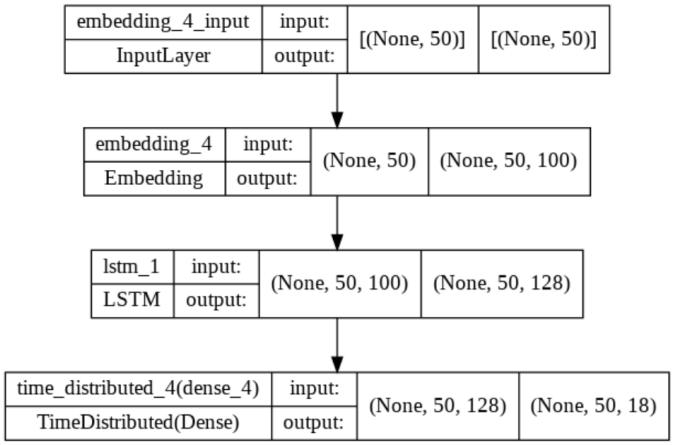
Slika 22 Arhitektura RNN modela

S obzirom na težine koje RNN model koristi tijekom treniranja implementirane su tri RNN arhitekture. Sve imaju arhitekturu sa slike (Slika 22), ali se razlikuju u načinu postavljanja početnih težina modela:

- **RNN1:** model s proizvoljno inicijaliziranim težinama koje se ne ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 11),
- **RNN2:** model s proizvoljno inicijaliziranim težinama koje se ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 12) i
- **RNN3:** model s težinama generiranim Word2vec modelom, a koje se ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 13).

Istom su arhitekturom implementirane i LSTM mreže (Slika 23), a jedina razlika je u zamjeni RNN sloja LSTM slojem. S obzirom na inicijalizaciju težina, ponovno imamo tri modela:

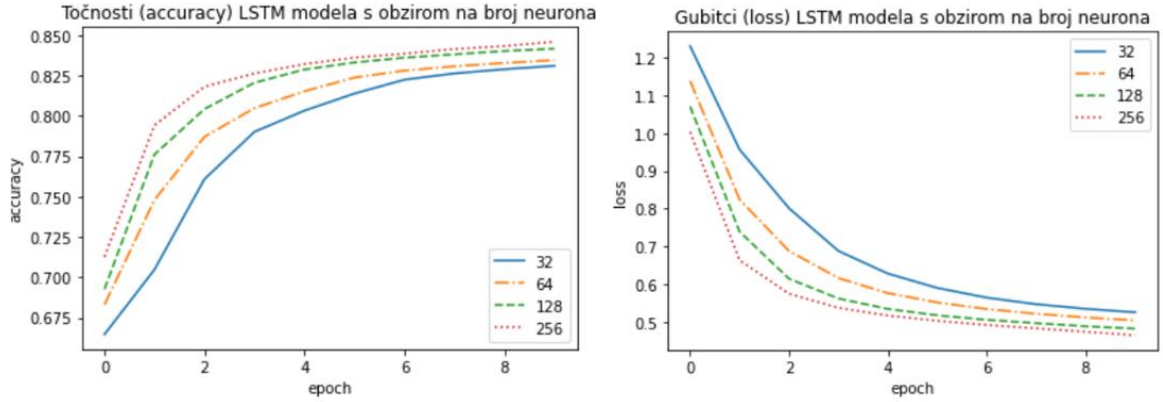
- **LSTM1:** model s proizvoljno inicijaliziranim težinama koje se ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 14),
- **LSTM2:** model s težinama generiranim Word2vec modelom, a koje se ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 15) te
- **BiLSTM:** dvosmjerni LSTM model s težinama generiranim Word2vec modelom, a koje se ažuriraju tijekom treniranja (dostupno u privitku kao Kôd 16).



Slika 23 Arhitektura LSTM modela

Slika 23 prikazuje arhitekturu LSTM modela koja je, ako je usporedimo sa slikom 22, gotovo jednaka RNN modelima. No, ovdje se umjesto RNN sloja dodaje LSTM sloj. A za dvosmjernu LSTM mrežu u taj se sloj još dodaje i *Bidirectional*() modifikator koji u LSTM na ulaz uz trenutni ulaz šalje i sljedeći ulaz.

Parametri LSTM mreže birani su na isti način kao i kod RNN mreža. S obzirom na različiti broj neurona u LSTM sloju (Slika 24), broj epoha te parametar *batch_size* (Slika 25) za LSTM odabrani su sljedeći parametri: 10 epoha, 128 neurona u LSTM sloju te *batch_size*=32 (implementacija prikazana u prilogu kao Kôd 14 i Kôd 15).



Slika 24 Treniranje LSTM modela s obzirom na broj neurona u LSTM sloju

Slika 25 Treniranje LSTM modela s obzirom na batch_size

Za POS označavanje hrvatskog jezika odlučeno je korištenje UPOS oznaka radi jednostavnosti implementacije i manjeg skupa oznaka nego kod MSD oznaka. No, kako je navedeno u prethodnim poglavljima, hrvatski je jezik morfološki dosta složen jezik te 17 UPOS oznaka nije dovoljno da se opišu sve njegove morfološke odlike. Stoga, za daljnji rad bilo bi dobro da se iz hr500k uzmu i MSD oznake te se ovdje implementirani modeli i na njima istreniraju.

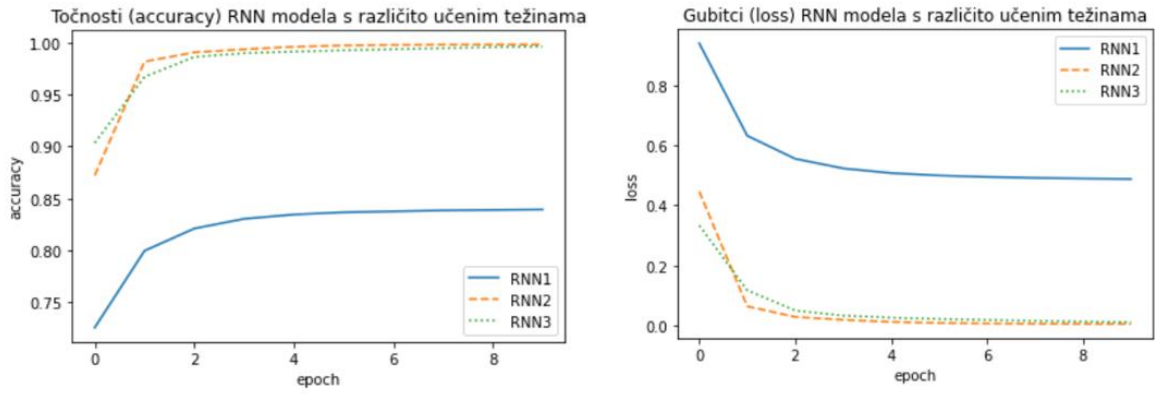
5 Evaluacija RNN i LSTM modela

Implementirane modele može se uspoređivati tijekom treniranja s obzirom na skup za validaciju te nakon treniranja uspoređujući predviđanja modela sa točnim oznakama iz skupa za testiranje.

5.1. Povijest treniranja

Za neuronsku mrežu koja se izgrađuje pomoću Kerasa postoji *Embedding()* sloj koji se može koristiti za neuronske mreže za učenje iz tekstualnih podataka. Ovaj sloj zahtijeva da ulazni podaci budu brojčane vrijednosti, tako da je svaka riječ predstavljena jedinstvenim cijelim brojem. *Embedding()* sloj inicijalizira nasumične težine i uči *embeddingse* za sve riječi u skupu podataka za treniranje, a može se koristiti na različite načine. Može se koristiti sam za učenje *embeddingsa* riječi koje se mogu spremiti i koristiti kasnije u nekom drugom modelu. Može se također koristiti kao dio modela dubokog učenja gdje se *embeddingsi* uče zajedno sa samim modelom, a može se i koristiti za učitavanje unaprijed istreniranog modela za word *embeddingse*.

Neuronske modele može se analizirati promatrajući njihovu izvedbu tijekom treniranja. Tijekom treniranja modela bilježi se metrika evaluacije za svaku epohu kroz koju model prođe. Metrike koje se promatraju su točnost (*accuracy*) i gubitak (*loss*). Pozivanjem metode *.fit()* dohvaća se objekt koji sadrži povijest treniranja te sadrži navedene metrike. Nakon što se sve tri arhitekture istreniraju njihove se metrike tijekom treniranja uspoređuju na grafu (Slika 26).



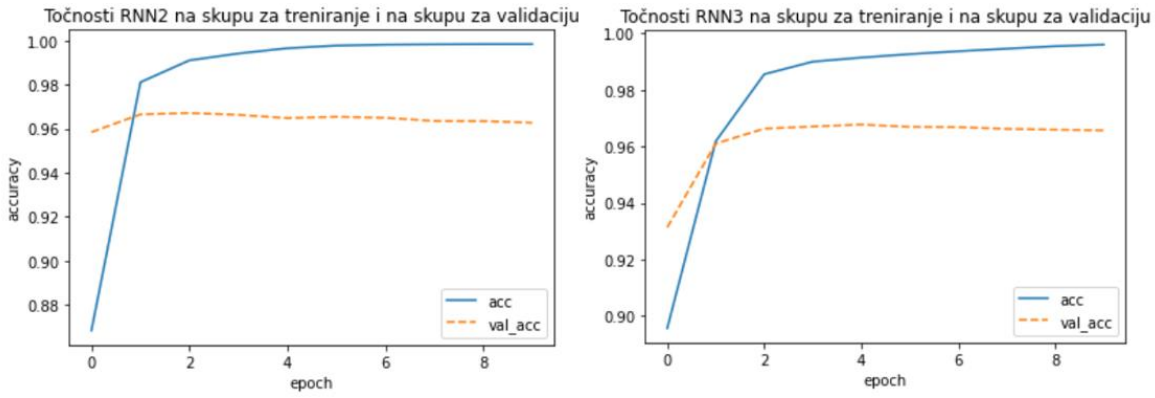
Slika 26 Usporedba evaluacije RNN modela tijekom treniranja

Slika 26 prikazuje evaluaciju metrike za vrijeme treniranja za RNN1, RNN2 i RNN3 modele, a na grafu prikazana metrika je točnost. RNN1 model je u tijekom treniranja cijelo vrijeme postizao najmanju točnost, koja kako graf prikazuje, ni u jednom trenutku nije postigla veću točnost od 85%. S druge strane, razlika u točnosti RNN2 i RNN3 modela gotovo da i nije značajna, no na samom kraju ipak je najveću točnosti imao RNN2 model. Druga promatrana metrika je gubitak (*loss)*, a koja označuje smanjenje pogreške modela. Na slici je vidljivo da je ponovno najlošiji model bio RNN1, dok su RNN2 i RNN3 modeli pogreške smanjili gotovo na nulu.

	acc %	loss %	val_acc %	val_loss %
RNN1	83.75	96.68	83.39	74.18
RNN2	99.84	46.14	96.71	16.02
RNN3	99.61	46.14	96.77	20.47

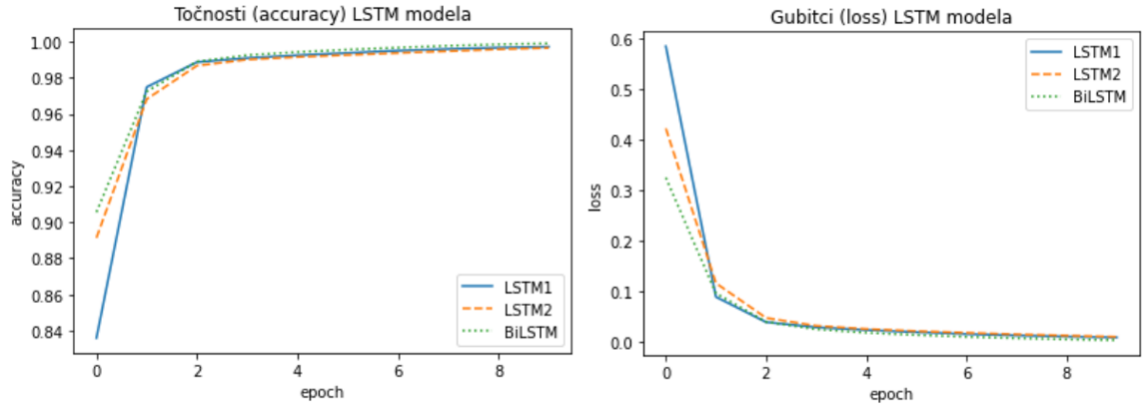
Slika 27 Metrike RNN modela tijekom treniranja

Povijest treniranja, uz točnosti i gubitke nad skupom za treniranje, pruža te iste metrike i za skup za validaciju. Tijekom treniranja model ne "uči" na ovom skupu, već ga koristi za validaciju koliko je dobro nešto "naučio". Informacije o točnosti nad skupom za treniranje i nad skupom za validaciju su korisne i kod određivanja problema koji se naziva *overfitting* Primjer ovakvog problema vidljiv je i na slici 28 gdje RNN2 i RNN3 modeli postižu značajnije veće točnosti nad skupom za treniranje nego nad skupom za validaciju. Ovo bi moglo značiti da je model previše "naučen" na skup za treniranje te bi se moglo predvidjeti da neće postizati velike točnosti nad skupom za testiranjem. Slika 27 prikazuje točnosti i gubitke za skup za treniranje (*acc* i *loss*) te za skup za validaciju (*val_acc i val_loss*). Prema rezultatima najveću točnost (99,84%) postiže RNN2 model s proizvoljno inicijaliziranim težinama u *Embedding* sloju, a koje se ažuriraju tijekom treniranja. S druge strane, najlošiju točnost (83,75%) s najvećim gubitkom (96,68%) ima RNN1 model.



Slika 28 RNN (*acc* vs *val_acc*)

Što se tiče LSTM modela, oni pokazuju gotovo jednake rezultate kao i RNN2 te RNN3 modeli. Slika 29 prikazuje grafički prikaz točnosti LSTM1, LSTM2 i BiLSTM modela tijekom treniranja. Može se reći da razlike između ova tri modela nisu velike, no najveću točnost sa najmanjim gubitkom ipak postiže BiLSTM model.

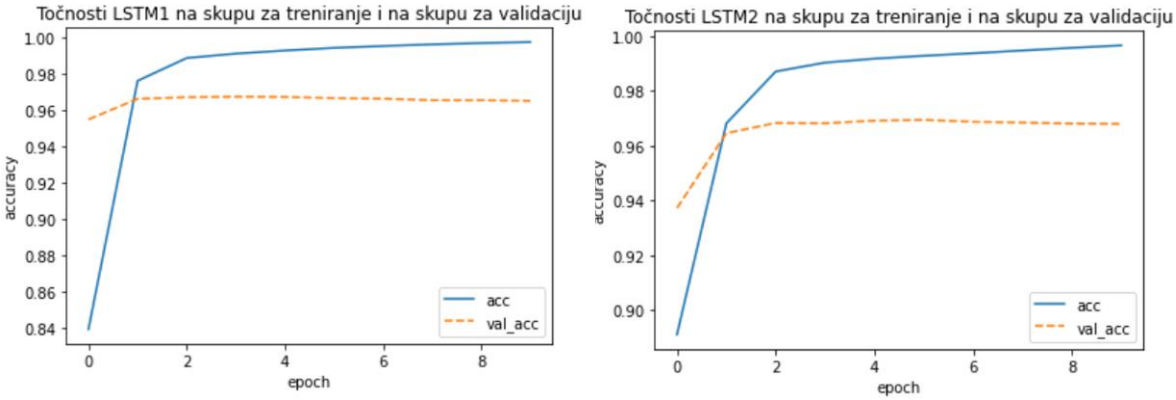


Slika 29 Usporedba evaluacije LSTM modela tijekom treniranja

Slika 31 prikazuje isti *overfitting* problem kao i kod RNN modela, a koji je vidljiv i za LSTM1 i LSTM2 modele. U ovom je slučaju razlika između točnosti modela nad skupom za treniranje i nad skupom za validaciju još izraženija (Slika 30). Ako uzmemo u obzir ove podatke, može se pretpostaviti da će i LSTM model davati loše rezultate kod skupa za testiranje.

	acc %	loss %	val_acc %	val_loss %
LSTM1	99.73	57.40	96.73	15.16
LSTM2	99.66	42.71	96.94	18.94
BiLSTM	99.92	32.78	97.52	15.79

Slika 30 Metrike LSTM modela tijekom treniranja



Slika 31 LSTM (*acc* vs *val_acc*)

5.2. Preciznost, točnost i F1

Prilikom predviđanja POS oznaka mogu se pojaviti četiri vrste ishoda:

- TP (*true positives*): kada model točno predvidi da određenoj riječi pripada određena POS oznaka;
- TN (*true negative*): kada model točno predvidi da određenoj riječi ne pripada određena POS oznaka;
- FP (*false positive*): kada model pogrešno predvidi da određenoj riječi pripada određena POS oznaka;
- FN (*false negative*): kada model pogrešno predvidi da određenoj riječi ne pripada određena POS oznaka.

Mjere kojima se mogu evaluirati modeli za POS označavanje su: točnost (engl. *accuracy*), preciznost (engl. *precision*), opoziv (engl. *recall*) i f-score (engl. *f-score*).

Točnost modela se definira kao postotak točnih predviđanja, a može se jednostavno izračunati ako se podijeli ukupan broj točnih predviđanja s brojem ukupnih predviđanja:

$$točnost = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

Preciznost modela se definira kao udio relevantnih riječi (istinskih pozitivnih) među svim riječima za koje je predviđeno da im pripada određena POS oznaka (svi pozitivni):

$$preciznost = \frac{TP}{TP + FP} \tag{16}$$

Opoziv se definira kao udio riječi za koje je predviđeno da im pripada određena POS oznaka u odnosu na sve riječi kojima doista pripada ta POS oznaka:

$$opoziv = \frac{TP}{TP + FN} \tag{17}$$

F-score se definira kao kombinacija preciznosti i opoziva da bi se dobila jedna mjera za procjenu:

$$f - score = 2 * \frac{preciznost * opoziv}{preciznost + opoziv} \tag{18}$$

Za svaku od implementiranih RNN i LSTM arhitektura izračunati su točnost i preciznost (Slika 32) .Kao model s najvećim postotkom točno pridijeljenih POS oznaka pokazao se jednostavni RNN model sa nasumično inicijaliziranim težinama koje se ne treniraju i to sa

točnošću označavanja od 77,96%. Najlošijim se modelom pokazao RNN model sa proizvoljno definiranim težinama koje se treniraju i to sa točnošću od 76,57%. Najveću preciznost, od 29,98%, je pokazao BiLSTM model, a najmanju, od 28,10% također RNN model sa proizvoljno definiranim težinama koje se ne treniraju. Razlog ovakvih rezultata može se povezati sa prethodno spomenutim *overfitting* problemom.

	rnn1	rnn2	rnn3	lstm1	lstm2	bilstm
accuracy	0.779581	0.765734	0.766100	0.766602	0.766587	0.767403
precision	0.281014	0.295468	0.296886	0.296574	0.296977	0.299782
f1-score	0.779581	0.765734	0.766100	0.766602	0.766587	0.767403

Slika 32 Metrika modela nad skupom za treniranje

	precision	recall	f1-score	support
0	1.00	1.00	1.00	83443
1-noun	0.39	0.43	0.41	10581
2-punct	0.76	0.76	0.76	6537
3-adj	0.21	0.18	0.19	5544
4-verb	0.18	0.19	0.19	4775
5-adp	0.58	0.60	0.59	4456
6-aux	0.64	0.63	0.63	3236
7-adv	0.14	0.15	0.15	2550
8-propn	0.07	0.05	0.06	2382
9-cconj	0.54	0.53	0.53	2217
10-pron	0.07	0.08	0.08	1984
11-det	0.04	0.04	0.04	1915
12-sconj	0.61	0.58	0.60	1485
13-part	0.11	0.10	0.11	937
14-num	0.02	0.01	0.02	846
15-x	0.04	0.02	0.03	631
16-intj	0.00	0.00	0.00	48
17-sym	0.00	0.00	0.00	33
accuracy			0.77	133600
macro avg	0.30	0.30	0.30	133600
weighted avg	0.76	0.77	0.77	133600

Slika 33 BiLSTM izvještaj o klasifikaciji

Ako se za BiLSTM model provjere točnosti predviđanja pojedinačno za svaku od UPOS oznaka (Slika 33) može se uočiti da je model najviše grešaka pravio predviđajući oznake *num* (brojevi), *x* (ostale ili nepoznate riječi) i *propn* (vlastite imenice). Također, najveću točnost koju BiLSTM model postiže kod POS označavanja hrvatskog jezika je 77%.

Zaključak

Označavanje vrsta riječi ili POS označavanje postupak je u obradi prirodnog jezika koji svakoj riječi u tekstu pridjeljuje pripadajuću joj oznaku vrste riječi. POS označavanje samo po sebi ne rješava određeni problem u obradi prirodnog jezika, ali je preduvjet za mnoge druge zadatke kao što su analiza sentimenta, prevođenje teksta, provjera gramatike, prepoznavanje govora i mnogih drugih. Ako je POS označavanje loše izvedeno to negativno utječe na sve zadatke koji slijede nakon njega te iz tog razloga mnogi autori navode važnost pronalaska modela koji će najuspješnije riješiti ovaj zadatak.

U ovom je radu dan pregled klasičnih metoda za POS označavanje te primjena neuronskih mreža kao njihovog poboljšanja. Opisan je rad RNN i LSTM modela neuronskih mreža te je njihova primjena u POS označavanju prikazana kroz pregled dostupnih istraživanja. S obzirom na to da se znanstveni članci i istraživanja o primjeni neuronskih mreža u POS označavanju u velikom broju odnose na engleski jezik, zanimljivom se pokazala ideja istraživanja označavanja vrsta riječi hrvatskog jezika. Hrvatski jezik se smatra morfološki veoma bogatim jezikom što za računalnu obradu teksta na hrvatskom jeziku predstavlja veći izazov nego za obradu teksta na engleskom jeziku. Označavanje vrsta riječi engleskog jezika u obradi je prirodnog jezika već dobro istražena tema te su izgrađeni različiti modeli sa preciznostima i do 98%.

U radu je prikazana implementacija RNN i LSTM arhitektura kako bi se dobili modeli koji bi naučili označavati tekstove hrvatskog jezika UPOS oznakama. Rezultati su pokazali, u odnosu na dosadašnja istraživanja, da istrenirani modeli postižu znatno manje točnosti na skupu za testiranje. Ovakva bi se pojava mogla objasniti kroz čest problem kod neuronskih mreža, a to je *overfitting*. Implementirani su modeli za označavanje vrsta riječi hrvatskog jezika UPOS oznakama postigli najveću točnost od 77,96%.

U slučaju flektivnih jezika kao što je hrvatski usporedba s postojećim radovima za POS označavanje hrvatskog jezika nije jednostavna zbog razlika u korištenom testnom skupu i skupu oznaka. Iz toga razloga se javlja potreba za daljnjim istraživanjem i temeljitom usporedbom različitih arhitektura neuronskih mreža za zadatak POS i MSD označavanja.

Literatura

[1] ACEDAŃSKI, S. A Morphosyntactic Brill Tagger for Inflectional Languages, *Advances in Natural Language Processing*, 6233 (2010), 3-14

[2] AGIĆ, Ž. i TADIĆ, M. Evaluating Morphosyntactic Tagging of Croatian Texts, *Proceedings of the 5th International Conference on Language Resources and Evaluation* (2006)

[3] AGIĆ, Ž., LJUBEŠIĆ, N. i MERKLER, D. Lemmatization and Morphosyntactic Tagging of Croatian and Serbian, *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing* (2013), 48–57

[4] AGIĆ, Ž., Tadić, M. i Dovedan, Z. Combining Part-of-Speech Tagger and Inflectional Lexicon for Croatian, *Proceedings of the 6th Language Technologies Conference* (2008), 116

[5] AGIĆ, Ž., Tadić, M. i Dovedan, Z. Improving Part-of-Speech Tagging Accuracy for Croatian by Morphological Analysis, *Informatica (Ljubljana)*, 32, 4 (2008), 445–451

[6] AGIĆ, Ž., Tadić, M. i Dovedan, Z. Tagger Voting Improves Morphosyntactic Tagging Accuracy on Croatian Texts, *Proceedings of the 32nd International Conference on Information Technology Interfaces* (2010), 61–66

[7] ALKHWITER, W. i AL-TWAIRESH, N. Part-of-speech tagging for Arabic tweets using CRF and Bi-LSTM, *Computer Speech and Language*, 65 (2021), 101138

[8] ANANDA, M.R., HANIFMUTI, M.Y. i ALFINA, I. A Hybrid of Rule-based and HMM-based Part-of-Speech Tagger for Indonesian, *International Conference on Asian Language Processing* (2021), 280–285

[9] BALWANT, M.K. Bidirectional LSTM Based on POS tags and CNN Architecture for Fake News Detection, https://www.webofscience.com/wos/woscc/full-record/WOS:000525828100053, 19. rujna 2022.

[10] BOLAÑOS, X. Natural Language Processing and Machine Learning, *Encora*, https://www.encora.com/insights/natural-language-processing-and-machine-learning, 5. rujna 2022.

[11] BROWNLEE, J. What Are Word Embeddings for Text?, *Machine Learning Mastery*. https://machinelearningmastery.com/what-are-word-embeddings, 10. rujna 2022.

[12] CHICHE, A. i YITAGESU, B. Part of speech tagging: a systematic review of deep learning and machine learning approaches, *J Big Data*, 9, 1 (2022), 10

[13] DEVOPEDIA, Part-of-Speech Tagging, https://devopedia.org/part-of-speech-tagging, 5. rujna 2022.

[14] DINARELLI, M. i TELLIER, I. New Recurrent Neural Network Variants for Sequence Labeling, (2016) https://hal.archives-ouvertes.fr/hal-01489955, 15. rujna 2022.

[15] ELEMENTS OF AI. Besplatni internetski uvod u umjetnu inteligenciju za nestručnjake. https://course.elementsofai.com, 9. rujna 2022.

[16] FARIZKI WICAKSONO A. i PURWARIANTI, A. HMM Based Part-of-Speech Tagger for Bahasa Indonesia, *Proceedings of the 4th International MALINDO (Malaysian-Indonesian Language) Workshop* (2010)

[17] GUPTA, M. POS Tagging using Hidden Markov Models (HMM) & Viterbi algorithm in NLP mathematics explained, *Data Science in your pocket*, https://medium.com/data-science-in-your-pocket/pos-tagging-using-hidden-markov-models-hmm-viterbi-algorithm-in-nlp-mathematics-explained-d43ca89347c4 19. rujna 2022.

[18] HASAN, F.M. Comparison of different POS tagging techniques for some South Asian languages. *Diplomski rad.* BRAC University, 2006.

[19] HIRSCHBERG, J. I MANNING, C.D. Advances in natural language processing, *Science*, 349, 6245 (2015), 261-266

[20] HJOUJ, M., ALARABEYYAT, A. i OLAB, I. Rule Based Approach for Arabic Part of Speech Tagging and Name Entity Recognition, *International Journal of Advanced Computer Science and Applications*, 7, 6 (2016)

[21] HOESEN, D. i PURWARIANTI, A. Investigating Bi-LSTM and CRF with POS Tag Embedding for Indonesian Named Entity Tagger, *International Conference on Asian Language Processing* (2018), 35–38

[22] INVESTOPEDIA. Neural Network Definition. https://www.investopedia.com/terms/n/neuralnetwork.asp, 9. Rujna 2022.

[23] JUHÁR, J., STAŠ, J. i HLADEK, D. Dagger: The Slovak morphological classifier, *Proceedings ELMAR-2012* (2012)

[24] JURAFSKY, D. i MARTIN, J.H. Speech and Language Processing, 2 izd. Prentice Hall, 2008.

[25] KUMAR, A. Hidden Markov Models Explained with Examples, *Data Analytics*, https://vitalflux.com/hidden-markov-models-concepts-explained-with-examples/, 6. rujna 2022.

[26] KUMAR, N.K.S. i MALARVIZHI, N. Bi-directional LSTM-CNN Combined method for Sentiment Analysis in Part of Speech Tagging (PoS), *International Journal of Speech Technologies*, 23, 2 (2020), 373–380

[27] KUMAR, S., HU, Y.C., QAISAR, S. i SRINIVASAN, K. Effective Pre-processing and Normalization Techniques for COVID-19 Twitter Streams with POS Tagging via Lightweight Hidden Markov Model, *Journal of Sensors* (2022)

[28] KUMAWAT, D. i JAIN, V. POS Tagging Approaches: A Comparison, *International Journal of Computer Applications*, 118 (2015), 32–38

[29] LAM CING, D. i Soe, K. Improving accuracy of Part-of-Speech (POS) tagging using hidden markov model and morphological analysis for Myanmar Language, *International Journal of Electrical and Computer Engineering*, 10 (2020), 2023

[30] LI, H., MAO, H. i WANG, J. Part-of-Speech Tagging with Rule-Based Data Preprocessing and Transformer, *Electronics*, 11, 1 (2021)

[31] LORINCZ, B., NUTU, M. I STAN, A. Romanian Part of Speech Tagging using LSTM Networks, *IEEE 15th International Conference on Intelligent Computer Communication and Processing* (2019), 223–228.

[32] LJUBEŠIĆ, N. i DOBROVOLJC, K. What does Neural Bring? Analysing Improvements in Morphosyntactic Annotation and Lemmatisation of Slovenian, Croatian and Serbian, *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing* (2019), 29–34

[33] MAIMAITI, M. et al. Bidirectional Long Short-Term Memory Network with a Conditional Random Field Layer for Uyghur Part-Of-Speech Tagging, *Information*, 8 (2017), 157

[34] MARQUES, N. i LOPES, G. A Neural Network Approach to Part-of-Speech Tagging, https://www.researchgate.net/publication/250806272_A_Neural_Network_Approach_to_Part-of-Speech_Tagging, 2008.

[35] MEFTAH, S., SEMMAR, N., SADAT, F. i RAAIJMAKERS, S. Using Neural Transfer Learning for Morpho-syntactic Tagging of South-Slavic Languages Tweets, *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects* (2018), 235–243

[36] MEGYESI, B. Brill's rule-based PoS tagger. *Diplomski rad*. University of Stockholm, 2001.

[37] MOHAMED, H., OMAR, N. i AZIZ, M. Penalizing unknown words' emissions in hmm pos tagger based on Malay affix morphemes, *Journal of Fundamental and Applied Sciences*, 9 (2017)

[38] PATHMIND. A Beginner's Guide to Neural Networks and Deep Learning. http://wiki.pathmind.com/neural-network 6. rujna 2022.

[39] PATTNAIK, S. i NAYAK, A.K. A Modified Markov-Based Maximum-Entropy Model for POS Tagging of Odia Text, *IJDSST*, 14, 1 (2022), 1–24

[40] PERADIN, H. i ŠNAJDER, J. Towards a Constraint Grammar Based Morphological Tagger for Croatian, *International Conference on Text, Speech and* Dialogue (2012), 174–182

[41] PHI, M. Illustrated Guide to LSTM's and GRU's: A step by step explanation, *Medium*, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 09. rujna 2022.

[42] QIAO, M., BIAN, W., XU, R.Y. I TAO, D. Diversified Hidden Markov Models for Sequential Labeling, *IEEE 32nd International Conference on Data Engineering (*2016), 1512-1513

[43] SIMPLILEARN. Recurrent Neural Network (RNN) Tutorial: Types and Examples https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn, 6. rujna 2022.

[44] SVOBODA, L. i BELIGA, S. Evaluation of Croatian Word Embeddings, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation* (2018), 1512-1518

[45] ŠUMAN, S. Pregled metoda obrade prirodnih jezika i strojnog prevođenja, *Zbornik Veleučilišta Rijeka*, 9, 1 (2021), 371–384

[46] ULČAR, M. i ROBNIK-SIKONJA, M. FinEst BERT and CroSloEngual BERT: less is more in multilingual models, *International Conference on Text, Speech, and Dialogue* (2020), 104-111

[47] ZVORNICANIN, E. Differences Between Bidirectional and Unidirectional LSTM, *Baeldung on Computer Science*, https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm, 10. rujna 2022.

[48] ŽIVKOVIĆ-MANDIĆ, J. Gramatika hrvatskoga jezika. *Oktatási Hivatal*, 2021.

Sažetak

Označavanje vrsta riječi jedan je od temeljnih zadataka u području obrade prirodnog jezika. Ono samo po sebi ne rješava određeni problem u obradi prirodnog jezika, ali je preduvjet za mnoge druge procese, kao što su analiza sentimenta, prevođenje teksta, provjera gramatike ili prepoznavanje govora. Označavanje vrsta riječi temeljeno na pravilima, transformacijama ili stohastičkim metodama opisani su i uspoređeni sa korištenjem neuronskih mreža u prvom dijelu rada. Kao praktični dio rada implementirane su RNN i LSTM arhitekture neuronske mreže za treniranje modela koji mogu provoditi automatsko označavanje vrsta riječi za dani tekst pisan hrvatskim jezikom. Kako bi se utvrdila efikasnost uporabe modela neuronske mreža za ovakav zadatak, na kraju rada modeli su testirani i evaluirani. Cilj ovog rada je prikazati primjenu neuronskih mreža za razvoj automatiziranog označivača vrsta riječi hrvatskog jezika.

Summary

Part-of-speech tagging is one of the main tasks of natural language processing. On its own, it does not solve a specific problem of natural language processing but it is a part of preprocessing in sentiment analysis, text translation, grammar checking or speech recognition. The rule-based, transformation based and stochastic methods of part-of-speech tagging are described and compared to neural network methods in the first part of this paper. As a practical part of the thesis, RNN and LSTM models are implemented in model training for an automatic part-of-speech tagger of the Croatian language. To check the efficiency of using the neural network approach for this task, models are tested and evaluated. The aim of this thesis is to implement neural networks for an automatic part-of-speech tagger for the Croatian language.

Skraćenice

AI	*Artificial Intelligence*	umjetna inteligencija
BiLSTM	*Bidirectional LSTM*	dvosmjerna LSTM mreža
BiRNN	*Bidirectional RNN*	dvosmjerna RNN mreža
DL	*Deep Learning*	duboko učenje
HMM	*Hidden Markov Model*	skriveni Markovljev model
LSTM	*Long Short Term Memory*	……
ML	*Machine Learning*	strojno učenje
MSD	*Morphosyntactic Descriptors*	morfosintaksni opisi (oznake)
NLP	*Natural Language Processing*	obrada prirodnog jezika
NN	*Neural Network*	neuronska mreža
POS	*Part-Of-Speech*	vrste riječi
RNN	*Reccurent Neural Network*	povratna neuronska mreža
UPOS	*Universal Part-Of-Speech*	univerzalne oznake vrsta riječi

Privitak

Instalacija programske podrške

Opisani je programski kôd napisan u Jupyter bilježnici (engl. *Jupyter Notebook*) koja omogućava kreiranje, dijeljenje i uređivanje dokumenata u kojima se može izvršiti Python kôd, napisati bilješke, umetnuti jednadžbe, vizualizirati rezultati i dr. Za pokretanje kôda koji se nalazi u ovakvoj bilježnici nije potrebna instalacija programske podrške, već samo Google račun te otvaranje bilježnice pomoću usluge u oblaku - Google Colab. Izvorni članak koji je poslužio kao inspiracija za praktični dio rada dostupan je preko poveznice: https://towardsdatascience.com/pos-tagging-using-rnn-7f08a522f849, a sadrži detaljniju analizu samog kôda i korištenih metoda.

Upute za korištenje programske podrške

Korišteni korpus hrvatskog jezika potrebno je preuzeti, a javno je dostupan preko poveznice: https://huggingface.co/datasets/classla/hr500k/tree/main. Preuzeti se podaci mogu učitati pokretanjem programskog kôda sa slike (Kôd 2).

```
import gensim
import numpy as np
from io import open
import pandas as pd
import tensorflow as tf
from conllu import parse_incr
from google.colab import files
from matplotlib import pyplot as plt
from keras.preprocessing.text import Tokenizer
from gensim.models import Word2Vec, KeyedVectors
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.utils import to_categorical, plot_model
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from tensorflow.keras.layers import Dense, Embedding, TimeDistributed
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Bidirectional
from sklearn.metrics import accuracy_score, precision_score, f1_score
```

Kôd 1 Učitavanje potrebnih Python biblioteka

```
uploaded = files.upload()
for fn in uploaded.keys():
   print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Kôd 2 Učitavanje korpusa sa lokalnog računala

```
def read_data(path):
  with open(path, "r", encoding='utf-8') as f:
    sent_id = ''
    text = ''
    tokens = []
    xpos_tags = []
    upos_tags = []
    data_id = 0
    for line in f:
      if line and not line == '\n':
        if line.startswith('#'):
          if line.startswith('# sent_id'):
            if tokens:
              yield {'sent_id': sent_id, 'text': text,
                     'tokens': tokens, 'upos_tags': upos_tags,
                     'xpos_tags': xpos_tags }
              tokens = []
              upos_tags = []
              xpos_tags = []
              data_id += 1
            sent_id = line.split(' = ')[1].strip()
          elif line.startswith('# text'):
            text = line.split(' = ')[1].strip()
        elif not line.startswith('_'):
          splits = line.split('\t')
          tokens.append(splits[1].strip())
          upos_tags.append(splits[3].strip())
          xpos_tags.append(splits[4].strip())
      yield { 'sent_id': sent_id,'text': text,'tokens': tokens,
              'upos_tags': upos_tags, 'xpos_tags': xpos_tags }
```

Kôd 3 Metoda za učitavanje podataka iz .conllu datoteke hr500k korpusa

```
train_hr500k = read_data("/pathToFile/train_ner.conllu")
test_hr500k = read_data("/pathToFile/test_ner.conllu")

# skup za treniranje:
X = [] # lista recenica gdje je svaka recenica lista rijeci
Y = [] # lista UPOS oznaka (universal POS)
Z = [] # lista MSD oznaka  (Multext-East)
for t in train_hr500k:
    X.append(t["tokens"])
    Y.append(t['upos_tags'])
    Z.append(t['xpos_tags'])

# skup za testiranje:
Xt = [] # lista recenica gdje je svaka recenica lista rijeci
Yt = [] # lista UPOS oznaka (universal POS)
Zt = [] # lista MSD oznaka  (Multext-East)
for t in test_hr500k:
    Xt.append(t["tokens"])
    Yt.append(t['upos_tags'])
    Zt.append(t['xpos_tags'])
```

Kôd 4 Definiranje skupova riječi, UPOS oznaka i MSD oznaka

```
#numericka reprezentacija liste rijeci:

X_tokenizer = Tokenizer()

X_tokenizer.fit_on_texts(X)

X_encoded = X_tokenizer.texts_to_sequences(X)

#numericka reprezentacija liste UPOS oznaka:

Y_tokenizer = Tokenizer()

Y_tokenizer.fit_on_texts(Y)

Y_encoded = Y_tokenizer.texts_to_sequences(Y)

#numericka reprezentacija liste MSD oznaka:

Z_tokenizer = Tokenizer()

Z_tokenizer.fit_on_texts(Z)

Z_encoded = Z_tokenizer.texts_to_sequences(Z)
```

Kôd 5 Numerička reprezentacija riječi (isto ponoviti i na skupu za testiranje)

```
MAX_LEN = 50
X_padded = pad_sequences(X_encoded, maxlen=MAX_LEN, padding="pre",
truncating="post")
Y_padded = pad_sequences(Y_encoded, maxlen=MAX_LEN, padding="pre",
truncating="post")
Z_padded = pad_sequences(Z_encoded, maxlen=MAX_LEN, padding="pre",
truncating="post")
```

Kôd 6 Postavljanje jednakih duljina rečenica (isto ponoviti i na skupu za testiranje)

```
EMBEDDING_SIZE  = 100
VOCABULARY_SIZE = len(X_tokenizer.word_index) + 1
embedding_weights_X = np.zeros((VOCABULARY_SIZE, EMBEDDING_SIZE))
embedding_weights_Xt = np.zeros((VOCABULARY_SIZE, EMBEDDING_SIZE))

word2vec = Word2Vec(sentences=X, size=EMBEDDING_SIZE, iter=100)
word_vectors = word2vec.wv
word_vectors.save('vectors.kv')
vectors = KeyedVectors.load('vectors.kv')
word2id = X_tokenizer.word_index
for word, index in word2id.items():
    try:
        embedding_weights_X[index, :] = vectors[word]
    except KeyError:
        pass

word2vec = Word2Vec(sentences=Xt, size=EMBEDDING_SIZE, iter=100)
word_vectors = word2vec.wv
word_vectors.save('vectors.kv')
vectors = KeyedVectors.load('vectors.kv')
word2id = Xt_tokenizer.word_index
for word, index in word2id.items():
    try:
        embedding_weights_Xt[index, :] = vectors[word]
    except KeyError:
        pass
```

Kôd 7 Implementacija Word2vec modela za skup riječi

```
# vektorizacija oznaka skupa za treniranje:
Y = to_categorical(Y_padded)       # one-hot encoding UPOS oznaka
Z = to_categorical(Z_padded)       # one-hot encoding MSD oznaka

# vektorizacija oznaka skupa za testiranje:
Yt = to_categorical(Yt_padded)     # one-hot encoding UPOS oznaka
Zt = to_categorical(Zt_padded)     # one-hot encoding MSD oznaka
```

Kôd 8 One-hot vektorizacija UPOS i MSD oznaka

```
MAX_LEN = 50
EMBEDDING_SIZE  = 100
NUM_CLASSES = Y.shape[2]
VOCABULARY_SIZE = len(X_tokenizer.word_index) + 1
```

Kôd 9 Postavljanje parametara za neuronske mreže

```
VALID_SIZE = 0.15
X_train, X_validation, Y_train, Y_validation, Z_train, Z_validation =
train_test_split(X_padded, Y, Z, test_size=VALID_SIZE, random_state=4)
```

Kôd 10 Podjela na skup za treniranje i validaciju

```
rnn1_model = Sequential()
rnn1_model.add(Embedding(input_dim = VOCABULARY_SIZE, output_dim =
EMBEDDING_SIZE, input_length = MAX_LEN, trainable = False))
rnn1_model.add(SimpleRNN(256, return_sequences=True))
rnn1_model.add(TimeDistributed(Dense(NUM_CLASSES,
activation='softmax')))
rnn1_model.compile(loss = 'categorical_crossentropy', optimizer =
'adam', metrics = ['acc'])
rnn1_train = rnn1_model.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 11 RNN with arbitrarily initialized, untrainable embeddings

```
rnn2_model = Sequential()
rnn2_model.add(Embedding(input_dim = VOCABULARY_SIZE, output_dim =
EMBEDDING_SIZE, input_length = MAX_LEN, trainable = True))
rnn2_model.add(SimpleRNN(256, return_sequences=True))
rnn2_model.add(TimeDistributed(Dense(NUM_CLASSES,
activation='softmax')))
rnn2_model.compile(loss = 'categorical_crossentropy', optimizer =
'adam', metrics = ['acc'])
rnn2_train = rnn2_model.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 12 RNN with arbitrarily initialized, trainable embeddings

```
rnn3_model = Sequential()
rnn3_model.add(Embedding(input_dim = VOCABULARY_SIZE, output_dim =
EMBEDDING_SIZE, input_length = MAX_LEN, weights = [embedding_weights_X],
trainable = True))
rnn3_model.add(SimpleRNN(256, return_sequences=True))
rnn3_model.add(TimeDistributed(Dense(NUM_CLASSES,
activation='softmax')))
rnn3_model.compile(loss = 'categorical_crossentropy', optimizer =
'adam', metrics = ['acc'])
rnn3_train = rnn3_model.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 13 RNN with trainable word2vec embeddings

```
lstm1 = Sequential()
lstm1.add(Embedding(input_dim = VOCABULARY_SIZE,  output_dim =
EMBEDDING_SIZE, input_length  = MAX_LEN, trainable = True ))
lstm1.add(LSTM(128, return_sequences=True))
lstm1.add(TimeDistributed(Dense(NUM_CLASSES, activation='softmax')))
lstm1.compile(loss = 'categorical_crossentropy', optimizer =
'adam',metrics = ['acc'])
lstm1_train = lstm1.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 14 LSTM with arbitrarily initialized, trainable embeddings

```
lstm2 = Sequential()
lstm2.add(Embedding(input_dim = VOCABULARY_SIZE,  output_dim =
EMBEDDING_SIZE, input_length  = MAX_LEN, weights=[embedding_weights_X],
trainable = True ))
lstm2.add(LSTM(128, return_sequences=True))
lstm2.add(TimeDistributed(Dense(NUM_CLASSES, activation='softmax')))
lstm2.compile(loss = 'categorical_crossentropy', optimizer =
'adam',metrics = ['acc'])
lstm2_train = lstm2.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 15 LSTM with trainable word2vec embeddings

```
blstm = Sequential()
blstm.add(Embedding(input_dim = VOCABULARY_SIZE,  output_dim =
EMBEDDING_SIZE, input_length  = MAX_LEN, weights=[embedding_weights_X],
trainable = True ))
blstm.add(Bidirectional(LSTM(128, return_sequences=True)))
blstm.add(TimeDistributed(Dense(NUM_CLASSES, activation='softmax')))
blstm.compile(loss = 'categorical_crossentropy', optimizer =
'adam',metrics = ['acc'])
blstm_train = blstm.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_data=(X_validation, Y_validation))
```

Kôd 16 BiLSTM

```
loss, accuracy = rnn_model.evaluate(X_test, Y_test, verbose = 1)
print("Loss: {0},\nAccuracy: {1}".format(loss, accuracy))

loss, accuracy = lstm_model.evaluate(X_test, Y_test, verbose = 1)
print("Loss: {0},\nAccuracy: {1}".format(loss, accuracy))
```

Kôd 17 Evaluacija RNN i LSTM modela

```
modeli = { 'rnn1': rnn1_model, 'rnn2': rnn2_model, 'rnn3': rnn3_model,
          'lstm1': lstm1, 'lstm2': lstm2, 'bilstm': blstm }

metrics = {}
test_labels = Yt.argmax(axis=-1)
tag_map = dict(map(reversed, Yt_tokenizer.word_index.items()))

classes = ['0']
tuples = tag_map.items()
for tupl in tuples:
  classes.append("-".join([str(value)for value in tupl]))

for key in modeli:
  predictions = modeli[key].predict(Xt_padded)
  predicted_labels = predictions.argmax(axis=-1)
  tlabels = [label for labels in test_labels for label in labels]
  plabels = [label for labels in predicted_labels for label in labels]
  metrics[key] = {
       'accuracy': accuracy_score(tlabels, plabels),
       'precision': precision_score(tlabels, plabels, average='macro'),
       'f1-score': f1_score(tlabels, plabels, average='micro')}

results = pd.DataFrame.from_dict(metrics)

predictions = blstm.predict(Xt_padded)
predicted_labels = predictions.argmax(axis=-1)
tlabels = [label for labels in test_labels for label in labels]
plabels = [label for labels in predicted_labels for label in labels]
matrix = confusion_matrix(tlabels, plabels)
result=classification_report(tlabels, plabels, target_names =  classes)
```

Kôd 18 Metrike evaluacije

OZNAKA (TAG)	ZNAČENJE	PRIMJERI U HRVATSKOM JEZIKU
ADJ	**pridjevi** (eng. *adjectives*): promjenjive riječi kojima se izriče osobina ili svojstvo onoga što je označeno imenicom	lijep, lijepo, zao, visoko, zeleno, gorko, slatko, bratovo, Markovo, najglasnije ...
ADP	**prijedlozi** (eng. *adpositions*): nepromjenjive riječi kojima se označuju odnosi među bićima, stvarima i pojavama, tj. koje izriču različite odnose između onoga što označuju imenice ili zamjenice	zbog, radi, sa, iznad, ispod, nasuprot, usprkos, među, prema ...
ADV	**prilozi** (eng. *adverbs*): riječi koje se najčešće dodaju glagolima ili pridjevima kako bi se dopunilo ili pobliže odredilo njihovo značenje	gdje, ovdje, ljeti, noću, kako, ovako, zašto, zato, koliko, ovoliko, toliko ...
AUX	**pomoćni glagoli** (eng. *auxiliary*): služe za tvorbu složenih glagolskih oblika te postoje samo dva pomoćna glagola – biti i htjeti. Oblici pomoćnih glagola biti i htjeti pomažu u tvorbi prošloga i budućega glagolskoga vremena.	sam, si, je, smo, ste, su, nisam, nije, hoću, ću, ćeš, će, hoćete, neću, nećeš, ...
CCONJ	**nezavisni veznici** (eng. *coordinating conjunction*): veznici koji povezuju nezavisno složene rečenice	i, pa, te, ni, niti, ili, samo, dakle, stoga, zato, a, ali, no, nego, već, ...
DET	**determinatori** (eng. *determiners*): svi elementi imenske skupine koji prethode upravnoj imenici skupine. Razlikuju se imenički, priložni i pridjevski determinatori.	Kakva sramota! Koliku važnost samo pridavaju sebi! Toliki su ih napustili!
INTJ	**usklici** (eng. *interjection*): nepromjenjive riječi kojima se izražavaju različiti osjećaji, pozdravlja, oponašaju zvukovi iz prirode i komunicira sa životinjama	ah, aj, eh, oh, bravo, ajme, au, halo, cmok, njam-njam, ha-ha, bok, zbogom ...
NOUN	**imenice** (eng. *nouns*): promjenjive riječi kojima se označuju bića, stvari i pojave	dječak, pas, klavir, računalo, torba, lopta, ogledalo, kiša, snijeg, vrijeme, ...
NUM	**brojevi** (eng. *numbers*): riječi kojima označujemo koliko čega ima ili koje je što po redu	jedan, jednom, dvije, dvjema, treći, osamnaestom, dvojica, petorica, ...
PART	**čestice** (eng. *participles*): nepromjenjive riječi koje služe za oblikovanje ili preoblikovanje rečeničnoga sadržaja. Česticama se iskazuje govornikovo stajalište o onome o čemu se govori; upotrebljavaju se u oblikovanju neke tvrdnje.	bar, baš, čak, da, evo, eto, eno, kao, li, ma, ne, zar, ...
PRON	**zamjenice** (eng. *pronouns*): promjenjive riječi kojima se zamjenjuju druge riječi, najčešće imenice	ja, ti, on, ona, ono, ovaj, onaj, oni, naš, vaš, ničiji, tkogod, štogod, ...
PROPN	**vlastite imenice** (eng. *proper nouns*): imenice koje služe kao ime svakom pojedinom čovjeku, životinji, zemljopisnom pojmu i sl.	Marko, Maja, Luna, Lucky, Alpe, Dunav, Jadran, Microsoft, Tesla, ...
PUNCT	**interpunkcija** (eng. *interpunction*): skup znakova kojima se u tekstu odjeljuju rečenice i rečenični dijelovi	točka (.), upitnik (?), uskličnik (!), zarez (,), točka sa zarezom (;), tri točke (...), zagrade (), ...
SCONJ	**zavisni veznici** (eng. *subordinating conjunction*): veznici koji povezuju zavisno složene rečenice	da, te, kako, gdje, kamo, kuda, kad, otkad, kako, neka, ako, iako, makar, koliko god, ...
SYM	**simboli** (eng. *symbols*): svaki znak kojemu je društveno ili kulturalno pripisano neko značenje	%, $, &, @, ☺, ☻, ♥, ♣, ♠, ...
VERB	**glagoli** (eng. *verbs*): promjenjive riječi kojima se izriče radnja, stanje i zbivanje	pjevati, pisati, slušati, stajati, gledam, gledaše, upitaše, saznah, kiši, grmi, ...
X	**nepoznate riječi**	fjfjeaf, aaaaa, ... miro

Tablica 2 Popis POS oznaka korištenog korpusa sa pripadajućim vrstama riječi (i primjerima) u hrvatskom jeziku