UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Interdisciplinarni študij računalništvo in matematika – 2. stopnja

Miha Štravs

AVTOMATSKA GRADNJA KORPUSA IN EKSTRAKCIJA RELACIJ V SLOVENŠČINI

Magistrsko delo

Mentor: doc. dr. Slavko Žitnik

Ljubljana, 2022

Zahvala

Neobvezno. Zahvaljujem se ...

Kazalo

Program dela

Mentor naj napiše program dela skupaj z osnovno literaturo. Na literaturo se lahko sklicuje kot [11].

Osnovna literatura

Literatura mora biti tukaj posebej samostojno navedena (po pomembnosti) in ne le citirana. V tem razdelku literature ne oštevilčimo po svoje, ampak uporabljamo okolje itemize in ukaz plancite, saj je celotna literatura oštevilčena na koncu.

[11] L. P. Lebedev in M. J. Cloud, *Introduction to Mathematical Elasticity*, World Scientific, Singapur, 2009

Podpis mentorja:

Avtomatska gradnja korpusa in ekstrakcija relacij v slovenščini

Povzetek

Iskanje relacij med entitetami v besedilu je področje obdelave naravnega jezika. Pri iskanju relacij želimo v stavku: "Ljubljana je glavno mesto Slovenije"odkriti, da med entitetama Ljubljana in Slovenija nastopa relacija glavno mesto.

V zaključnem delu smo najprej naredili pregled metod za učenje modelov za napovedovanje relacij. Nato smo si izbrali tri metode z različnimi pristopi za učenje modelov, ki napovedujejo relacije. Metodo s povratno nevronsko mrežo z dolgim kratkoročnim spominom, metodo z vložitvami BERT in metodo RECON, ki uporabi grafovsko nevronsko mrežo s pozornostjo. Za učenje modelov smo uporabili slovenski korpus, ki smo ga polavtomatsko generirali iz besedil slovenske Wikipedije. Naučene modele smo nato testirali na testnem korpusu besedil slovenske Wikipedije in testnem korpusu člankov strani 24ur.com. Na testnem korpusu slovenske Wikipedije so vse tri metode dosegle visoke priklice in točnosti, najbolje pa se je odrezala metoda RECON. Veliko slabše rezultate pa so dosegle na testni množici člankov 24ur.com, kjer se je še najbolje izkazala metoda z vložitvami BERT, ko je uporabila vložitve CroSloEngual.

English translation of the title

Abstract

Finding relations between entities in a text is an area of natural language processing. In the sentence: "Ljubljana is the capital of Slovenia"we want to find the relation capital between entities Ljubljana and Slovenia.

We first start with a review of the methods used for training models to predict relations. We then chose three methods with different approaches. The method with long short-term memory neural network, method which uses BERT encoder representations and method RECON which uses graph attention networks. To train the models, we used the Slovenian corpus which was generated semi-automatically from the text of the Slovenian Wikipedia. We test the models on a test corpus of Slovenian Wikipedia and the test corpus of articles on 24ur.com. All three methods achieved high recall and precision for the test corpus of the Slovenian Wikipedia and the RECON method performed best. Results were worse on the test set of 24ur.com articles, where the method which used BERT encoder representations CroSloEngual achieved the best results.

Ključne besede: ekstrakcija relacij, ekstrakcija informacij, globoko učenje, BERT, grafovske mreže pozornosti , LSTM

Keywords: relation extraction, information extraction, deep learning, BERT, graph attention networks , LSTM

1 Uvod

Na svetovnem spletu – internetu in v raznovrstnih arhivih obstaja ogromna količina tekstovnih, podatkovnih, zvočnih in grafičnih vsebin. Za njihovo preučevanje je priročno, če lahko uporabimo orodja, s katerimi omenjene vsebine pregledujemo, analiziramo ter iz njih pridobimo uporabna znanja in informacije. Področje, ki raziskuje orodja za pridobivanje informacij iz besedil, imenujemo tekstovno rudarjenje. Za pridobivanje informacij iz velike količine podatkov, ki so običajno bolj strukturirani od besedil, uporabljamo orodja za podatkovno rudarjenje. Orodja za tekstovno rudarjenje se torej od orodij za podatkovno rudarjenje razlikujejo v tem, da jih uporabimo na besedilih.

Da lahko uporabimo orodja za tekstovno rudarjenje, moramo besedilo najprej pretvoriti v zapis, ki nam bo podal čim več informacij o besedilu. Najpreprostejši način predstavitve besedila bi bil z nizom, vendar samo iz zaporedja znakov računalnik težko razbere strukturo in vsebino. Boljši zapis bi vseboval seznam osnovnih gradnikov besedila (besede, ločila) ter pomen in vlogo vsakega gradnika v besedilu. Takšen zapis lahko pridobimo s pomočjo orodij za procesiranje naravnega jezika. Pri tem besedilo najprej razdelimo na stavke, ki jih nato razdelimo na pojavnice s tokenizatorjem. Pojavnice so osnovni gradniki v besedilu (besede, ločila). Vsaki pojavnici nato z označevalnikov besednih vrst dodelimo še besedno vrsto, da določimo njeno vlogo v stavku. Na podlagi besedne vrste pa lahko določimo še lemo ali koren pojavnice. Z lemo ali korenom besede lažje poiščemo besede, ki so enake tudi če so zapisane v drugačnem sklonu ali številu. Za določanje vsebine pojavnic uporabimo vektorske vložitve besed, ki preslikajo besedo v vektor. Podobnost pomena dveh besed je sorazmerna s kosinusno razdaljo vektorjev vektorske vložitve, ki pripadata besedama. Na primer, vektorja besed *avto* in *tovornjak* bosta imela manjšo kosinusno razdaljo kot vektorja besed *maček* in *pesek*. Na ta način torej dobimo več podatkov o besedilu, poznamo osnovne gradnike, njihove vloge v stavkih in lahko hitro ugotovimo, če imata dva gradnika enako osnovno obliko ali podoben pomen.

Poleg strukture je za razumevanje besedila pomembna tudi vsebina. Za iskanje informacij o vsebini uporabimo metode ekstrakcije informacij. Cilj metod ekstrakcije informacij je povzeti uporabniku pomemben del vsebine iz besedila. Na primer katere osebe so omenjene v besedilu, čas in lokacija dogodkov v besedilu in če se besedila nanašajo na katero drugo znano besedilo.

Tri glavne naloge ekstrakcije informacij so:

- Razpoznavanje imenskih entitet, kjer se entitete nanašajo na stvari omenjene v besedilu.
- Odkrivanje koreferenčnosti (iskanje ponovne omembe entitete).
- Ekstrakcija povezav (iskanje povezav med entitetama).

Pri razpoznavanju imenskih entitet želimo v besedilu poiskati pojavnice, ki omenijo entiteto. Entitete se nanašajo na predmete v svetu, ki ga besedilo opisuje. Entitete nato razvrstimo v naprej določene skupine. Ponavadi entitete delimo na

osebe, lokacije, organizacije ter ostale entitete. Odkrivanje koreferenčnosti je proces, s katerim poiščemo, katere omembe v besedilu se nanašajo na isto entiteto. Pri ekstrakciji povezav pa želimo odkriti povezave med entitetami v besedilu.

Ponazorimo primer uporabe metode ekstrakcije informacij na naslednjih dveh stavkih:

Ljubljana je glavno mesto Slovenije.

V njej se nahaja Fakulteta za računalništvo in informatiko.

Pri razpoznavanju imenskih entitet bi v zgornjih dveh stavkih našli entitete *Ljubljana*, *Slovenije* in *Fakulteta za računalništvo in informatiko*. Prvi dve entiteti sta lokacija, zadnja pa je organizacija. Beseda *njej* v drugem stavku se nanaša na entiteto *Ljubljana* v prvem stavku, kar ugotovimo pri odkrivanju koreferenčnosti. Pri ekstrakciji povezav pa najdemo dve povezavi. Prvi stavek nam pove, da je *Ljubljana* **glavno mesto** *Slovenije*. Drugi stavek pa vsebuje povezavo **se nahaja** med *Fakulteta za računalništvo in informatiko* in besedo *njej*, ki se nanaša na entiteto *Ljubljana* iz prvega stavka.

Na koncu se podatki zapišejo v relacijsko podatkovno bazo, ki vsebuje vse entitete in povezave odkrite v besedilu. Tako torej iz nestrukturiranega besedila pridobimo strukturirane podatke, ki so veliko primernejši za tekstovno rudarjenje, analizo in izpeljavo informacij o vsebini besedila.

V tej nalogi se bomo osredotočili na problem ekstrakcije povezav za slovenski jezik. Različne metode za ekstrakcijo povezav so že opisane v velikem naboru člankov, vendar pa se ti članki večinoma osredotočajo na bolj uporabljene jezike kot sta na primer angleščina in kitajščina.

V drugem poglavju naloge so predstavljeni različni pristopi reševanja problema ekstrakcije relacij. V tretjem poglavju predstavimo metode in korpuse iz člankov, ki smo jih uporabili pri iskanju povezav entitet. Četrto poglavje opisuje postopek pridobitve korpusa, peto poglavje pa postopek učenja metod in rezultate naših testiranj. Temu sledi šesto poglavje z diskusijo, kjer diskutiramo rezultate naše metode. Na koncu povzamemo ugotovitve, rezultate ter možne nadgradnje.

2 Pregled področja

Poznamo več metod napovedovanja relacij med entitetami. Delimo jih na metode nadzorovanega učenja (angl. Supervised learning), ki zahtevajo učne podatke z že označenimi relacijami, in metode nenadzorovanega učenja (angl. Unsupervised learning), ki ne zahtevajo označenih učnih podatkov. Posebnost problema iskanja relacij je možnost pol-avtomatske izdelave označenih učnih podatkov. Ta nam omogoča gradnjo označenega korpusa, ki ga lahko potem uporabimo pri metodah nadzorovanega učenja. Tako se lahko izognemo ročnemu označevanju korpusa, ki je časovno potratno. V literaturi tak pristop velikokrat omenijo kot delno nadzorovan (angl. Distant supervision), medtem ko nadzorovani pristopi uporabijo ročno označen korpus. V tem poglavju predstavimo pogoste pristope metod nadzorovanega in nenadzorovanega učenja ter pol-avtomatsko gradnjo učnih podatkov za delno nadzorovan pristop.

2.1 Nadzorovano učenje

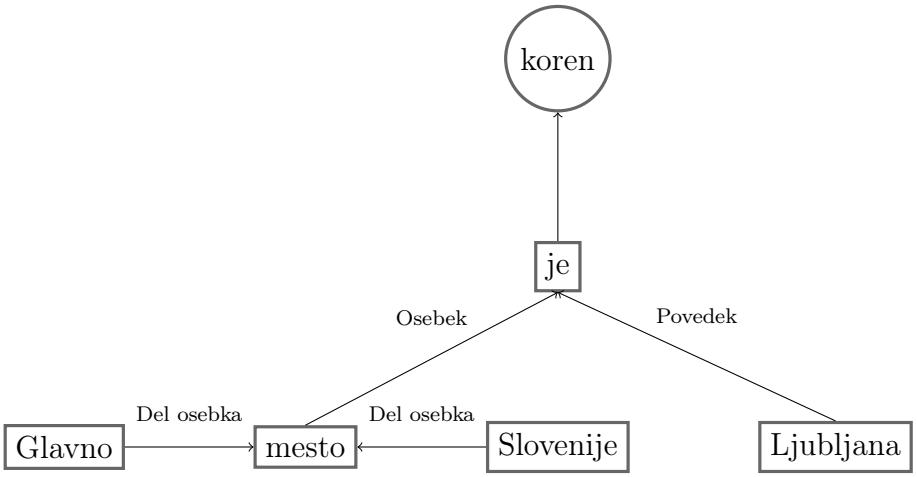
Pristopi nadzorovanega učenja se osredotočijo na problem iskanja relacij znotraj stavka. Za te metode potrebujemo označene podatke z vnaprej določenimi relacijami. Med določene tipe relacij dodamo še prazno relacijo. Če med parom entitet znotraj povedi ni relacije, rečemo, da ima prazno relacijo. Problem definiramo kot multi razredno klasifikacijo, torej lahko vsakemu paru entitet določimo tudi več kot eno relacijo. Metode za iskanje relacij s pomočjo nadzorovanega učenja uporabijo enega izmed treh pristopov. Na začetku se je za iskanje relacij uporabljalo jedrne metode (angl. Kernel methods) in metode, ki uporabijo značilke (angl. attribute-based methods). V zadnjih časih pa se raziskave zaradi boljših rezultatov vedno bolj pomikajo proti globokemu učenju z uporabo nevronskih mrež.

2.1.1 Metode, ki temeljijo na uporabi značilk

Za vsak primer relacije se najprej naredi značilke, na podlagi katerih bomo nato klasificirali nove relacije. Kot so pokazali v [29] in [15], je bila za klasifikacijo relacij z uporabo značilk zelo uspešna metoda podpornih vektorjev (angl. support vector machine SVM). Spodaj bomo pokazali, katere značilke so uporabili v [10] in primer teh značilk na stavku.

Glavno mesto Slovenije je Ljubljana.

- **Pojavnice** (angl. tokens) entitet in pojavnice med entitetama. Pojavnica prve entitete v primeru je *Ljubljana*, druge entitete je *Slovenije*, med entitetama pa se nahaja pojavnica *je*.
- **Tipi entitet**, ki nastopajo v relaciji, entiteti *Ljubljana* in *Sloveniji* sta obe lokaciji.
- **Tip omembe entitete**, ki je lahko nominalen, poimenski ali z uporabo zaimka. V primeru sta Ljubljana in Slovenija omenjeni z imenom.



Slika 1: Drevo skladenjske razčlenitve stavka.

- **Podatki o pojavnicah med entitetama**, sem lahko spada število pojavnic med omembama entitet in, če katera izmed pojavnic omeni katero drugo entiteto. V našem primeru se med entitetama *Ljubljana* in *Slovenije* nahaja ena pojavnica in ta ne omeni drugih entitet.

- **Skladenjski razčlenjevalnik** zgradi drevo iz besed v povedi. Drevo je zgrajeno tako, da kažejo povedki vsakega stavka na koren drevesa. Drevo nato razdeli stavek še na besede osebka in predmeta. Če znotraj stavka nastopi kakšen odvisni stavek se tega razčleni po principu rekurzije. Tako dobimo drevo, ki nam razčleni stavke v povedi glede na njihovo odvisnost. Izkaže se, da nam pri določanju relacije pomaga podatek o delu stavka (povedek, predmet, odvisni stavek...) in odvisnost stavka, v katerem nastopa entiteta. Primer preproste razčlenitve stavka je prikazan na sliki 1.

2.1.2 Jedrne metode

Pri metodah, ki uporabijo značilke, je uspešnost v veliki meri odvisna od izbire značilk. Izbiranju dobrih značilk se lahko izognemo z uporabo jedrnih metod. Za klasifikacijo se zopet uporabi metoda podpornih vektorjev, za izračun podobnosti med predstavitvijo dveh relacij pa uporabimo jedrne funkcije. Jedrne metode delimo na tiste, ki predstavijo relacije z zaporedjem, in tiste, ki predstavijo relacije z razčlenitvenim drevesom.

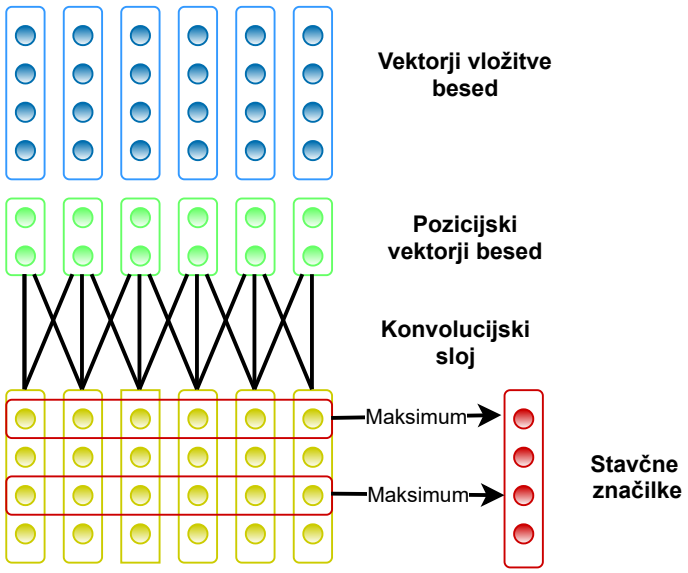
1. **Predstavitev z zaporedjem**
   Relacijo predstavimo kot zaporedje in uporabimo jedrno funkcijo, ki izračuna podobnost dveh relacij tako, da poišče skupna podzaporedja. Ta pristop je bil prvič uporabljen v [2]. Zaporedje predstavljajo vse besede v stavku. Zraven ustvarimo še tri dodatna zaporedja. Eno vsebuje besedne vrste (glagol, pridevnik, prislov, ...), drugo vsebuje tip entitete, če jo besede omenijo, in tretje vsebuje dodatne informacije o besednih vrstah (npr. prislov, ki je lahko krajevni, časovni, lastnostni ali prislovni). Tako torej dobimo štiri zaporedja besed stavka. Iz učne množice nato s pomočjo jedrnih funkcij poiščemo pod-

zaporedja, ki dobro napovejo tip relacije. Ta podzaporedja jedrna funkcija preslika v vektor, ki se uporabi pri določanju relacije s pomočjo SVM metode.

2. **Predstavitev z razčlenitvenim drevesom**
   Za predstavitev relacije z metodo, opisano v [3] najprej določimo besedne vrste in tipe entitet, ki nastopajo v relaciji. Nato ustvarimo razčlenitveno drevo stavka in izberemo najmanjšo pod drevo, ki še vsebuje obe entiteti relacije. Z izbiro pod drevesa zmanjšamo možnost šuma v podatkih. Podobno kot prej s pomočjo jedrnih funkcij ustvarimo vektorje, ki vsebujejo pojavitve značilnih podzaporedji v drevesu. Z uporabo metode SVM pa nato model naučimo klasifikacije relacij iz učnih podatkov.

2.1.3 Metode globokega učenja

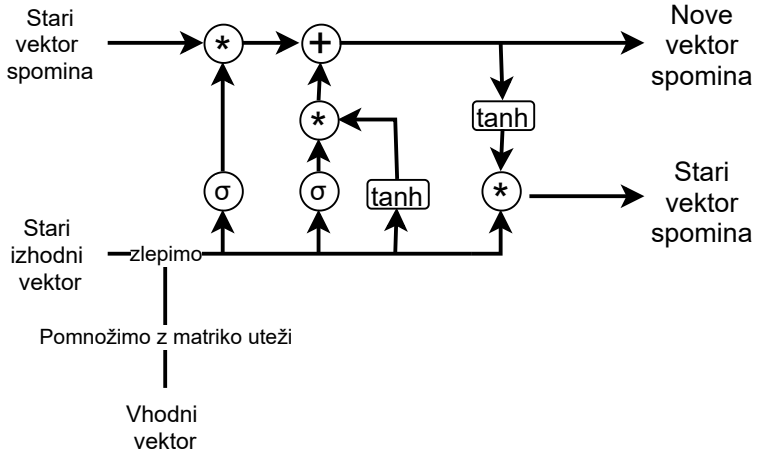
Prejšnji dve skupini metod potrebujeta zahtevno preprocesiranje besedila za iskanje kvalitetnih značilk ali pa za iskanje dobrih jedrnih funkcij. Prednost metod globokega učenja je, da zahtevnega preprocesiranja ne potrebujemo. Metode globokega učenja uporabijo veliko različnih pristopov. V nadaljevanju bomo najprej na kratko opisali pristop s konvolucijsko nevronsko mrežo in pristop z nevronsko mrežo s povratno zanko. Nato pa bomo opisali še dva sodobnejša pristopa s transformerjem BERT ter grafovskimi nevronskimi mrežami.



Slika 2: Konvolucijska nevronska mreža za pridobivanje stavčnih značilk, kjer se za izračun vmesnih rumenih vektorjev uporabi pojavnica skupaj z eno desno in eno levo pojavnico.

Konvolucijska nevronska mreža (angl. convolutional network) je mreža, ki

ima vsaj en konvolucijski sloj. Konvolucijski sloj izvede operacijo konvolucije na vhodnem zaporedju. Sledi agregacijski sloj (angl. pooling layer), ki ga uporabimo, da izločimo manj pomembne vrednosti izhoda konvolucijskega sloja. Iz izhoda agregacijskega sloja pa izpeljemo klasifikacijo s pomočjo polno-povezane mreže. Konvolucijske mreže se velikokrat uporabi za iskanje predmetov (pes, letalo, ...) na slikah. Namreč konvolucijski sloj je narejen tako, da pogleda vsak del slike in ugotovi, če se tam pojavi vzorec, ki bi lahko predstavljal objekt. Modeli nevronske mreže so torej bolj preprosti in točni, saj iščejo vzorce na delih slike, in ne le če je celotna slika podobna tistim v učnih primerih. Entiteti relacije se lahko pojavita kjerkoli v stavku. Morda je del stavka, ki je na levi ali desni strani entitet, popolnoma nepomemben za določitev entitet. Zato se konvolucijski sloj v mreži izkaže kot dober za iskanje relacij. Prvo metodo s konvolucijsko mrežo so predstavili v članku [27]. Metoda sprejme vektorske vložitve besed povedi kot vhod. Vsakemu vektorju se doda še pozicijski vektor, ki pove oddaljenost besede od obeh entitet, ki nastopata v relaciji. Za detekcijo relacij se nato uporabi besedne značilke, ki so vektorji samostalnikov, in pa stavčne značilke, ki jih pridobimo, ko pošljemo vse vektorje v konvolucijsko mrežo, ki je predstavljena v sliki 2. S stavčnimi značilkami je okolica entitet bolje opisana in zato metoda doseže boljše rezultate kot samo z uporabo besednih značilk.



Slika 3: Struktura LSTM celice.

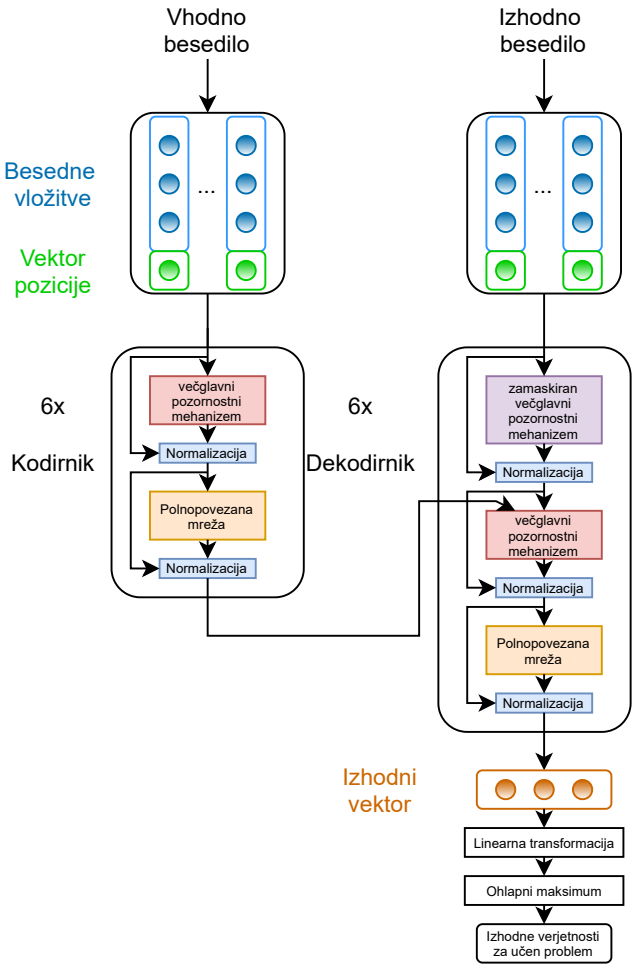
Nevronska mreža s povratno zanko (angl. recurrent neural network) je še en primer mreže, ki je primerna za analizo besedil. Standardna nevronska mreža računa izhod le na podlagi vhodnih podatkov. Vhodne vrednosti pretvori v izhodni vektor s pomočjo matrike uteži. Mreža s povratno zanko deluje podobno, s to razliko, da za vhod uporabi vhodni vektor in izhodni vektor prejšnjega koraka. Tak model torej uporabi znanje, ki je bilo pridobljeno na prejšnjem koraku, za razumevanje novih vhodnih podatkov. Podobno deluje tudi človek, ki prebira besedilo. Vsakič prebere naslednjo besedo in ustvari njeno interpretacijo glede na besede, ki jih je že prebral. Iz te primerjave bi lahko sklepali, da mreže s povratno zanko

zajamejo vsebino besedila bolje kot navadne mreže. Za napovedovanje relacij se večinoma uporabi povratno nevronsko mrežo z dolgim kratkoročnim spominom (angl. long short-term memory LSTM), ki je prikazana na sliki 3. Mreža s povratno zanko v osnovi uporabi samo prejšnji izhod, torej se gleda samo pridobljeno informacijo prejšnje besede. Mreža z dolgim kratkoročnim spominom pa zraven hrani še vektor spomina, ki veliko bolje povzame informacije iz več predhodnih besed. Primer modela, ki uporabi dvosmerno nevronsko mrežo z dolgim kratkoročnim spominom, je opisan v članku [28]. Za vhod se uporabijo vektorske vložitve besed stavka. Vektorjem besed se pripne še njihove pozicijske vektorje. Vektorje dajemo v mrežo LSTM v takem vrstnem redu kot se pojavijo v stavku, potem pa to ponovimo še v obratnem vrstnem redu. Za vsak vhodni vektor besede dobimo dva izhodna vektorja (po enega za vsak vrstni red). Ta dva vektorja se seštejeta v vektor vektorske predstavitve besede. Iz pridobljenih vektorskih predstavitev besed želimo pridobiti vektorsko predstavitev stavka. To storimo tako, da za vsako pozicijo vektorjev besednih predstavitev izberemo največjo. Na pridobljenem vektorju nato za klasifikacijo relacije uporabimo polno-povezano mrežo.

Uporaba vložitve BERT (predstavitve dvosmernega kodirnika iz transformeja) (angl. Bidirectional Encoder Representations from Transformers) je pogost pristop za odkrivanje relacij v sodobni literaturi. Učenje s prenosom znanja (angl. transfer learning) se veliko uporablja na področju procesiranja naravnega jezika. Pri takem učenju uporabimo mrežo, ki je bila že naučena za nek splošen problem (npr. iskanje skrite besede v stavku), in jo potem še dodatno naučimo za bolj specifičen problem. Učenje s prenosom znanja je priročno, ker dosega dobre rezultate z manj učnimi podatki kot bi jih potrebovali, če bi mrežo učili od začetka.

Struktura BERT uporabi transformer. Transformerji uporabljajo pozornost (angl. attention), ki nam pove katere, besede v stavku so pomembne pri reševanju problema. Če zakrijemo besedo v stavku in želimo ugotoviti katero besedo smo zakrili, nam pozornost pove, katere ostale besede v stavku nam najbolje povedo katero besedo smo zakrili. Za razliko od mreže LSTM je transformer brez smeri in obravnava celoten stavek hkrati. Prednost tega je, da imamo za vsako besedo v stavku izračunano pozornost za vse ostale besede v stavku. Poleg tega pa za razliko od mrež, ki smo jih prej opisali, oddaljenost besed v stavku ne vpliva na izračun konteksta besede.

Za učenje BERT so uporabili transformer iz članka [24]. Ta transformer uporabi pozornost in polno povezane sloje mreže za kodirnik in dekodirnik. Kodirnik sestoji iz šestih identičnih slojev kjer je vsak sloj sestavljen iz dveh podslojev. Prvi je večglavni mehanizem pozornosti (angl. multi-head self-attention mechanism), drugi pa polno povezana mreža. Za vsakim podslojem se izvede še normalizacija. Normalizacijo izvedemo tako, da po komponentah seštejemo vhodni in izhodni vektor podsloja in vsoto normaliziramo, da ima velikost 1. Dekodirnik ima tudi 6 slojev, ki imajo poleg dveh podslojev v kodirniku še en dodatni podsloj za pozornost na začetku. Ta podsloj je spremenjena različica večglavnega pozornostnega mehanizma, ki išče le pozornosti od besed, ki se pojavijo prej. Ta podsloj in dejstvo, da so vhodne vložitve zamaknjene za eno pozicijo v desno, nam zagotovi, da se za napovedovanje besede uporabi le besede, ki nastopajo pred njo. Shema transformerja je prikazana



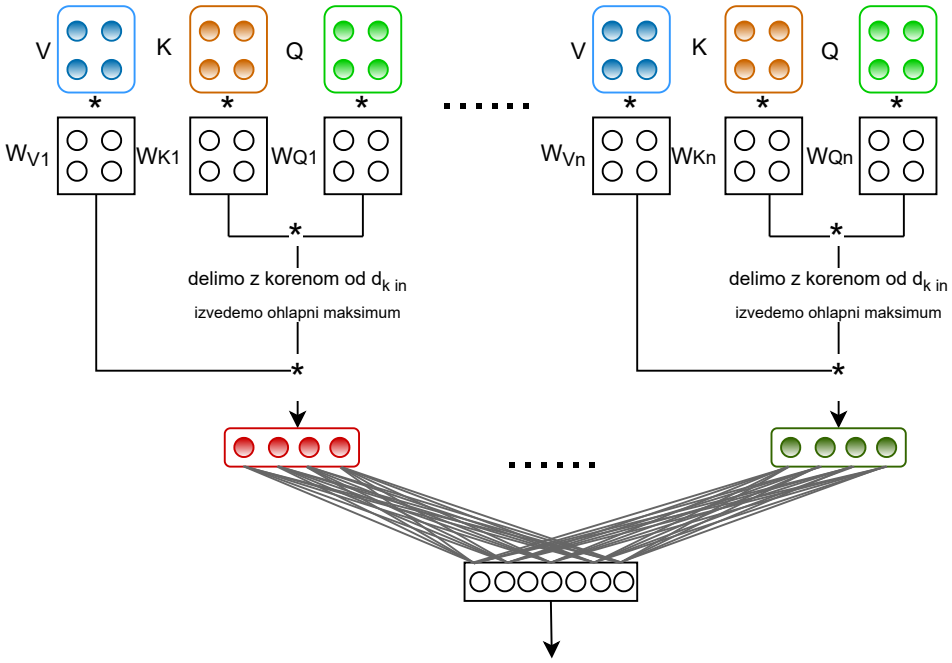
Slika 4: Struktura BERT transformerja.

na sliki 4.

Funkcija pozornosti v transformerju je funkcija, ki prejme vhod Q, ključe K in vrednosti V, kjer so vse tri vrednosti predstavljene z matriko. Vsaka vrstica matrike ključev K pripada vrstici matrike vrednosti V. Izhod je utežena vsota vrednosti V, kjer so uteži vrednosti, ki jih izračunamo tako, da izračunamo ujemanje pripadajočih ključev K in vhoda Q. Pozornost izračunamo na naslednji način:

$$Pozornost(Q, K, V) = softmax(\frac{W_Q Q (W_K K)^T}{\sqrt{d_k}}) W_V V$$

Kjer so W_Q, W_K in W_V matrike uteži, ki predstavljajo polno povezane mreže, ki jih uporabimo na vhodnih matrikah preden začnemo računati pozornost. Ujemanje vhoda Q in ključev K dobimo z množenjem matrik. Rezultat normaliziramo s korenom d_k, ki vsebuje število ključev v K, in z ohlapnim maksimumom. Normalizirana ujemanja nato pomnožimo z V. Rezultat funkcije *Pozornost* je vektor o, ki ga pošljemo še skozi polno povezano mrežo tako, da ga pomnožimo z utežmi W. Opisani postopek izračuna eno glavo pozornost transformer pa uporabi večglavo pozornost.



Slika 5: Struktura večglavega pozornostnega mehanizma.

Večglava pozornost izračuna funkcijo pozornost za več različnih W_{Q_i}, W_{K_i} in W_{V_i} ter na koncu zlepi vse izhodne rezultate o_i in jih nato pomnoži z matriko uteži W. Večglava pozornost zagotovi bolj stabilno učenje. Struktura je prikazana na sliki 5

Mrežo BERT se uči na dveh splošnih problemih. Pri prvem problemu zakrijemo besedo v stavku in nato transformer poskusi ugotoviti katero besedo smo skrili. Tako se transformer uči iskanja pozornosti med besedami v stavku. Drugi problem pa zahteva od BERT, da pri podanem stavku iz nekega besedila ugotovi, kateri je naslednji stavek v besedilu. S tem pa se transformer uči iskanja pozornosti med stavki.

Primer uporabe vložitve BERT za ekstrakcijo relacij predstavimo v podpoglavju 3.2.

Grafovske mreže pozornosti (angl. graph attention network (GAT)) so trenutno najnovejše odkritje pri iskanju relacij med entitetam v besedilu. Grafovske mreže se uporabi, ko vhodne podatke problema boljše definiramo v obliki grafa kot pa v obliki matrike. Prejšnji pristopi so stavek predstavili kot matriko besednih vložitev, kjer sosednja stolpca predstavljata dve besedni vložitvi besed, ki nastopata kot sosedi v stavku. Z grafom pa lahko že na začetku določimo katere besede so med seboj bolj povezane ne glede na vrstni red besed. Mreži lahko rečemo, da je grafovska mreža pozornosti, če ima vsaj en sloj z grafom pozornosti [25]. Tak sloj sprejme n vozlišč h_i, kjer vsako vozlišče predstavimo z vektorjem z m lastnostmi. Iz vhodnih vozlišč izračunamo za vsako vozlišče izhodni vektor h'_i dimenzije m'. To pa storimo s pomočjo matrike uteži W, ki ima dimenzije $m * m'$, in pozornosti a_{ij}, ki nam pove pomembnost vrednosti v vhodnem vektorju h_j za izhodni vektor h'_i.

Za izračun pozornosti uporabimo funkcijo samo pozornosti, ki preslika Wh_i in Wh_j v nenormalizirano pozornost. Za funkcijo uporabimo polno povezano mrežo, ki za vhod vzame zlepljena vektorja $Wh_i||Wh_j$, ju pomnoži z matriko uteži A in uporabi $LeakyReLU$ za aktivacijsko funkcijo. Pozornosti nato normaliziramo z ohlapnim maksimumom. Celoten postopek izračuna pozornosti je zapisan v spodnji enačbi.

$$a_{ij} = \frac{exp(LeakyReLU(A[Wh_i||Wh_j]))}{\sum_{k \in N_i} exp(LeakyReLU(A[Wh_i||Wh_k]))}$$

V zgornji enačbi je N_i množica vozlišč, ki so sosednja s h_i in exp je eksponentna funkcija, ki jo uporabimo pri normalizaciji z ohlapnim maksimumom. Vektorje novih vozlišč izračunamo na naslednji način:

$$h'_i = \sigma(\sum_{j \in N_i} a_{ij} W_2 h_j)$$

Kjer je σ sigmoidna funkcija in W_2 matrika uteži polno povezane mreže.

2.2 Polavtomatska generacija korpusa

Globoko učenje zahteva velike količine učnih podatkov, ki pa nam žal velikokrat niso dostopni. Ideja delno nadzorovanega učenja (angl. distant supervision) je, da naredimo metodo, ki bo avtomatsko izdelala učne podatke, ki bodo vsebovali nekaj šuma zaradi napak te metode. S pomočjo nadzorovanega učenja nato na izdelanih podatkih naučimo model za napovedovanje relacij.

2.2.1 Osnovni pristop gradnje podatkov

Ekstrakcijo relacij predstavimo kot multi klasifikacijski problem. Torej za vsak par entitet, ki nastopajo v istem stavku, določimo tip relacije. Če relacije ni, jo označimo kot prazno relacijo. Za avtomatsko označevanje entitet in njihovih relacij v nekem prostem besedilu najprej uporabimo standardna orodja za obdelavo naravnega jezika. Sem spadajo označevalec z razčlenilnim drevesom, označevalec besedilnih vrst in označevalnik imenskih entitet. V drugem koraku uporabimo bazo znanja (angl. knowledge base). Gre za baze, ki vsebujejo ročno vnesene entitete ter relacije med njimi. Primer takšnih baz sta WikiData in FreeBase. Entitetam, ki smo jih označili v besedilu, želimo določiti entiteto iz baze znanja, s katero se ujema. Proces ujemanja entitet (angl. entity linking) je kompleksen, ker se lahko več različnih entitet napiše na enak ali podoben način. Primer reševanja problema ujemanja entitet je opisan v [13], kjer so iskali ujemanja entitet na semantičnem omrežju BabelNet 1.1.1. Sedaj imamo v besedilu označene entitete in za nekatere izmed teh entitet tudi ujemanja v bazi znanja. Za vsak par entitet, ki nastopata v istem stavku in imamo za obe entiteti ujemanji, zberemo še par dodatnih podatkov za kasnejšo klasifikacijo. Primeri takih podatkov so:

- Zaporedje besed med entitetama.
- Besedne vrste teh besed.

- Katera entiteta v relaciji nastopi prva v stavku.
- Dve besedi levo od prve entitete ter dve besedi desno od druge entitete. Za te besede shranimo še besedne vrste.
- Pot med entitetama v razčlenitvenem drevesu.

Nato za vsak tak par entitet v bazi znanja poiščemo, če med njima obstaja relacija. Če relacija obstaja, rečemo, da ta relacija nastopa tudi v stavku. Pri tem pa privzamemo, da če se v stavku pojavi par entitet, bo ta par imel vedno enako relacijo. Na primer, da imamo dva stavka: prvi je *Borut Pahor je predsednik Slovenije*, drugi pa *Borut Pahor se je včeraj udeležil prireditve, ki je potekala v Sloveniji*. V prvem stavku je omenjeno, da je Borut Pahor predsednik Slovenije. V drugem stavku pa je omenjen Borut Pahor in Slovenija, ne piše pa, da je Borut Pahor predsednik Slovenije. Ker označujemo na podlagi relacij, shranjenih v bazi znanja, bomo najprej v bazi poiskali entiteti *Borut Pahor* in *Slovenija*. Nato bomo izvedeli iz baze, da je Pahor predsednik Slovenije in bomo v obeh stavkih označili, da ta relacija nastopa med najdenima entitetama. Tako dobimo med učne podatke stavke, ki imajo označeno relacijo, ne da bi ta nastopala v stavku. Poleg tega nam postopek označi samo pozitivne primere relacij. Za pare entitet, med katerimi ni označenih relacij, ne vemo ali relacije obstajajo. Tak način pridobivanja podatkov pa se je kljub temu izkazal za dobrega. Ker je gradnja učnih podatkov avtomatska, lahko brez večjih problemov izdelamo dovolj veliko podatkov za metode globokega učenja. Te metode pa se lahko pri veliki količini učnih podatkov delno izognejo šumu.

2.2.2 Načini za zmanjševanje šuma v podatkih

V prejšnjem podpoglavju 2.2.1 smo omenili, da pri polavtomatski generaciji podatkov dobimo primere napačnih relacij in smo podali primer napačne relacije. Število napačnih relacij v pridobljenih podatkih pa je možno zmanjšati na več načinov:

- Spremenimo pomen iskanja relacij entitet. Do sedaj smo govorili o iskanju omemb relacij v tekstu (angl. mention level extraction). Lahko pa rahlo spremenimo definicijo iskanja relacij. Namesto, da gledamo ali se v stavku med dvema entitetama pojavi relacija, raje gledamo katere relacije se pojavijo v besedilu vsaj enkrat (angl. at least one models). Zanima nas torej, če lahko iz celotnega besedila povzamemo, da relacija obstaja. Ker se pari entitet v besedilu ponavljajo, povečamo možnost, da je vsaj en par tak, ki vsebuje omembo iskane relacije.
- **Tematski modeli** (angl. topic based models) izdelajo podatke posebej za vsako temo (na primer posebej za šport, politiko, ...).
- **Modeli korelacije vzorcev** (angl. Pattern correlations models) uporabijo seznam vzorcev, pri katerih je velika verjetnost, da do relacije ne pride. S tem lahko odstranimo veliko lažnih pozitivov.

2.3 Nenadzorovano učenje

Ekstrakcijo relacij lahko izvedemo tudi z metodami nenadzorovanega učenja. Nenadzorovano učenje ne zahteva učnih podatkov in ne potrebujemo vnaprej določenih tipov relacij. Namesto, da uporabimo označeno množico za učenje metode s katero potem napovedujemo relacije na drugih besedilih, uporabimo metodo nenadzorovanega učenja, da na neoznačenem besedilu sama razdeli relacije v skupine. Vsaka skupina predstavlja tip relacije. Pri klasifikaciji novih relacij metoda poišče skupino z najbolj podobnimi relacijami in označi relacijo s tipom skupine.

2.3.1 Gručenje

Za nenadzorovano učenje pri ekstrakciji relacij je najbolj uporabljen pristop gručenja. Osnovni primer metode za gručenje relacij je opisan v [7]. Metoda ima štiri korake.

1. V besedilu se najprej **označi imenske entitete**.

2. V naslednjem koraku se **določi pare entitet** in za vsak par **shrani kontekst relacije**. Rečemo, da sta entiteti v paru, če med njima nastopa največ N besed. Primer konteksta para entitet so besede, ki nastopajo med njima, in besede, ki vsebujejo entiteti. Pare razlikujemo po vrstnem redu. Isti par entitet, ki nastopa v obratnem vrstnem redu, je obravnavan kot drugačen.

3. Za vsak par prevedemo besede v vektor, ki predstavi relacijo. V [7] uporabijo vektorje TFIDF. Vektor TFIDF izračunajo s pomočjo TF, ki pove, kako pogosta je beseda v kontekstu za par, in IDF, ki je inverz od števila različnih parov entitet, v katerih se beseda pojavi. Število pojavitev besed za izračun TF od para entitet E1 in E2, uporabijo besede, ki se pojavijo med parom E1 in E2 in odštejejo pojavitve besed, ki se pojavijo med parom E1 in E2. S tem poizkusijo modelirati smer relacije.

4. Na vektorjih se nato uporabi hierarhično gručenje, da dobimo različne gruče relacij. Razdaljo med vektorji izračunamo s pomočjo kosinusne razdalje. Vsaka dobljena gruča predstavlja svojo relacijo.

3 Pregled metod

V tem poglavju smo predstavili metode, ki smo jih uporabili za ekstrakcijo relacij za slovenski jezik. Metode, ki smo jih predstavili, so bile že preizkušene za angleški jezik in so opisane v literaturi, mi smo pa preizkusili njihovo uspešnost za slovenski jezik.

Ker želimo uporabiti metode za globoko učenje, potrebujemo obsežen korpus, ki pa za slovenski jezik še ne obstaja. Ker bi ročno označevanje vzelo preveč časa, smo se odločili za polavtomatsko gradnjo korpusa. Za gradnjo podatkov se zgledujemo po članku, ki ga predstavimo v podpoglavju 3.1. Postopek, opisan v podpoglavju 3.1.1, nam zagotovi preprosto in točno označevanje entitet baze znanja WikiData v besedilih na Wikipedijinih straneh. Poleg tega si ogledamo še metodo za iskanje relacij, ki smo jo uporabili, da smo lahko primerjali rezultate našega korpusa, in rezultate za korpus v angleškem jeziku.

Nadaljujemo s podpoglavjem 3.2, kjer smo predstavili metodo, ki uporabi vložitve BERT. To metodo smo si izbrali, ker nas zanima, če nam bo učenje s prenosom znanja pomagalo pri doseganju boljših rezultatov. Poleg tega pa BERT temelji transformerju, ki uporabi pozornosti, ta pa se je v literaturi izkazala za zelo uspešno pri določanju relacij med entitetami.

Zadnja metoda, ki smo jo izbrali, je predstavljena v podpoglavju 3.3. Metoda uporabi grafovske mreže pozornosti, ki trenutno dosegajo najboljše rezultate pri našem problemu, in metoda poleg stavka, kjer relacija nastopa, uporabi še kontekst entitet, ki ga izračuna znotraj baze znanja. Poleg tega v članku [1], kjer metodo opišejo, za testiranje metode uporabijo generirani korpus iz podpoglavja 3.1.1, po katerem smo se zgledovali pri izdelavi našega korpusa. To nam poda še eno priložnost za primerjavo korpusov.

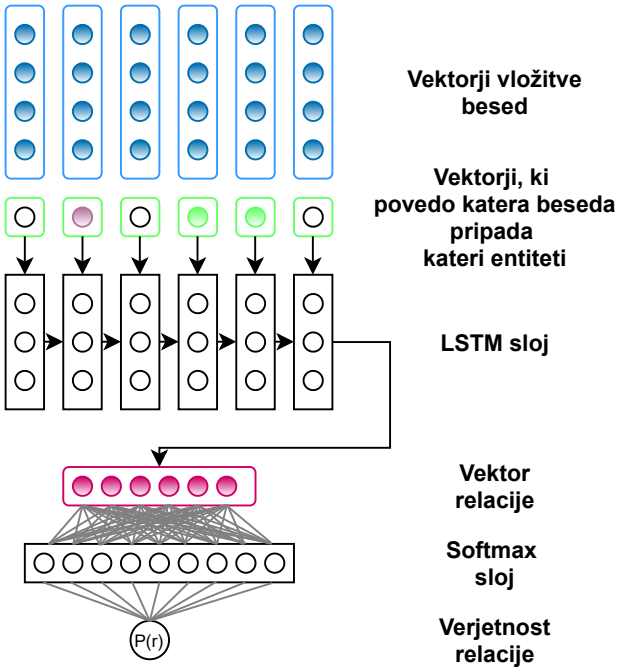
3.1 Izdelava učnih podatkov iz besedil Wikipedije s pomočjo baze znanja WikiData

WikiData je baza znanja, ki vsebuje splošno znanje v obliki entitet (Ljubljana:Q437, Slovenija:Q215, ...) in relacij med njimi (Glavno mesto:P36, ...). V tem podpoglavju bomo predstavili članek [20], v katerem so uporabili bazo WikiData za izdelavo podatkov iz celotnega korpusa angleških besedil Wikipedije, na kateri so s povratno nevronsko mrežo z dolgim kratkoročnim spominom naučili napovedovanti relacije.

3.1.1 Izdelava korpusa

Wikipedija in WikiData sta tesno povezani. Vsaka stran na Wikipediji predstavlja točno eno entiteto v bazi WikiData. Ta lastnost nam poda možnost za zelo enostaven in natančen proces ujemanja entitet. Če se v vsebini strani pojavi omemba entitete iz WikiData, je ta del besedila velikokrat notranja povezava na stran Wikipedije za to entiteto. V članku [20] so znotraj spletnih strani poiskali vse notranje povezave Wikipedije in ugotovili, kateri entiteti pripada spletna stran na povezavi. Dodatne entitete so poiskali s pomočjo označevalnika imenskih entitet Stanford CoreNLP. Za vsako tako entiteto so nato poiskali, če obstaja vnos v WikiData, in če je ta obstajal, so ga označili. Poleg tega pa so uporabili še orodje HeidelTime [21], s katerim so

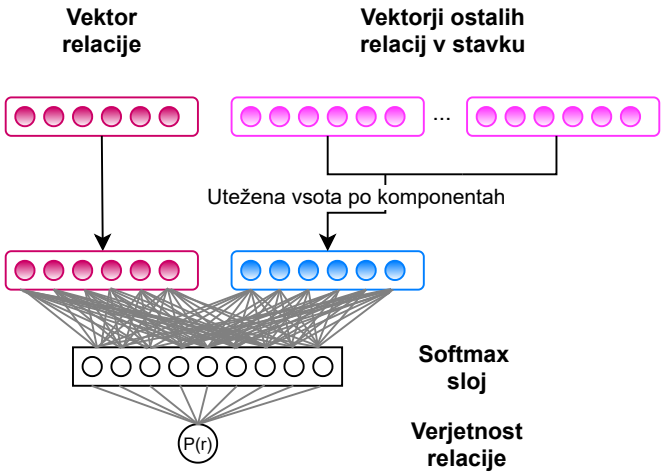
poiskali datume v besedilu, ki so jih prav tako označili kot entitete. Za vsak par entitet znotraj iste povedi so poiskali, če med njima obstaja relacija, ter jo označili. V primeru, da je za nek par obstajala več kot ena relacija, so ga zaradi dvoumnosti odstranili iz podatkov. Iz zgrajenih podatkov so nato vzeli 200 relacij in preverili točnost podatkov. Pravilno označenih je bilo 158 relacij.



Slika 6: Diagram modela za napovedovanje relacij.

3.1.2 Predpriprava podatkov

Za vsako relacijo uporabijo poved, v kateri se relacija nahaja. Vsako besedo najprej pretvorijo v besedni vektor s pomočjo vektorske vložitve GloVe [16]. Vsakemu besednemu vektorju se na koncu pripne še pozicijski vektor. Imamo tri pozicijske vektorje. Enega za besede, ki nastopajo v omembi prve entitete, drugega za omembo druge entitete in še zadnjega za preostale besede. Pozicijski vektorji so naključno inicializirani na začetku predpriprave podatkov in so dolžine d, kjer je d na začetku določen parameter. Za vsako relacijo dobijo seznam vektorjev. Ker želimo, da ima vsaka relacija po en vektor, ki ima enako dolžino za vsako relacijo, vstavljamo vektorje iz seznama v povratno nevronsko mrežo. Iz mreže dobimo vektor o, ki predstavlja relacijo.



Slika 7: Uporaba ostalih relacij znotraj stavka za napovedovanje relacije.

3.1.3 Metoda za napovedovanje relacij

V osnovi se za model uporabi povratno nevronsko mrežo z dolgim kratkoročnim spominom (LSTM). To omrežje ima samo en sloj z ohlapnim maksimumom (angl. softmax layer), ki je uporabljen za določanje tipa relacije iz vektorja o pridobljenega v prejšnjem koraku. Diagram celotnega modela je predstavljen na sliki 6.

Poleg uporabe osnovnega LSTM modela pa avtorji članka [20] postavijo hipotezo, da na iskano relacijo vplivajo tudi preostale relacije v isti povedi. Nekateri tipi relacij se pogosto pojavijo v istem stavku. Primer takih relacij sta *producent* in *direktor*. To pa zato, ker je velika možnost, da producent in direktor nastopata skupaj s filmom v istem stavku. Obstajajo pa tudi relacije, ki v stavku velikokrat nastopajo same. Primer take relacije je *rojstni kraj*.

To hipotezo se preizkusi na naslednji način. Znotraj stavka se vzame vse entitete in poišče vse možne preostale pare entitet. Zaradi omejitve računskih sposobnosti so se v članku omejili na največ šest dodatnih parov entitet. Za vsak par pripravimo vektor kot je opisano v podpoglavju 3.1.2. Ker niso vse ostale relacije v stavku enako pomembne za izračun relacije, uporabimo uteži pozornosti (angl. attention score weight). Pozornost nam pove pomembnost ostalih relacij pri napovedovanju trenutne relacije. Utež pozornosti relacije i pri napovedovanju relacije s izračunamo po formuli:

$$a_i = \frac{exp(o_i A o_s)}{\sum_{j=0}^{M} exp(o_j A o_s)}$$

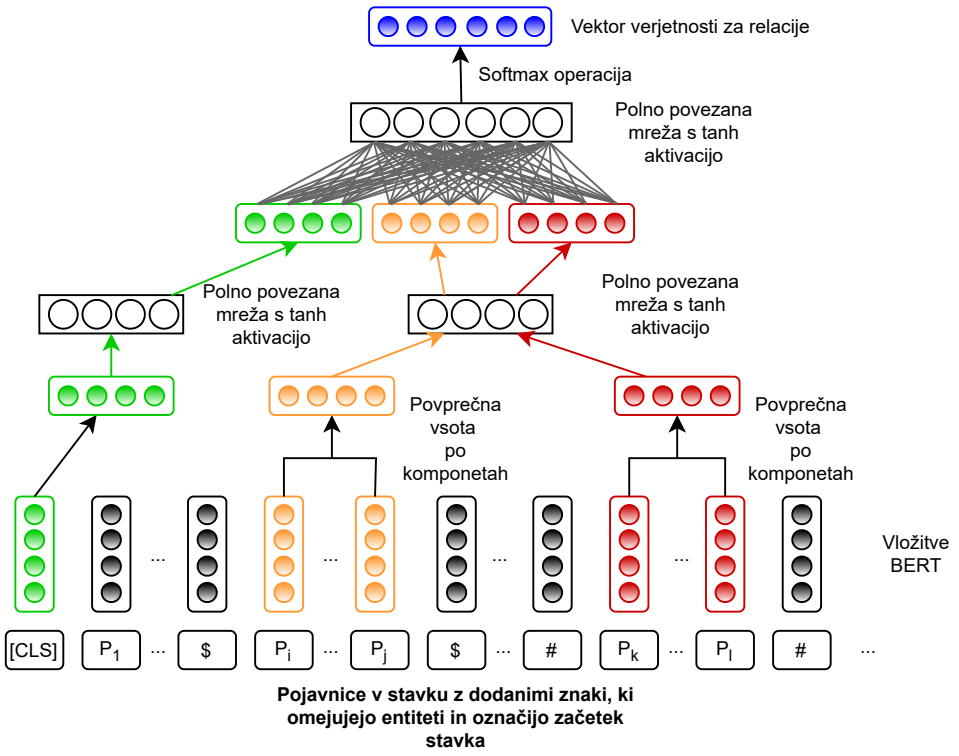
Kjer je A matrika uteži polno povezane mreže, ki jo uporabimo za računanje pozornosti, M število ostalih relacij v stavku, o_x vektor relacije x in exp eksponentna funkcija, ki jo uporabimo za normalizacijo uteži pozornosti z ohlapnim maksimumom. Vsak vektor ostalih relacij pomnožimo z njegovo utežjo pozornosti in jih seštejemo v en vektor. Dobljeni vektor pripnemo na obstoječi vektor relacije. To

potem pošljemo v prej opisano nevronsko mrežo LSTM. Postopek je ponazorjen v sliki 7. Izkaže se, da kontekst v obliki parov ostalih entitet pripomore k boljši točnosti modela.

3.2 Uporaba vložitev BERT za napovedovanje relacij

Z uporabo vektorskih vložitev je do neke mere možno nadomestiti veliko učno množico. Trenutno se vložitve BERT [5] izkazujejo za zelo učinkovite pri večini opravil za razumevanje naravnega jezika. V tem podpoglavju predstavimo metodo [26], ki najprej natrenira vložitve BERT in jih nato uporabi pri napovedovanju relacij.

3.2.1 BERT



Slika 8: Diagram metode, ki napoveduje relacije s pomočjo vložitev BERT.

Vložitve BERT uporabijo kodirnik večslojnega dvosmernega transformerja za predstavitev besed z vektorji. Podatki za učenje vložitev BERT so sestavljeni iz stavkov in parov stavkov, ki nastopajo skupaj v besedilu. Posameznim stavkom na začetek dodamo pojavnico [CLS], parom stavkom dodamo na začetek pojavnico [CLS] in pojavnico [SEP] med stavkoma. BERT se uči na dveh problemih. Pri

prvemu se zakrije eno pojavnico v stavku in poskusi ugotoviti katera pojavnica je bila zakrita glede na ostale pojavnice stavka. Drugi problem pa poskusi napovedati stavek, ki nastopa za podanem stavkom.

3.2.2 Napovedovanje relacij

Metoda za napovedovanje relacij sprejme stavek, ki vsebuje relacijo. Stavek se razdeli na pojavnice. V stavek se vrine pojavnico $ pred začetkom in po koncu omembe prve entitete in pojavnico # pred začetkom in po koncu omembe druge entitete. Na začetek pa se doda pojavnico [CLS]. Torej, če imamo stavek *Matevž je včeraj bil v operi* dobimo pojavnice *[CLS],* $*, Matevž,* $*, je, včeraj, bil, v,* #*, operi,* #*.*

Seznam pojavnic pošljemo v BERT kodirnik, da za vsako pojavnico dobimo vektor. Prvo entiteto predstavimo s povprečno vsoto vektorjev pojavnic, ki omenijo prvo entiteto. Podobno velja za drugo entiteto. Vrednosti v vektorju spremenimo v vrednost hiperbolične tangente, ki je aktivacijska funkcija. Nato oba vektorja pošljemo skozi polno povezan sloj mreže (jo pomnožimo z matriko uteži W in prištejemo vektor b). Poleg dveh vektorjev entitet pa vzamemo še vektor BERT vložitve začetne pojavnice [CLS]. Ta vektor pošljemo skozi polno povezan sloj mreže, ki ima enake dimenzije a drugačne uteži.

Dobljene tri vektorje zlepimo v en vektor in ga pošljemo skozi sloj polno povezane mreže, ki na koncu uporabi ohlapni maksimum. Dobljeni vektor ima L vrednosti, kjer je L število vseh relacij. Vrednosti v vektorju pa predstavljajo verjetnost, da med entitetama v stavku nastopa določena relacija. Postopek je prikazan na sliki 8.

3.3 RECON

V tem podpoglavju opišemo metodo *RECON* [1], ki poleg konteksta iz stavka s pomočjo grafovskih mrež pozornosti uporabi še podatke o entitetah, shranjenih v bazi znanja.

3.3.1 Korpus

V članku [1] za testiranje uporabijo dva korpusa iz drugih del. Prvi je korpus, ki je bil sestavljen iz podatkov z Wikipedije in baze znanja WikiData. Ustvarili so ga v članku [20]. Ima 353 tipov relacij, 372.059 stavkov za treniranje in 360.334 stavkov za testiranje. Izdelava korpusa je bolje opisana v podpoglavju 3.1.1. Drugi korpus pa je bil sestavljen iz člankov *New York Times* in *FreeBase* bazo znanja [17]. Vsebuje 53 različnih relacij, 455.771 stavkov za učenje in 172.448 stavkov za testiranje. Metoda je za izračun konteksta entitet uporabila tisto bazo znanja, ki je bila uporabljena za izdelavo korpusa.

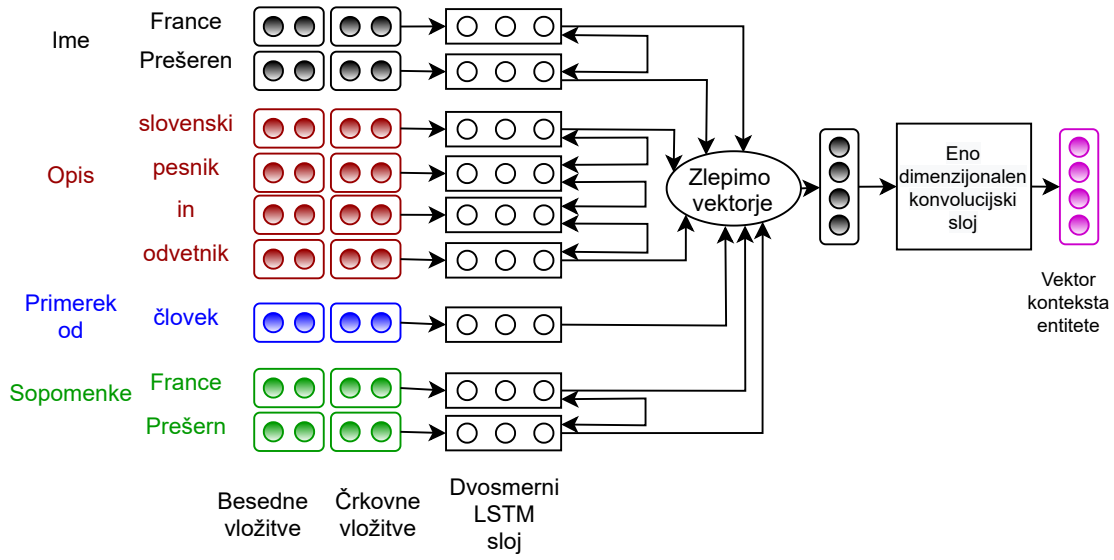
3.3.2 Metoda za napovedovanje relacij

Metoda je sestavljena iz treh korakov. Prvi korak za vsako entiteto iz relacije pridobi dodatne atribute iz grafa znanja (angl. knowledge graph [8]) in jih shrani v vektorsko predstavitev entitete. Drugi del poišče in ugotovi dodatni kontekst entitete iz relacij

shranjenih v grafu znanja. Na koncu se uporabi grafovska nevronska mreža na kontekstu in vektorji vektorske vložitve besed stavka za napovedovanje tipa relacije.

Za prva dva koraka potrebujemo graf znanja iz baze znanja (WikiData ali FreeBase). Vozlišča grafa so entitete. Vsako vozlišče vsebuje ime entitete, seznam sopomenk, opis entitete in primerek česa je entiteta (npr. avto je primerek vozila). Graf je usmerjen multigraf, kjer so povezave relacije med entitetami, ki so bile shranjene v bazi znanja.

V prvem koraku želimo pridobiti kontekst iz entitet. Za vsako lastnost entitete (ime, sopomenke, ...) zlepimo skupaj vse besedne in črkovne vložitve in jih pošljemo v dvosmerno mrežo z dolgim kratkoročnim spominom. Končne rezultate te mreže za vsako lastnost zlepimo skupaj in jih pošljemo v enodimenzionalno konvolucijsko mrežo. Končni rezultat je vektor, ki predstavlja kontekst entitete. Postopek je prikazan v sliki 9.



Slika 9: Uporaba podatkov iz baze znanja za izračun konteksta entitete.

V drugem koraku pridobimo kontekst okolice entitete v grafu baze znanja. V članku [1] so uporabili razširitev pozornosti entitete iz grafa znanja za izdelavo vložitve okolice entitete v grafu znanja [14]. Začnemo z začetnimi vektorji za vsako entiteto in vektorjem relacije za vsako relacijo. Vektorje spnemo skupaj in jih množimo z utežmi polno povezane mreže.

$$\tau_{str} = W[e_s||e_t||r_r]$$

Pomembnost trojice izračunamo s pomočjo polno povezane matrike z utežmi W_2 in prepustne popravljene linearne enote (angl. leaky ReLU):

$$b_{str} = LeakyReLU(W_2\tau_{str})$$

Pomembnosti nato normaliziramo z ohlapnim maksimumom. Normalizirana vrednost je a_{str}. Nove vektorske vložitve za prvo entiteto izračunamo z vsoto vložitve trojčkov τ in njihove pomembnosti a_{str}. V enačbi nastopa še znak za množico sosedov N_s za entiteto s, množica vseh relacij med entitetama s in t R_{st} in nelinearna funkcija σ.

$$e'_s = \sigma(\sum_{t \in N_s} \sum_{r \in R_{st}} a_{str} \tau_{str})$$

V tem koraku uporabimo še večglavo pozornost. To pomeni, da izračunamo e'_s večkrat in vsakič pri izračunu b_{sek} uporabimo različne uteži W_2. Dobljene e'_s nato zlepimo skupaj v večji vektor E'_s.

Da se bolje ohrani lastnosti začetnega vektorja entitete se novi vložitvi prišteje še staro vložitev pomnoženo z matriko uteži W^E.

$$e''_s = E'_s + W^E e_s$$

Za novo vložitev relacije se uporabi staro vložitev pomnoženo z matriko uteži W^R.

$$r''_k = W^R r_k$$

Sedaj želimo doseči, da če vektorju prve entitete e''_s prištejemo vektor relacije r''_k moramo dobiti zadnjo entiteto e''_e. Do sedaj vložitve entitet niso bile v enakem vektorskem prostoru kot vložitve relacij. Zato uporabimo polno povezano mrežo in nelinearno operacijo, da dobimo vložitve entitet za vektorski prostor relacij.

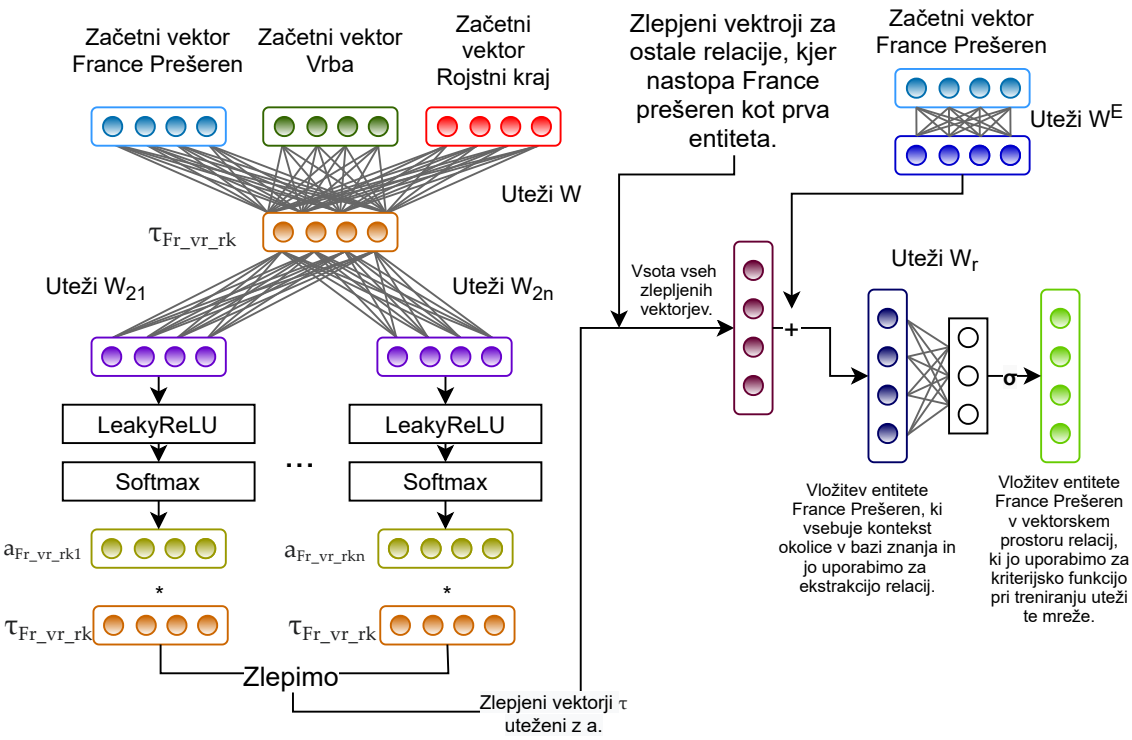
$$e^r_e = \sigma(W_r e''_e)$$

Sedaj želimo, da drži $e^r_s + r''_r = e^r_e$. Oziroma, želimo, da je vrednost $d_{\tau_{ser}} = e^r_s + r''_r - e^r_e$ čim bližje 0. Pri treniranju minimiziramo $d_{\tau_{ser}}$ za relacije, ki se pojavijo v bazi znanja in maksimiziramo $d_{\tau_{ser}}$, ki se v bazi znanja ne pojavijo. Postopek drugega koraka je prikazan na sliki 10.

V zadnjem koraku uporabimo izračunane kontekste iz prejšnjih dveh korakov. Za napovedovanje relacije uporabimo grafovsko nevronsko mrežo z splošnimi parametri [30]. Začnemo s stavkom s, ki vsebuje relacijo r med entitetama e in f. Pridobimo vektorske vložitve za besede v stavku in jim pripnemo pozicijske vektorje besed. Pozicijski vektor pove, ali entiteta pripada kateri izmed entitet v relaciji. Vektorje zlepimo skupaj v en velik vektor in ga pošljemo v dvosmerno mrežo z dolgim kratkoročnim spominom z N sloji in nato še v polno povezano omrežje s katerim iz vektorja zgradimo stohastično matriko za vsak sloj dvosmerne mreže. Mreže poimenujemo $B^{(i)}_{ef}$ za matriko i-tega sloja mreže, ko gre i od 0 do $N-1$. To storimo za vse pare entitet, znotraj stavka.

Če je $h^{(0)}_e$ vektor konteksta entitete e pridobljen v prvem koraku, ga lahko obogatimo z relacijami, ki nastopajo v stavkih učne množice na sledeč način:

$$h^{(i+1)}_e = \sum_{f \in N_e} \sigma(B^{(i)}_{ef} h^{(i)}_f)$$



Slika 10: Uporaba okolice entitete iz baze znanja za izračun konteksta entitete.

Kjer je N_e množica entitet, ki so v relaciji z e v stavku in σ nelinearna aktivacijska funkcija.

Predstavitev relacije r med entitetama e in f dobimo na naslednji način:

$$r_{ef} = \|_{i=1}^{N} h_e^{(i)} \odot h_f^{(i)}$$

Verjetnost, da nastopa relacija r med entitetama e in f znotraj stavka s dobimo z uporabo polno povezane mreže in ohlapnega maksimuma na pridobljenemu kontekstu relacije v stavku in konteksta entitet e in f v grafu znanja, ki smo ga dobili v drugem koraku.

$$P(r|e, f, s) = softmax(MLP(r_{ef} \parallel e_e'' \parallel e_f''))$$

4 Pridobivanje podatkov

Pri uporabi nadzorovanih metod za učenje modelov za napovedovanje relacij, ki smo jih opisali v poglavju 3. potrebujemo korpus. Korpus mora vsebovati besedilo z označenimi entitetami in označenimi relacijami med entitetami. Poleg tega moramo poiskati ujemanja med entitetami iz besedila in entitetami baze znanja, ker to zahteva metoda iz poglavja 3.3, da uporabi vektorske vložitve entitet iz baze znanja. Primer baze znanja je baza WikiData, ki vsebuje entitete in relacije med njimi. Entitete so shranjene pod Q ključi, relacije pa pod P ključi. Primer označenega stavka iz korpusa, ki si ga želimo:

Q1031:France Prešeren se je rodil v **Q122649:Vrbi** decembra 1800.

Relacija: (P19:Rojstni kraj, Q1031:France Prešeren, Q122649:Vrbi)

Nismo zasledili, da bi za slovenski jezik že obstajal korpus, ki ga potrebujemo, zato moramo ustvariti svojega. Za ročno označeni korpus, ki bi bil dovolj velik za treniranje metod globokega učenja, bi potrebovali preveč časa. Preostane nam možnost, da izdelamo svojega s polavtomatsko izdelavo korpusa.

4.1 Učni korpus

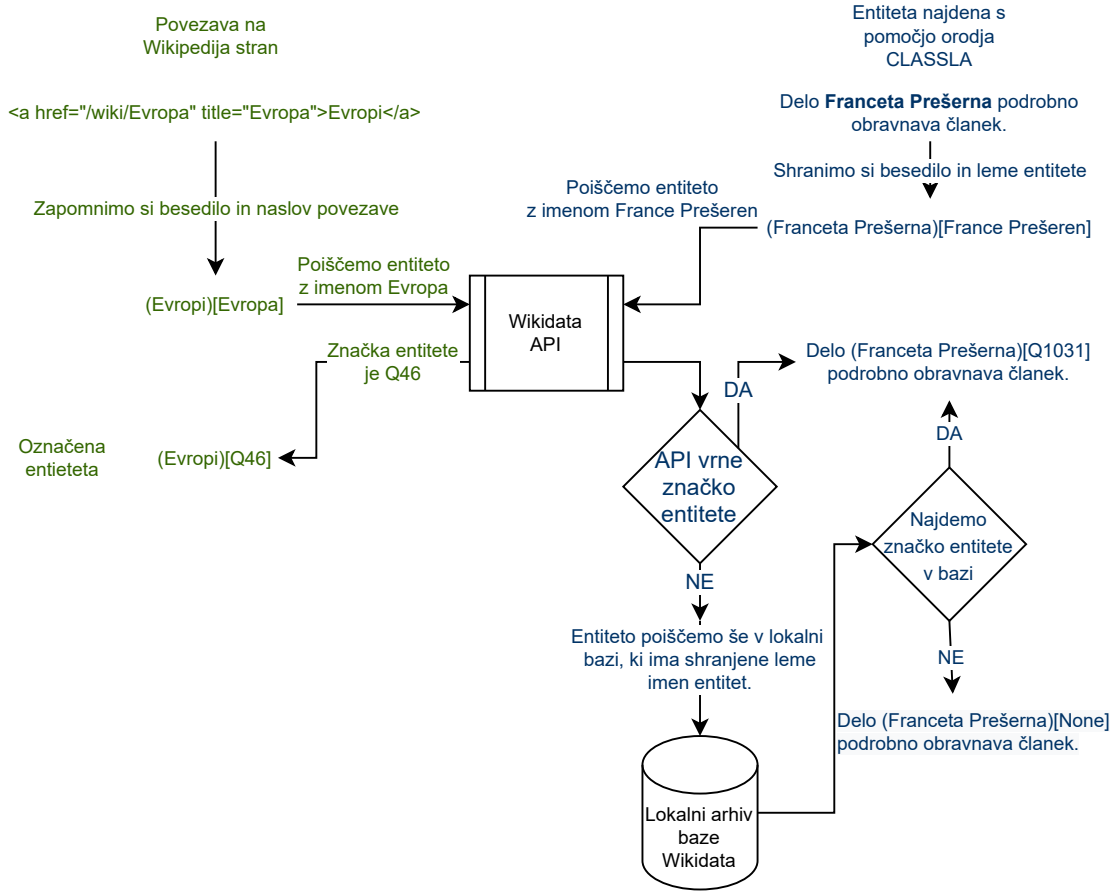
Učni korpus smo uporabili za učenje modelov. Pri gradnji učnega korpusa smo se zgledovali po postopku, ki so ga uporabili v članku [20] in je opisan v podpoglavju 3.1.1. Slovenska Wikipedija ima veliko strani z besedili in baza znanja WikiData vsebuje veliko slovenskih imen za entitete, zato lahko s postopkom, ki ga opišemo v tem podpoglavju, pridobimo slovenski korpus, ki vsebuje 30 različnih relacij, kjer imamo za vsako relacijo vsaj 1.000 primerov.F

Postopek pridobitve učnega korpusa izvedemo po korakih:

1. S pomočjo pajka pridobimo vsebino HTML iz slovenske Wikipedije.
2. Iz vsebin HTML izluščimo besedilo.
3. V besedilih poiščemo kandidate za entitete.
4. S pomočjo baze WikiData označimo entitete.
5. Parom entitet znotraj istega stavka dodelimo relacije iz baze WikiData.
6. Odstranimo relacije, ki so v pridobljenem korpusu slabo predstavljene.

4.1.1 Pridobivanje besedil iz Wikipedije

Za prenos vsebin HTML smo uporabili pajka, ki je prenašal strani iz domene *sl.wikipedia.org*. Za začetni strani smo izbrali *Ljubljana* in *Barack_ Obama*, ker vsebujeta veliko povezav znotraj domene. Pajek nam je prenesel 191.606 strani. Od teh nam ostane 140.221, če odstranimo strani, kjer se vsebina ponovi. Iz vsake strani smo nato izluščili vsebino v elementih *<p>*.



Slika 11: Postopek označevanja entitet v besedilih Wikipedije.

4.1.2 Označevanje entitet

V izluščeni vsebini smo poiskali vse povezave znotraj domene *sl.wikipedia.org*. Torej v pridobljenemu besedilu smo poiskali povezave oblike:

Omenitev entitete

Te povezave se uporabi, ko se v besedilu omeni nekaj, kar ima svojo stran na wikipediji. Besedilo znotraj povezave je omenitev, naslov pa ime strani povezave. Vsaka entiteta baze znanja WikiData se nanaša na točno eno stran na Wikipediji in vsaka stran na Wikipediji se nanaša na točno eno entiteto v bazi znanja WikiData. Ta lepa lastnost nam poda možnost, da vse take povezave označimo kot entitete in rečemo, da se nanašajo na entiteto iz WikiData, ki je povezana z Wikipedijinih stranjo povezave. Poleg entitet, ki smo jih pridobili s povezavami, poiščemo še druge kandidate za entitete. Za iskanje entitet, pojavnic in lem uporabimo označevalnik CLASSLA [12]. Najdene entitete označimo z lemami pojavnic, ki omenijo entiteto.

V drugem koraku smo povezali entitete v besedilu na entitete v bazi znanja WikiData. Na primer entiteta *Zemlja* se poveže na entiteto *Q2* v bazi. Najprej smo poskusili z uporabo WikiData API klica, ki vrne oznako entitete iz WikiData za točno ime naslova spletne strani Wikipedije. Tako smo zagotovili, da imamo

entitete, ki smo jih pridobili s pomočjo povezav, pravilno označene. Včasih pa katera izmed entitet, ki smo jih pridobili s pomočjo označevalnika imenskih entitet CLASSLA, nastopi prosto v besedilu v drugačnem sklonu ali osebi. Za take primere smo uporabili bazo WikiData, ki smo jo prenesli. Iz baze smo pobrali pare oznak (npr. *Q2*) in imen (npr. *Zemlja*) entitet. S pomočjo lematizatorja iz knjižnice CLASSLA smo pridobili leme imen, ki smo jih uporabili za proces ujemanja entitet. Če so leme iz omembe entitete v besedilu Wikipedije enake lemam imena entitete v WikiData, smo označili ta del besedila z oznako entitete iz WikiData. Postopek je prikazan na sliki 11. Postopek nam je podal dovolj primerov entitet, da smo lahko na korpusu v naslednjem koraku označili dovolj veliko število relacij. Zato ne uporabimo bolj kompleksnih pristopov, ker bi lahko v korpus tako dobili več napačnih primerov. Iz enakega razloga se izpusti zaimke, ker bi to zahtevalo uporabo odkrivanja koreferenčnosti in dodatno možnost napak v korpusu.

4.1.3 Označevanje relacij

Ko smo označili vse entitete, smo poiskali še relacije med entitetami. Cilj našega dela je, da naučimo metode, da odkrijejo omembe relacije med entitetami znotraj stavka. Zato bomo označili le relacije znotraj stavka. To storimo tako, da pogledamo katere entitete nastopijo v stavku in preverimo, če znotraj baze WikiData obstaja katera relacija med njimi. Če obstaja, označimo, da ta relacija nastopi med entitetama znotraj stavka. Od vseh strani jih je imelo 96.808 vsaj eno relacijo. Skupaj smo našli 527.136 relacij. Ker označujemo relacije glede na relacije, ki so shranjene znotraj baze WikiData, in ne relacije, ki so omenjene znotraj stavka, dobimo tudi napačne oznake relacij. V naslednjem podpoglavju na kratko razložimo, kako smo del napačnih oznak relacij odstranili iz korpusa.

4.1.4 Čiščenje korpusa

Odločili smo se, da pregledamo označene relacije in slabe primere odstranimo iz učnega korpusa. Postopek polavtomatske generacije je v besedilih Wikipedije odkril kar 558 različnih relacij. Od teh smo najprej izbrali 40, ki so vsebovale vsaj 1.000 označenih primerov, ostale pa smo zaradi premajhne zastopanosti odstranili. Izjema je bila relacija P25(mati), ki je imela je 451 primerov in smo jo obdržali, ker smo želeli, da metode napovedujejo relacije družinskih vezi. Za vsako izmed teh relacij smo preleteli označene primere in ocenili njihovo primernost.

- Relacija P47(meji na) se pojavi kar 73.342-krat v učnem korpusu. Večinoma se pojavi v stavkih, kjer se našteva več držav, med katerimi so nekatere sosednje. Relacije meji na pa ni izražene. Ker je večina označenih primerov napačnih, smo vse relacije P47 odstranili.

- Relacija P17(država, kateri objekt pripada) je bila označena 72.806-krat. Relacija je imela dobre označbe z izjemo nekaterih napačnih primerov, ki so se v besedilu velikokrat pojavili. Da odstranimo del takih primerov, smo od vseh označb za vsak par entitet uporabili samo en primer. Po filtriranju nam je ostalo skupno 13.333 primerov.

- Relacija P530(diplomatski odnosi) je bila označena na vseh parih držav, ki imajo diplomatske odnose. Vendar pa tako kot pri relaciji P47, relacija P530 ni bila izražena med pari, ki so bili označeni. Zato smo izpustili vseh 61.464 oznak.

- Relacija P131(se nahaja) sicer v večini primerov ni bila izražena direktno, vendar pa bilo iz konteksta možno razbrati, da ta relacija drži. Zato smo obdržali vseh 31.745 primerov.

- Relacija P150(administrativne podenote) je podobna in velikokrat obratna relacija od P131. Zato smo obdržali vseh 22.061 primerov.

- Oznake relacij P31(primerek od), P106(poklic), P527(vsebuje), P156 (naslednik), P155(predhodnik), P361(je del), P3450(sezona tekmovanja), P138 (poimenovano po), P641(šport povezan s tem), P172(etnična skupina), P276 (lokacija), P607(bitka povezana s tem), P1001(spada pod upravo), P140(vera), P136(žanr), P39(uradni položaj) ter P463(članica) smo ocenili kot dobre in smo zato obdržali vse primere.

- Relacija P279(podpomenka od) je imela podobne napačne primere kot relacija P530. Na primer pri opisih oseb, kjer so bili naštevani poklici so bili nekateri poklici zaznani kot podpomenke drugih (pesnik podpomenka pisatelja). Te relacije niso bile direktno izražene in jih zato obravnavamo kot napačne. Ker so se poklici večinoma ponavljali smo se teh primerov v večini znebili tako, da smo uporabili le en primer za vsak par entitet. Od 12.621 primerov smo obdržali 2.589 primerov.

- Relacija P19(kraj rojstva) ima 9.812 označenih primerov. Ker so osebe velikokrat omenjene v istem stavku kot njihov kraj rojstva, smo obdržali le stavke, ki vsebujejo besede povezane z rojstvom. Po filtriranju nam je ostalo 8.773 primerov.

- Pri relaciji P27(država državljanstva) bi pa radi odstranili vse primere, ki omenjajo rojstvo ali smrt, ker država državljanstva ni izražena s krajem rojstva ali krajem smrti. Od 8.861 je ostalo 3.981 primerov.

- Učni korpus vsebuje 9.226 primerov relacije P36(glavno mesto) in relacije P1376 (glavno mesto od). Stavki s to relacijo izražajo relacijo P131(se nahaja v), nikjer pa ni nekega dodatnega podatka o glavnem mestu. Zato smo se odločili, da združimo primere P36, P1376 in P131 pod oznako P131.

- Relacija P20(kraj smrti) nastopi v korpusu 7.041-krat. Podobno kot pri relaciji P19 smo se omejili le na oznake iz stavkov, ki vsebujejo besede povezane s smrtjo. Po filtriranju je ostalo 5.130 primerov.

- Oznake relacij P190(pobrateno mesto), P1889(podobno a drugače), P206(ob vodnem telesu), P30(kontinent), P737(pod vplivom (za umetnike, filozofe, .. pod vplivom obdobja)), P1317(bil aktiven v) ter P1344(je nastopil v) večinoma niso izražale teh relacij zato smo odstranili vse primere.

- Relacije P3373(sorojenec), P40(otrok), P22(oče) in P25(mati) izražajo družinske vezi. Za boljšo točnost smo ohranili le relacije v stavkih, ki vsebujejo izbrane ključne besede, ki izražajo družinske vezi.

- Relaciji P50(avtor) in P800(pomembna dela) sta si zrcalni. Ker za relacijo P800 v besedilih ni bilo izraženo če so dela pomembna smo se odločili, da relaciji združimo v P50(avtor) z 2.068 primeri.

Stavke označene z relacijami, ki so ostale po filtriranju smo razporedili v tri množice. To smo storili tako, da smo od vsakih desetih stavkov izbrali sedem za učno množico, enega za validacijsko množico in dva za testno množico. Vsak stavek je nastopal izključno v eni izmed množic. V tabeli 1 prikažemo porazdelitev števila označenih primerov v označenih množicah in število označenih relacij preden smo relacije filtrirali in združevali. Iz tabele je razvidno, da z naključno delitvijo stavkov dosežemo podobno porazdelitev označenih relacij v vseh treh množicah. Po čiščenju je od 527.136 relacij ostalo skupno 214.377 relacij. Poleg označenih relacij smo naključno izbrali nekaj parov entitet za katere nimamo označenih relacij in jih označili z prazno relacijo P0. Prazna relacija označuje primere kjer med entitetama ne nastopa relacija.

4.1.5 Primeri uporabljenih relacij

V tem podpoglavju bomo za vsako relacijo prikazali pravilen primer stavka, ki vsebuje relacijo za lažje razumevanje relacij.

- **P131 se nahaja v** med Šetarova in Občini Lenart:
  Šetarova je gručasto naselje na štajerskem v **Občini Lenart**.

- **P150 administrativna podenota** med Romunija in Bukarešta:
  Decembra 1916 je padla **Bukarešta** s čimer je bila **Romunija** praktično izločena iz vojne, njen popolni kolaps pa je morala reševati Rusija in zahodni zavezniki.

- **P31 primerek** med Škocjan in naselje:
  Škocjan je **naselje** v Občini Škocjan.

- **P106 poklic** med Franu Šukljetu in politiku:
  Imenuje se po slovenskem **politiku** in zgodovinarju **Franu Šukljetu**.

- **P527 vsebuje** med poletnih olimpijskih igrah in šprintu:
  Leta 1972 je nastopil v **šprintu** na **poletnih olimpijskih igrah**.

- **P17 država, kateri objekt pripada** med Šlisselburg in Rusiji:
  Šlisselburg je mesto in rečno pristanišče v **Rusiji** ob iztoku reke Neve iz Ladoškega jezera.

- **P156 naslednik** med petek in sobotnem:
  Versko obeleževanje šabata se prične v **petek** ob sončnem zahodu in konča ob **sobotnem** zahodu sonca.

WikiData značka	ime relacije	celoten korpus	učna množica	validacijska množica	testna množica
P0	prazna	0	30.721	4.455	8.901
P131	se nahaja v	31.745	28.644	4.110	8.217
P150	administrativne podenote	22.061	15.400	2.165	4.496
P31	primerek od	18.463	13.064	1.819	3.580
P106	poklic	16.605	11.543	1.670	3.392
P527	vsebuje	13.521	9.460	1.410	2.651
P17	država, kateri objekt pripada	7.2806	9.299	1.333	2.701
P156	naslednik	10.841	7.733	1.041	2.067
P155	predhodnik	10.709	7.647	1.030	2.032
P361	je del	9.930	6.988	1.005	1.937
P19	kraj rojstva	9.812	6.094	880	1.799
P3450	sezona športne prireditve	5.820	4.105	603	1.112
P20	kraj smrti	7.041	3.650	491	989
P138	poimenovano po	4.459	3.083	440	936
P641	šport povezan s tem	4.147	2.910	410	827
P172	etnična skupina	4.015	2.837	366	812
P27	država državljanstva	8.861	2.830	389	762
P276	kraj, lokacija	3.641	2.575	361	705
P3373	sorojenec	3.139	2.206	322	611
P607	bitka povezana s tem	3.013	2.093	301	619
P1001	spada pod upravo	2.667	1.884	258	525
P279	podpomenka od	12.621	1.818	271	500
P50	avtor	1.344	1.444	193	431
P140	vera	2.052	1.421	219	414
P136	žanr	1.673	1.191	160	322
P40	otrok	2.455	1.047	133	238
P39	uradni položaj	1.294	910	129	255
P463	član organizacije	1.231	902	113	216
P22	oče	1.989	837	102	203
P25	mati	451	220	28	38

Tabela 1: Relacije in njihovo število v nefiltriranem učnem korpusu ter v učni validacijski ter testni množici.

- **P155 predhodnik** med sobotnem in petek:
  Versko obeleževanje šabata se prične v **petek** ob sončnem zahodu in konča ob **sobotnem** zahodu sonca.

- **P361 je del** med debla in drevo:
  Štor je ostanek **debla** s koreninami, ki ostane, ko posekajo **drevo**.

- **P19 kraj rojstva** med Šaban Šaulić in Šabcu:
  Šaban Šaulić se je rodil 6. septembra 1951 v srbskem **Šabcu**.

- **P3450 sezona športne prireditve** med sezoni 1965 in Formule 1:
  BRDC International Trophy 1965 je bila peta neprvenstvena dirka **Formule 1** v **sezoni 1965**.

- **P20 kraj smrti** med Abdul Hamid I. in Konstantinoplu:
  Abdul Hamid I. je umrl v **Konstantinoplu** 7. aprila 1789, star 64 let.

- **P138 poimenovano po** med Jadransko morje in Adria:
  Druga **Adria** je na področju Benetk in po njej je **Jadransko morje** dobilo svoje ime.

- **P641 šport povezan s tem** med Amaterskega prvenstva Francije in tenisu:
  Rezultati **Amaterskega prvenstva Francije** 1934 v **tenisu** za ženske posamično.

- **P172 etnična skupina** med Rustavelija in gruzijskega:
  Od leta 1921 nosi ime **gruzijskega** nacionalnega pesnika **Rustavelija**.

- **P27 država državljanstva** med Peter I. in Rusiji:
  V **Rusiji** je bil najpomembnejši absolutni vladar **Peter I.**, kajti njemu gre največ zaslug za takratni razvoj.

- **P276 kraj lokacija** med poletne olimpijske igre in Peking:
  Peking je leta 2008 gostil **poletne olimpijske igre**.

- **P3373 sorojenec** med Peter in Cene Prevc:
  Njegova starejša brata sta **Peter** in **Cene Prevc**.

- **P607 bitka povezana s tem** med redne nemške kopenske vojske in drugo svetovno vojno:
  28. lovski polk je bil lovski polk v sestavi **redne nemške kopenske vojske** med **drugo svetovno vojno**.

- **P1001 spada pod upravo** med FDA in ameriški:
  Leta 2002 je **ameriški** vladni urad **FDA** odobril uporabo GHB-ja za zdravljenje narkolepsije.

- **P279 podpomenka od** med lev in velikih mačk:
  Napadi drugih velikih **mačk** kot je **lev**, je kombinacija presenečenja, zasede in lova s krdelom.

- **P50 avtor** med Da Vincijeva šifra in Dan Brown:
  Pisatelj **Dan Brown** je zgodbo nadaljeval v svojem romanu **Da Vincijeva šifra**.

- **P140 vera** med mošejo in islamsko:
  Stolnico je na spoštljiv način pretvoril v **islamsko mošejo** in s tem utrdil islamsko oblast v mestu.

- **P136 vera** med Arctic Monkeys in indie rock:
  Arctic Monkeys je angleška **indie rock** skupina, ustanovljena leta 2002.

- **P40 otrok** med Zmagom Jelinčičem in Klemna Jelinčiča Boeta:
  Leta 1973 se je poročila še enkrat, z **Zmagom Jelinčičem** in sta istega leta dobila sina **Klemna Jelinčiča Boeta**.

- **P39 uradni položaj** med Joe Biden in podpredsednika ZDA:
  Njegov namestnik Joe **Biden** je bil prav tako ponovno izvoljen za **podpredsednika ZDA**.

- **P463 član organizacije** med Rokom in BQL:
  Od leta 2016 skupaj s svojim bratom **Rokom** pojeta v duetu pod imenom **BQL**.

- **P22 oče** med Arwen in Elrondom:
  Poročila se je z **Elrondom**, s katerim sta imela dva sinova in hčerko **Arwen**.

- **P25 mati** med Hebo in Hero:
  Poročen je s svojo starejšo sestro **Hero**, ki mu je rodila Aresa, Hefajsta in **Hebo**.

4.2 Testni Korpus

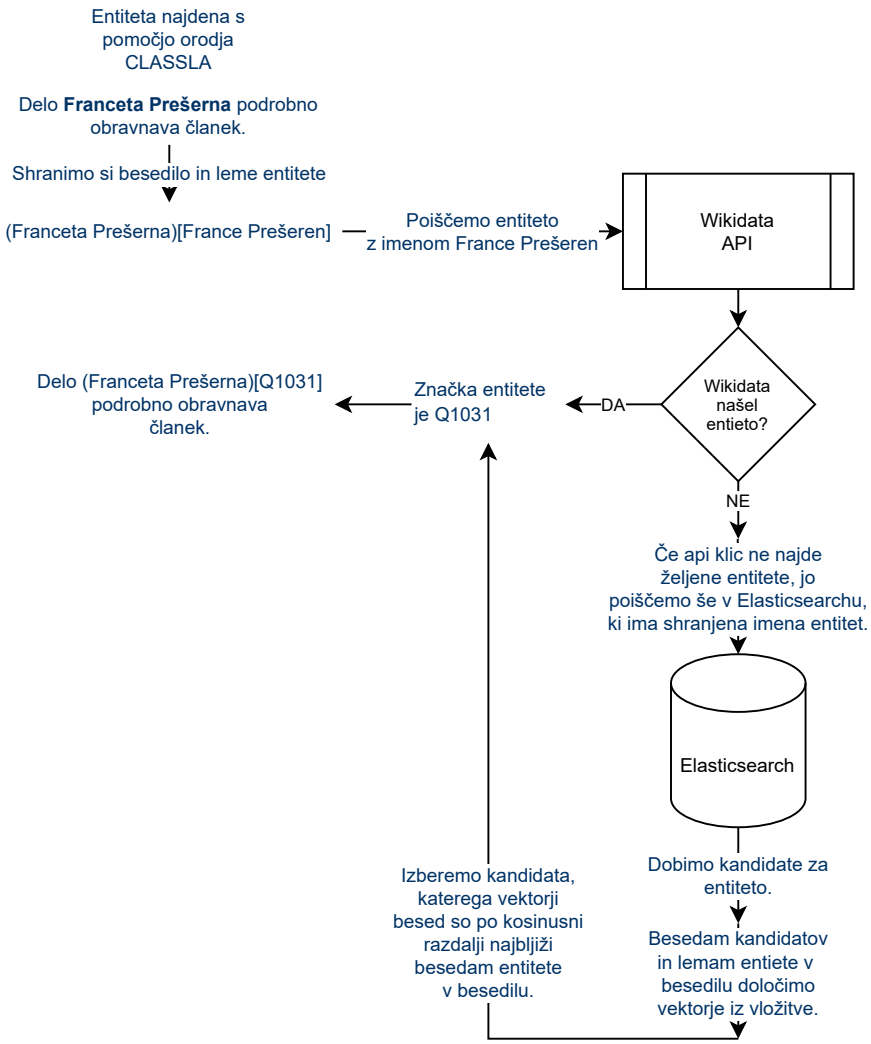
Na korpusu, ki smo ga pridobili iz slovenske Wikipedije smo naučili naše modele. Testirati pa jih želimo še na drugačnih podatkih, da lahko preverimo, kako dobro se metode obnesejo na besedilih, ki se razlikujejo od učnih. Ker tokrat ne uporabimo besedila iz Wikipedije, se ne moremo zanašati na notranje povezave za ujemanje entitet in zato uporabimo drugačen pristop.

Postopek pridobitve korpusa izvedemo po korakih:

1. S pomočjo pajka pridobimo vsebino HTML iz člankov 24ur.com.
2. Iz vsebin HTML izluščimo besedilo.
3. V besedilih poiščemo kandidate za entitete.
4. S pomočjo Elasticsearch-a poiščemo entitete iz baze baze WikiData, da označimo entitete.

4.2.1 Pridobivanje testnih podatkov

S pomočjo pajka smo prenesli vsebino 956 člankov. Članki so razdeljeni na novice, športne novice ter trače, ki vsebuje modo, trače in zabavno industrijo. Iz strani smo vzeli vsebino članka in v vsebini pobrali besedilo znotraj elementov <p>.



Slika 12: Postopek označevanja entitet v testnem korpusu.

4.2.2 Označevanje entitet v testnem korpusu

Za delitev besedila na stavke in pojavnice tudi tokrat uporabimo knjižnico CLASSLA [12]. Knjižnico CLASSLA uporabimo še za dodeljevanje lem pojavnicam in označevanje imenskih entitet v besedilu. Sedaj imamo imamo označene lokacije entitet želimo pa jih še povezati z entitetami v bazi znanja WikiData in jim določiti ustrezno WikiData značko. Če ima entiteta identičen zapis kot entiteta v WikiData bazi, ali pa so njene leme identične zapisu entitete v bazi WikiData, lahko najdemo ujemajočo entiteto kar preko API klica WikiData. Žal je pa takih primerov v besedilu bolj malo in zato potrebujemo algoritem za iskanje ujemanj ostalih entitet v bazi WikiData.

Prva metoda, ki smo jo preizkusili, je Opentapioca [4]. Opentapioca uporabi Apache Solr za hitro iskanje kadidate entitet baze WikiData. Nato uporabi naučeno metodo podpornih vektorjev za izbiro najboljšega kandidata. Za lokalno namestitev smo najprej shranili slovenske entitete v lokalni Apache Solr in nato trenirali metodo

SVM na označenih entitetah v našem učnem korpusu. Pri učenju je klasifikator na učni množici dosegel 44% točnost in 13% priklic. Tudi na besedilih člankov spletne strani 24ur.com se ni izkazal in smo se zato odločili, da ga ne bomo uporabili.

Druga metoda, ki smo jo preizkusili je Falcon 2.0 [18]. Falcon uporabi Elasticsearch za iskanje kandidatov entitet baze WikiData. Nato s pomočjo levensteinov razdalje določi najboljšo izbiro entitete. Elasticsearch je med kandidati za entitete velikokrat pokazal pravilno entiteto. Pravilna entiteta pa ni imela vedno najvišje ocene, ker Elasticsearch ocenjuje rezultate glede na Levensteinovo razdaljo in ne na vsebino besed.

Za ujemanje entitet smo zato uporabili naš postopek, ki ima enaka koraka kot prejšnji dve metodi. Najprej uporabimo hitro iskanje po bazi WikiData za kandidate in nato izberemo kandidata, ki dobi najboljšo oceno po našem kriteriju. Bazo slovenskih WikiData entitet shranimo na Elasticsearch. Za iskanje uporabimo mehko iskanje (angl. fuzzy search) z AUTO parametrom. Torej, ko primerjamo besede entitete v besedilu in ime entitete v bazi WikiData imajo lahko besede, ki imajo največ dve črki, Levensteinovo razdaljo največ 0, tiste besede, ki imajo tri do pet črk, Levensteinovo razdaljo največ 1 in ostale besede Levensteinovo razdaljo največ 2. V drugem koraku primerjamo vsebino kandidatov in entitete v besedilu. Vsebino kandidata predstavimo kot vsoto vektorskih vložitev besed imena, vsebino entitete pa predstavimo kot vsoto vektorskih vložitev lem entitete v besedilu. Uporabimo FastText [9] vložitve, ki so bile natrenirane na slovenskih straneh Wikipedije. Entiteti sta podobni, če imata vektorja, ki predstavljata njuno vsebino, majhno kosinusno razdaljo. Če imajo vsi kandidati razdaljo večjo kot 0.2, ne izberemo nobenega in entiteto označimo kot prazno. Postopek je ponazorjen na sliki 12. Metode bodo lahko vseeno odkrile relacije s prazno entiteto vendar pa metoda RECON, ki je opisana v podpoglavju 3.3, ne bo imela dodatnih podatkov o kontekstu entitete iz baze WikiData. V testnem korpusu smo v 956 člankih s CLASSLA označili 27.483 entitet, od tega smo jih 18.477 označili z WikiData entiteto, ostalih 9.006 entitet pa smo označili kot prazne entitete.

V našem učnem korpusu smo pridobili drugačen nabor entitet, ker smo označili večino entitet preko spletnih povezav, namesto z označevalnikom CLASSLA. Zato smo za bolj izčrpno testiranje izdelali še dodatni testni korpus, ki vsebuje enaka besedila člankov s spletne strani 24ur.com, vendar pa tokrat izberemo entitete na drugačen način. Najprej uporabimo postopek povezovanja entitet na bazo WikiData opisan v tem podpoglavju, da označimo vse kombinacije največ treh soležnih pojavnic. Vse kombinacije, ki nam jih uspe povezati na bazo WikiData, obdržimo, ostale pa odstranimo. V kolikor se dve taki kombinaciji pojavnic prekrivata, izberemo tisto, ki doseže manjšo kosinusno razdaljo med postopkom povezovanja entitet. S tem požrešnim pristopom smo označili 68.331 entitet z WikiData entitetami.

4.2.3 Označevanje relacij v testnem korpusu

Za označevanje relacij v učnem korpusu smo uporabili bazo znanja WikiData. Kot smo že prej omenili, poteka tako označevanje na podlagi relacij iz splošnega znanja in ne relacij, ki so izražene v besedilu. Torej nam tako označevanje prinese šum, ki ga pa med testnimi podatki ne želimo. Poleg tega smo za označevanje entitet v testnem korpusu morali uporabiti manj točen postopek za označevanje entitet, kar pa bi pri

avtomatskem označevanju relacij prineslo še dodatne napake. Ročno označevanje relacij je pa zamudno, zato testnega korpusa ne bomo označevali z relacijami. Pri testiranju bomo iskali relacije za vse pare entitet, ki nastopajo v istem stavku. Pri ocenjevanju se bomo osredotočili na pravilne pozitive in lažne pozitive in preskočili relacije, ki jih je metoda zaznala kot prazne.

5 Učenje in testiranje metod

V tem poglavju predstavimo testiranje naših metod ter naše rezultate. Za testiranje in učenje metod iz poglavja 3 uporabimo podatke, ki smo jih pridobili po postopku, ki je opisan v poglavju 4. Delitev podatkov na učne in testne podrobneje opišemo v podpoglavju 5.1. Nato sledi poglavje 5.2, kjer opišemo pomembne parametre, ki smo jih uporabili pri učenju metod. V podpoglavju 5.3 pa predstavimo rezultate metod v obliki krivulj točnosti v odvisnosti od priklica ter tabel, ki vsebujejo priklice, točnosti in normalizirani diskontirani kumulativni prispevek.

5.1 Delitev in priprava podatkov za učenje in testiranje modelov

Naš učni korpus, ki smo ga polavtomatsko zgradili na besedilih slovenske Wikipedije in je opisan v podpoglavju 4.1, smo razdelili na učno, validacijsko in testno množico. Učna množica vsebuje 71.140 stavkov korpusa in se uporabi za učenje modelov. Validacijska množica vsebuje 10.163 stavkov in se uporabi med učenjem za ustavitveni pogoj, da se model pri učenju ne prilagodi pretirano učni množici. Testna množica vsebuje preostalih 20.325 stavkov in jo uporabimo za testiranje metod na podatkih, ki so podobni učnim. Vsem trem množicam smo dodali še prazne relacije *P0*. Prazna relacija pomeni, da med entitetama ne nastopa nobena izmed relacij, ki jih bo naš model napovedal. Prazno povezavo smo dodali za vsak par entitet *e1* in *e2*, kjer med *e2* in *e1* poteka relacija med *e1* in *e2* pa ni relacije. To smo storili, ker so modeli drugače napovedovali relacijo ne glede na vrstni red entitet. Poleg tega smo za vsak stavek naključno izbrali par dveh entitet med katerima ne poteka relacija in med entiteti dodali prazno povezavo. V tabeli 1 je zapisano število relacij za vsako množico učnega korpusa.

Poleg podatkov, ki smo jih uporabili za testiranje na podobnih podatkih, smo uporabili še testno množico s stavki iz člankov s spletne strani 24ur.com, kjer smo entitete označevali z označevalnikom entitet CLASSLA. Za lažje testiranje smo iz množice vzeli vsak šesti stavek izmed tistih, ki vsebujejo med dve in tri entitete. Tako nam je ostalo 896 stavkov, ki jih uporabimo, da vidimo kako se naučeni modeli obnesejo na drugačnih podatkih. Gradnja testnega korpusa je opisana v podpoglavju 4.2. Poleg tega smo pa za testiranje uporabili še množico testnega korpusa, kjer smo entitete označevali s požrešnim pristopom. Ker smo v tej bazi označili veliko število entitet smo se odločili, da bomo iz vsakega tridesetega stavka vzeli naključni dve entiteti in testirali relacije med tema dvema entitetama. Modeli so na zadnji testni množici dosegali nižje točnosti. Iz tega sklepamo, da je bil način označevanja entitet neprimeren in smo zato v tem poglavju predstavili le rezultate prvih dveh testnih množic.

5.2 Učenje metod

V tem podpoglavju predstavimo treniranje modelov po metodah, ki smo jih opisali v poglavju 3. Za vsako metodo predstavimo pomembne parametre, ki smo jih uporabili pri učenju.

5.2.1 Metoda s povratno nevronsko mrežo z dolgim kratkoročnim spominom (LSTM)

Prva metoda, ki smo jo učili napovedovati relacije, je metoda s povratno nevronsko mrežo z dolgim kratkoročnim spominom (LSTM) opisana v podpoglavju 3.1. Za vektorske vložitve smo namesto angleških vektorskih vložitev GloVe [16], ki so jih uporabili v članku [20], uporabili vložitve FastText [9], ki so bile naučene na straneh slovenske Wikipedije. V članku [20] je predstavljenih več različnih metod. Za naš model smo izbrali metodo, ki uporabi utežene vektorje ostalih relacij znotraj stavka, ker ta metoda v članku doseže najboljše rezultate. Izbrano metodo smo opisali v podpoglavju 3.1.3. Z metodo smo učili naš model sedem epohov na učnih podatkih dokler se je rezultat kriterijske funkcije na validacijski množici izboljševal.

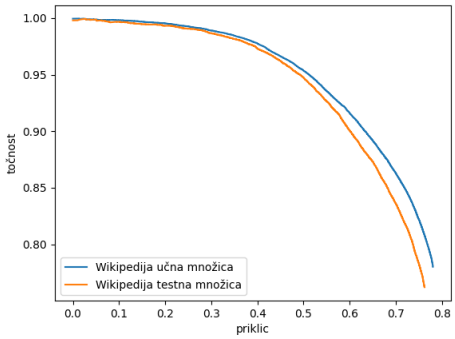
5.2.2 Metoda, ki uporabi vektorske vložitve BERT SloBERTa 2.0

Druga metoda, ki smo jo uporabili temelji na prenosu znanja iz vektorskih vložitev transformerja BERT. To metodo iz članka [26] bolj podrobno opišemo v podpoglavju 3.2.2. Za naše vektorske vložitve uporabimo že naučen transformer SloBERTa 2.0 [23], ki je bil naučen na petih velikih slovenskih korpusih, ker je bil v času pisanja edini javno dostopen naučen enojezični transformer za slovenski jezik. Zaradi časovne kompleksnosti učenja in napovedovanja relacij smo se omejili na stavke z največ štiriinsedemdesetimi pojavnicami. Model smo učili šest epohov na učni množici iz besedil Wikipedije dokler se je kriterijska funkcija na validacijski množici izboljševala.

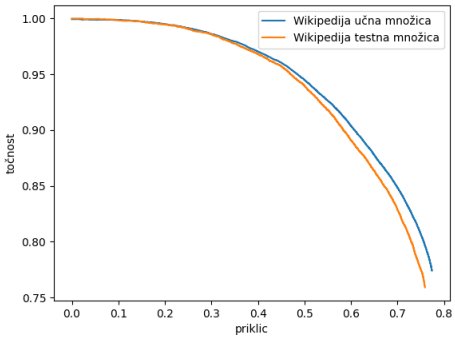
5.2.3 Metoda, ki uporabi vektorske vložitve BERT CroSloEngual 1.1

V tem podpoglavju predstavimo rezultate metode, ki smo jo uporabili tudi v prejšnjem podpoglavju 5.2.2 z enakimi parametri. Namesto vložitev SloBERTa pa tokrat uporabimo vložitve CroSloEngual [22]. CroSloEngual so večjezične vložitve, ki so naučene na angleških, hrvaških in slovenskih korpusih. Model lahko tako učimo tudi na angleških podatkih, ki so jih pridobili iz angleške Wikipedije v članku [20], ker so CroSloEngual naučene za slovenski in angleški jezik. Za učne podatke uporabimo celoten korpus, ki je bil uporabljen za testiranje in učenje v [20]. Iz korpusa smo odstranili vse tipe relacij, ki ne nastopajo v našem slovenskem filtriranem korpusu. Za testiranje uporabimo tri modele. Vsi trije modeli za učenje uporabijo slovensko učno množico s tem, da drugi model poleg slovenske učne množice uporabi za učenje še 20% stavkov angleških podatkov tretji model pa 100% angleških podatkov. Cilj naloge je ustvariti model napovedovanja relacij za slovenski jezik, zato pri testiranju ne uporabimo dodatnih angleških testov.

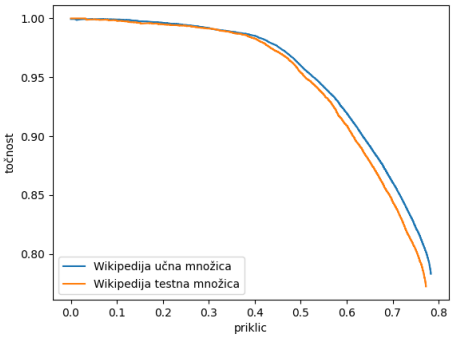
Na podslikah slike 13 lahko opazimo, da vsi trije modeli dosežejo točnost okoli 78% za testno in učno množico besedil slovenske Wikipedije. Dodatni angleški učni podatki torej ne prinesejo boljše rezultate na teh dveh množicah. Iz slike 14 pa lahko razberemo, da model, ki uporabi samo slovenske učne podatke, doseže za 2% boljšo točnost kot pa model, ki se dodatno uči na 20% angleške učne množice. Skupna točnost pa je višja kot pri metodi LSTM in vložitvah SloBERTa. Angleški učni podatki niso pomagali pri točnosti modelov zato bomo za primerjavo z drugimi



(a) Učenje na slovenskih podatkih.

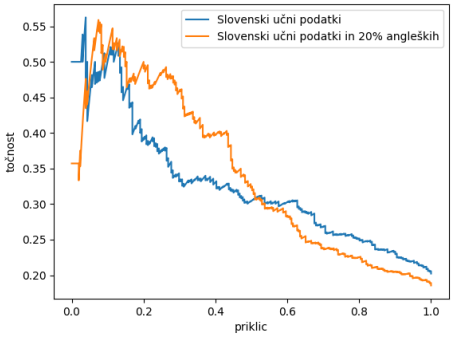


(b) Učenje na slovenskih podatkih in 20% angleških podatkih.



(c) Učenje na slovenskih podatkih in 100% angleških podatkih.

Slika 13: Krivulje priklica in točnosti na testni in učni množici Wikipedije za za metodo z vložitvami BERT CroSloEngual.



Slika 14: Rezultati testne množice člankov 24ur.com za model z vložitvami BERT CroSloEngual, ki je naučen na slovenskih učnih podatkih ter model z vložitvami BERT CroSloEngual, ki je naučen na slovenskih učnih podatkih in 20% angleških učnih podatkov.

metodami uporabili model, ki je bil naučen samo na slovenskih podatkih. Vložitve CroSloEngual dosežejo višjo točnost kot pa vložitve SloBERTa. Dodatni angleški

učni podatki niso vplivali na točnost zato sklepamo, da je razlog za višjo točnost boljša predstavitev jezika v vložitvah CroSloEngual.

5.2.4 Metoda RECON

Zadnjo metodo, ki smo jo testirali, uporabi dodatni kontekst entitet, ki ga izračuna iz baze znanja s pomočjo grafovskih nevronskih mrež. Metoda je bolj podrobno opisana v podpoglavju 3.3. Za izračun konteksta smo uporabili bazo znanja WikiData, ki vsebuje slabih sto milijonov entitet. Ker bi za obdelavo vseh entitet potrebovali preveč časa smo se omejili na 559.182 entitet. Med te entitete spadajo vse entitete, ki so jih uporabili v članku [1], ko so učili metodo na angleških besedilih Wikipedije, ter vse entitete, ki smo jih označili v našem učnem korpusu iz besedil Wikipedije. Za postopek učenja konteksta entitet, ki je prikazan na sliki 9, smo uporabili vektorske vložitve FastText [9], ki so bile naučene na straneh slovenske Wikipedije. Za učenje uteži konvolucijskega sloja smo uporabili dvesto epohov in nato še dvajset epohov za učenje vseh uteži povratnih nevronskih mrež z dolgim kratkoročnim spominom, ki se uporabijo pri izračunu konteksta entitet. Za učenje grafovske nevronske mreže, ki izračuna kontekst okolice entitete entitete e, smo uporabili entitete, ki so v grafu baze znanja, kjer so vozlišča entitete in povezave relacije, oddaljene za največ dve relaciji. Grafovsko mrežo smo učili dvesto epohov. Metodo RECON smo nato učili na učnih podatkih Wikipedije devet epohov dokler se je izboljševala kriterijska funkcija na validacijski množici Wikipedije.

5.3 Rezultati metod

Rezultate metod prikažemo v tabeli 2 in tabeli 3 s točnostjo (T), priklicom (P) in normaliziranim diskontiranim kumulativnim prispevkom (KP) (angl. normalized discounted cumulative gain). Če metoda ni napovedala relacije vsaj enkrat, smo to v tabeli označili z /. Za vsako relacijo smo najboljše rezultate odebelili. Testna množica 24ur.com ni imela označenih relacij in zato nimamo podatka o številu vseh pojavitev relacije. Tabela 3 ima zato pod priklicom število najdenih pravilnih primerov relacije. S priklicom in točnostjo prikažemo količino najdenih relacij in verjetnost, da je relacija, ki jo metoda najde pravilna. Normalizirani diskontirani kumulativni prispevek pa uporabimo za vrednotenje ocene samozavesti. Pri napovedi relacije nam model poleg relacije poda oceno samozavesti s katero pove verjetnost, da je napovedana relacija pravilna. Ocena samozavesti bi morala biti višja pri napovedih pravilnih relacij kot pa pri napačnih. Za izračun KP uredimo napovedi padajoče po oceni samozavesti. Vrednost v_i pravilne napovedi nastavimo na 1, vrednost v_i napačne napovedi pa nastavimo na 0. Nato izračunamo diskontirani kumulativni prispevek po formuli:

$$\sum_{i=1}^{|napovedi|} \frac{v_i}{\log_2(i+1)}$$

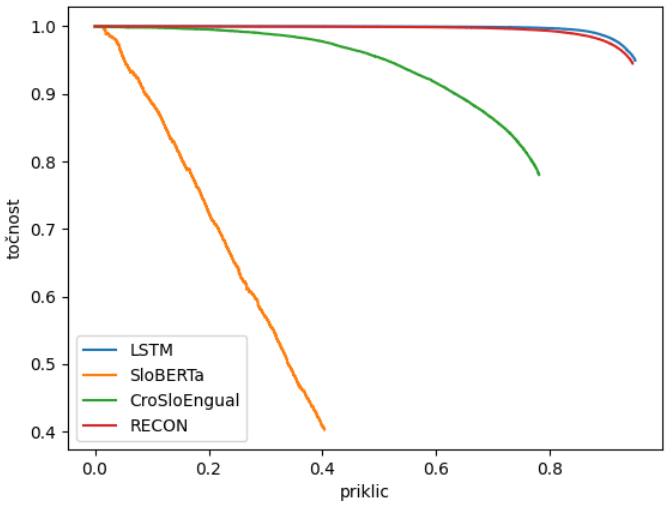
Prispevek nato normaliziramo tako, da ga delimo z diskontiranim kumulativnim prispevekom, ki ga izračunamo na razporeditvi napovedi kjer se vse pravilne napovedi pojavijo pred napačnimi.

5.3.1 Rezultati na učni in testni množici Wikipedije

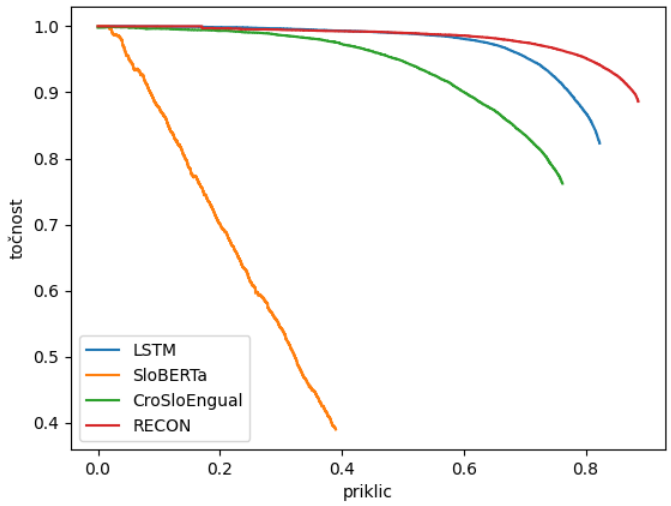
Slika 15 vsebuje krivulje skupne točnosti v odvisnosti od priklica za modele na učni množici. Metodi LSTM in RECON sta se najbolje prilegli učni množici in dosežeta skupno točnost in priklic okoli 95%. Metoda z vložitvami CroSloEngual doseže točnost in priklic okoli 78% in metoda z vložitvami SloBERTa doseže najnižjo skupno točnost in priklic okoli 40%.

Na sliki 16 imamo krivulje skupne točnosti v odvisnosti od priklica za modele na testni množici Wikipedije. Modela, ki uporabita vložitve BERT dosežeta podobno skupno točnost in priklic kot na učni množici. Metoda RECON doseže za 5% nižjo točnost in priklic, LSTM metoda pa za kar 13% nižjo točnost in priklic. Točnosti metod na obeh množicah za vse modele so zapisane v tabeli 4.

V tabeli 2 imamo za vsak model zapisane vrednosti priklica, točnosti in normaliziranega diskontiranega kumulativnega prispeveka za vsako relacijo na testni množici Wikipedije. Metoda LSTM je v primerjavi z ostalimi modeli dosegla povprečne rezultate za vse relacije. Edini model, ki ni napovedoval vseh relacij je model z vložitvami SloBERTa. Poleg tega je za vse relacije dosegel priklice nižje od metode z vložitvami CroSloEngual in metode RECON. Vsi modeli razen modela z vložitvami SloBERTa so dosegli visok diskontiran kumulativni prispevek za skoraj vse relacije z izjemo P25(mati). Ta relacija je bila edina, ki je imela manj kot 1.000 primerov v našem učnem korpusu. Edina metoda, ki je dosegla dobre rezultate za to relacijo, je bila metoda z vložitvami CroSloEngual. Vse najboljše priklice sta dosegli metoda z vložitvami CroSloEngual in metoda RECON. Metoda z vložitvami CroSloEngual je bolje napovedovala družinske vezi P22(oče), P25(mati), P40(otrok) in P3373(sorojenec), metoda RECON pa relacije P136(žanr), P138(poimenovano po), P140(vera) ter P463(član). Obe metodi pa sta malo slabše napovedovali P279(podpomenka od) in P3373(sorojenec).



Slika 15: Krivulje točnosti v odvisnosti od priklica za modele na učni množici Wikipedije.



Slika 16: Krivulje točnosti v odvisnosti od priklica za modele na testni množici Wikipedije.

5.3.2 Rezultati na učni in testni množici 24ur.com

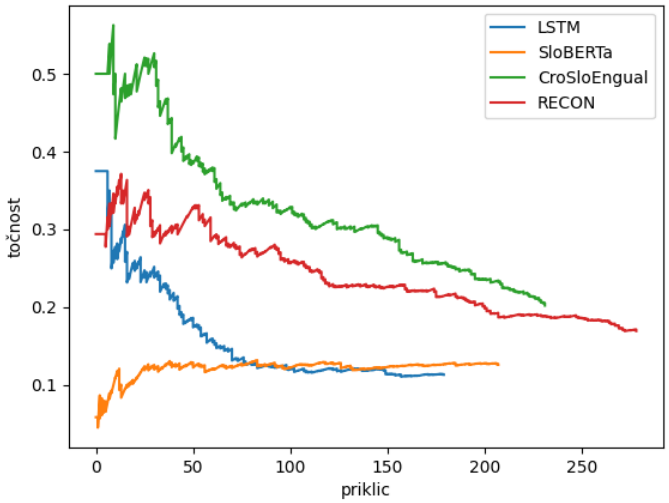
Na sliki 17 imamo prikazane krivulje točnosti v odvisnosti od priklica na testni množici člankov 24ur.com. Ta testna množica nima označenih relacij in zato ne poznamo števila vseh pravilnih relacij. Za priklic smo zato uporabili število najdenih pravilnih relacij in pri izračunu točnosti izvzeli primere, ko je model napovedal prazno relacijo P0. Po priklicu in točnosti se je najslabše odrezala metoda LSTM. Malce boljše rezultate je dosegla metoda z vložitvami SloBERTa. Izmed vseh metod je najvišjo točnost dosegla metoda z vložitvami CroSloEngual in najvišji priklic metoda RECON.

V tabeli 3 imamo za vsak model zapisane vrednosti priklica, točnosti in normaliziranega diskontiranega kumulativnega prispevka za vsako relacijo na testni množici 24ur.com. Na tej množici je metoda z vložitvami SloBERTa zopet napovedovala najmanj različnih relacij. Iz tabele vidimo, da je je ta metoda v večini primerov pravilno napovedovala le relacijo P527(vsebuje), P361(je del) in P31(primerek od). Poleg tega je pri teh treh relacijah dosegla majhno točnost. Metoda LSTM je napovedovala bolj raznolike relacije, je pa imela pri vseh relacijah nizko točnost. Izmed vseh modelov je najbolje napovedovala samo relacijo P50(avtor). Metoda z vložitvami CroSloEngual dosegla najvišjo skupno točnost 20%. Od vseh metod je najbolj točno napovedala relacije P22(oče), P31(primerek od), P39(uradni položaj), P131(se nahaja v), P150(administrativne podenote), P172(etnična skupina), P276(kraj), P361(je del), P463(član), P527(vsebuje), P641(šport) in P1001(spada pod upravo). Metoda z vložitvami CroSloEngual je imela tudi najvišji normaliziran diskontirani kumulativni prispevek kar pomeni, da so imele pravilne relacije v večini primerov višjo oceno samozavesti kot napačni. Metoda RECON je dosegla za 3% manjšo točnost kot metoda z vložitvami CroSloEngual, imela pa je največ pravilnih napovedi relacij. Pri priklicu vidimo, da je metoda RECON pogosto napovedovala relacije P17(država), P131(se nahaja v), P276(kraj), P361(je del) in P527(vsebuje).

	LSTM			SloBERTa			CroSloEngual			RECON		
Relacija	T	P	KP	T	P	KP	T	P	KP	T	P	KP
Vse	82%	82%	0.99	39%	39%	0.93	76%	76%	0.97	**89%**	**89%**	**1.0**
P0	75%	80%	0.98	43%	59%	0.93	59%	**83%**	0.97	**83%**	79%	**0.99**
P17	79%	75%	0.98	33%	38%	0.92	59%	80%	0.95	**86%**	**95%**	**0.99**
P19	88%	95%	**0.99**	90%	82%	0.98	**94%**	**97%**	**0.99**	93%	92%	**0.99**
P20	92%	79%	**0.99**	87%	69%	0.98	**95%**	87%	**0.99**	88%	**88%**	**0.99**
P22	53%	24%	0.86	65%	8%	0.92	**80%**	**75%**	**0.95**	39%	47%	0.83
P25	20%	5%	0.45	/			**63%**	**71%**	**0.87**	23%	18%	0.62
P27	60%	67%	0.95	/			84%	61%	0.97	**91%**	**95%**	**0.99**
P31	85%	82%	**0.99**	26%	85%	0.93	87%	93%	**0.99**	**93%**	**95%**	**0.99**
P39	78%	63%	0.97	/			84%	**91%**	0.97	**86%**	**91%**	**0.98**
P40	39%	50%	0.83	41%	5%	0.85	**67%**	**75%**	**0.94**	46%	40%	0.87
P50	68%	60%	0.95	/			**91%**	77%	**0.99**	90%	**84%**	0.98
P106	92%	92%	0.99	84%	69%	0.98	97%	91%	**1.0**	**99%**	**99%**	**1.0**
P131	86%	91%	**0.99**	74%	14%	0.95	84%	57%	0.98	**94%**	**94%**	**0.99**
P136	70%	62%	0.96	/			86%	88%	**0.98**	**88%**	**93%**	**0.98**
P138	84%	73%	**0.99**	**100%**	0%	0.0	56%	45%	0.95	84%	**80%**	0.98
P140	91%	79%	**0.99**	/			**94%**	49%	**0.99**	93%	**94%**	**0.99**
P150	**96%**	94%	**1.0**	95%	7%	0.99	92%	61%	0.99	**96%**	**98%**	**1.0**
P155	79%	80%	0.98	**86%**	19%	0.98	82%	**86%**	**0.99**	83%	81%	0.98
P156	78%	82%	**0.98**	84%	22%	0.97	82%	**89%**	**0.98**	**86%**	75%	**0.98**
P172	85%	84%	**0.99**	/			76%	67%	0.95	**90%**	**96%**	**0.99**
P276	**81%**	66%	**0.98**	45%	24%	0.88	61%	**75%**	0.95	80%	**75%**	0.97
P279	32%	35%	0.83	16%	2%	0.5	**51%**	39%	**0.89**	48%	**57%**	0.88
P361	74%	70%	**0.98**	22%	25%	0.9	68%	60%	0.97	**79%**	**80%**	**0.98**
P463	55%	44%	0.92	/			**94%**	40%	**0.98**	71%	**79%**	0.96
P527	81%	81%	**0.99**	18%	76%	0.92	73%	82%	0.98	**84%**	**89%**	**0.99**
P607	92%	90%	0.99	74%	3%	0.9	**97%**	**98%**	**1.0**	95%	94%	0.99
P641	91%	84%	0.99	96%	42%	**1.0**	74%	89%	0.98	**98%**	**98%**	**1.0**
P1001	96%	86%	**1.0**	/			96%	91%	**1.0**	**98%**	**92%**	**1.0**
P3373	73%	57%	0.95	31%	9%	0.76	82%	**63%**	**0.97**	**84%**	53%	0.96
P3450	90%	92%	0.99	77%	71%	0.97	**98%**	97%	**1.0**	**98%**	**98%**	**1.0**

Tabela 2: Tabela točnosti, priklica in normaliziranega diskontiranega kumulativnega prispevka za ekstrakcijo relacij na testni množici Wikipedije.

Pri teh relacijah z izjemo P17 je dosegla nižjo točnost kot pa metoda z vložitvami CroSloEngual.



Slika 17: Krivulje točnosti v odvisnosti od priklica za modele na testni množici člankov 24ur.com.

	LSTM			SloBERTa			CroSloEngual			RECON		
Relacija	T	P	KP	T	P	KP	T	P	KP	T	P	KP
Vse	11%	179	0.68	13%	207	0.79	**20%**	231	**0.97**	17%	**278**	0.88
P17	23%	30	0.7	5%	2	0.3	26%	20	0.78	**43%**	**45**	**0.83**
P19	10%	**1**	**1.0**	**100%**	**1**	0.0	20%	**1**	**1.0**	0%	0	0.0
P20	**0%**	**0**	**0.0**	/			**0%**	**0**	**0.0**	**0%**	**0**	**0.0**
P22	0%	0	0.0	/			**38%**	**3**	**0.68**	0%	0	0.0
P25	/			/			45%	**5**	0.72	**80%**	4	**0.9**
P27	10%	**19**	0.7	/			13%	9	0.68	**36%**	5	**0.9**
P31	7%	6	0.46	3%	**16**	0.35	**13%**	9	**0.48**	10%	3	0.36
P39	18%	2	0.43	/			**31%**	**4**	**0.87**	11%	1	0.41
P40	24%	4	0.73	/			47%	**7**	**0.87**	**67%**	2	0.81
P50	**13%**	**9**	**0.48**	/			9%	7	0.38	6%	5	0.47
P106	9%	**7**	0.38	**33%**	1	**1.0**	9%	1	0.38	0%	0	0.0
P131	19%	35	0.69	46%	6	0.69	**51%**	22	**0.89**	22%	**38**	0.69
P136	0%	0	0.0	0%	0	0.0	9%	**1**	**1.0**	**50%**	**1**	**1.0**
P138	0%	0	0.0	0%	0	0.0	**2%**	**1**	0.24	1%	**1**	**1.0**
P140	**0%**	**0**	**0.0**	/			/			/		
P150	6%	3	0.33	/			**33%**	3	**1.0**	7%	**5**	0.35
P155	**1%**	**1**	**0.18**	0%	0	0.0	0%	0	0.0	0%	0	0.0
P156	**1%**	**1**	0.18	**1%**	**1**	**0.21**	0%	0	0.0	0%	0	0.0
P172	16%	4	0.41	0%	0	0.0	**52%**	**12**	**0.85**	27%	3	0.47
P276	26%	5	0.57	/			**48%**	25	**0.8**	26%	**32**	0.66
P279	0%	0	0.0	**2%**	**2**	0.23	0%	0	0.0	**2%**	**2**	**0.24**
P361	18%	14	0.58	23%	**72**	0.67	**37%**	24	**0.82**	16%	39	0.65
P463	33%	9	0.67	17%	1	0.46	**62%**	**21**	**0.88**	28%	9	0.66
P527	20%	24	0.59	18%	**103**	0.64	**30%**	38	**0.76**	17%	81	0.67
P607	**0%**	**0**	**0.0**	/			**0%**	**0**	**0.0**	**0%**	**0**	**0.0**
P641	9%	2	0.35	/			**25%**	**6**	**0.55**	/		
P1001	11%	1	0.41	/			**30%**	**3**	**0.83**	25%	1	0.52
P3373	6%	2	0.32	17%	2	0.71	5%	**9**	0.35	**50%**	1	**1.0**
P3450	**0%**	**0**	**0.0**	/			/			**0%**	**0**	**0.0**

Tabela 3: Tabela točnosti, priklica in normaliziranega diskontiranega kumulativnega prispevka za ekstrakcijo relacij na testni množici 24ur.com.

	Wikipedija učna množica	**Wikipedija testna množica**	**24ur.com testna množica**
LSTM	**95%**	82%	11%
SloBERTa	40%	39%	13%
CroSloEngual	78%	76%	**20%**
RECON	94%	**89%**	17%

Tabela 4: Tabela točnosti modelov za učne in testne množice.

6 Diskusija

V tem poglavju ovrednotimo rezultate naših metod in podatkov v korpusih, ki smo jih zgradili v sklopu našega dela.V podpoglavju 6.1 ocenimo pravilnost označenih relacij v našem učnem korpusu, ki smo ga zgradili iz besedil Wikipedije s pomočjo baze znanja WikiData po vzoru članka [20], iz podpoglavja 4.1. Temu sledi poglavje 6.2, kjer ocenimo označevanje entitet v našem testnem korpus iz podpoglavja 4.2, ki smo ga zgradili iz člankov 24ur.com. Na koncu pa v podpoglavju 6.3 primerjamo rezultate metod za napovedovanje relacij med seboj in z rezultati, ki jih metode dosežejo za angleški jezik v člankih, kjer so metode predstavili.

6.1 Polavtomatska gradnja korpusa iz besedil Wikipedija s pomočjo baze znanja WikiData

Za izdelavo našega učnega korpusa smo se odločili uporabiti pristop polavtomatske gradnje korpusa. S tem pristopom smo zgradili korpus, ki je bil dovolj velik za učenje metod globokega učenja, vendar je ta korpus vseboval tudi napačne primere. Pri polavtomatskem označevanju relacij smo uporabili le entitete, ki smo jih povezali z entitetami v bazi znanja WikiData. Večino označenih entitet, ki smo jih povezali, so izhajale iz notranjih povezav Wikipedije, ki so nam zagotovile točno označevanje entitet z značkami WikiData entitet. Napake so torej večinoma nastale pri označevanju relacij. Za vsak par entitet, ki smo ju povezali z entitetama na bazi WikiData, smo znotraj stavka preverili, če na bazi WikiData obstaja relacija med entitetama, ki smo ju povezali. Če je relacija obstajala, smo označili, da med entitetama v učnem korpusu nastopa ta relacija. To pomeni, da smo označevali relacije na podlagi splošnega znanja, shranjenega v bazi znanja, in ne glede na vsebino stavka, kar je bil glavni razlog za napačno označene relacije.

6.2 Označevanje entitet na testnem korpusu iz člankov 24ur.com

Na testnih podatkih ni bilo možno uporabiti enakega postopka za označevanja entitet in smo se zato morali zanašati na točnost označevalnika entitet CLASSLA in povezovalnika entitet za bazo znanja WikiData, ki smo ga implementirali sami.

Pri učni množici imamo drugačen nabor primerov entitet, ker smo poleg entitet, pridobljenih s CLASSLA, uporabili notranje povezave Wikipedije, ki so vsebovali razne vsebine (npr. poklice), ki jih označevalnik CLASSLA ne označi. Zato imamo znotraj testne množice malo primerov z določenimi relacijami (npr. P106(poklic)). Poleg tega označevalnik CLASSLA kdaj označi eno entiteto v več delih. V testnih primerih je označil državo Bosna in Hercegovina kot entiteto Bosna in entiteto Hercegovina. Metode nato med tema dvema entitetama zaznajo relacije, ki so napačne, ker sta ti dve entiteti ena sama. Včasih pa se zgodi, da se entiteta označi enkrat in je označen le del entitete. Takšne primere smo med evalvacijo metod obravnavali kot da je bila entiteta označena pravilno in je bila relacija pravilna, če je celotna entiteta nastopala v tej relaciji.

Drugi del pri označevanja entitet je bil povezovanje označenih entitet na bazo znanja WikiData. Ta korak je pomemben samo za metodo RECON, ki za napovedovanje relacij potrebuje dodatni kontekst entitete, ki ga pridobi iz baze znanja. Metodi Opentapioca [4] in Falcon 2.0 [18] sta obe odkrili ujemanje pri manj kot 25% primerov, naša metoda pa je označila kar 67% primerov. Poleg tega smo do neke mere zagotovili vsebinsko podobnost pri napačnih povezavah povezane entitete v bazi znanja in entitete v besedilu s pogojem največje možne kosinusne razdalje vektorskih vložitev. Smo pa opazili pomanjkljivost baze Elasticsearch pri iskanju kandidatov za povezovanje entitete. Elasticsearch določenih entitet, ki so shranjene v bazi ne najde niti, če v iskanje vnesemo niz, ki je identičen nizu entitete (npr. entiteta Vuhred). Ta problem delno razrešimo z uporabo API klica WikiData, ki za popolna ujemanja niza najde pravilno entiteto.

Problem drugače označenih entitet smo želeli delno razrešit z uporabo požrešnega pristopa povezovanja entitet, ki je opisan na koncu podpoglavja 4.2.2. S tem pristopom pridobimo več različnih vsebin v entitetah (npr. poklici) vendar pa povečamo problem delno označenih entitet in problem entitet, ki so označene po več delih. Testni korpus s požrešnim pristopom se ni obnesel med testiranjem metod in ga zato nismo uporabili za evalvacijo metod.

6.3 Metode za napovedovanje relacij

Na podatkih, ki smo jih pridobili, smo učili in testirali tri metode, ki temeljijo na različnih pristopih. Metodo s povratno nevronsko mrežo z dolgim kratkoročnim spominom (mrežo LSTM), iz članka [20] smo izbrali, ker smo se pri generaciji učnih podatkov zgledovali po članku in lahko zato lažje primerjamo naše rezultate z njihovimi zaradi uporabe podobnih učnih podatkov. Metoda iz članka [26] uporabi vložitve BERT za napovedovanje relacij. Metodo smo uporabili, ker nas je zanimalo kako dobro lahko napovemo relacije s podatki, shranjenimi v vložitvah transformerja BERT. Metodo smo testirali za slovenske vložitve SloBERTa in večjezične vložitve CroSloEngual. Metodo RECON iz članka [1] smo uporabili, ker uporablja trenutno najbolj sodobne pristope za napovedovanje relacij. Poleg tega pa uporabi isti korpus kot prej omenjena metoda LSTM, torej je bila tudi ta metoda narejena s korpusom, ki je podoben našemu.

6.3.1 Primerjava rezultatov metod za slovenski jezik z rezultati za angleški jezik

Metoda LSTM na testnih podatkih Wikipedije doseže podobno krivuljo točnosti v odvisnosti s priklicem kot so v avtorskem članku [20] za angleški jezik. V članku so za angleški jezik dosegli skupno točnost 82% tako kot smo mi za slovenski jezik.

Nekoliko boljše rezultate dosežemo z metodo RECON [1], ki na naših testnih podatkih Wikipedije doseže točnost 89% in na angleških podatkih v avtorskem članku doseže okoli točnost 87%. Razlog za boljši rezultat je morda proces filtriranja povezav, ki smo ga opisali v podpoglavju 4.1.4.

Slabše rezultate pa doseže metoda z vložitvami BERT [26]. Rezultate sicer bolj težko primerjamo, ker je bila metoda za angleški jezik testirana na podatkih osme naloge iz SemEval-2010. Ti podatki so bili ročno označeni in manj obsežni kot

naš učni korpus. Metoda se je pri uporabi vložitev SloBERTa pretirano prilegla določenim relacijam in je zato slabo napovedovala preostale relacije. Z vložitvami CroSloEngual pa je model dosegel rezultate, ki so bili primerljivi z modelom metode RECON ter modelom metode LSTM.

6.3.2 Primerjava rezultatov metod za slovenski jezik

Za napovedovanje relacij smo uporabili tri metode. Vse tri smo naučili na isti učni množici iz besedil Wikipedije in jih testirali na testni množici iz besedil Wikipedije in testni množici člankov 24ur.com. Dve različni testni množici smo uporabili, da smo lahko ugotovili, kako dobro delujejo metode na podatkih, ki so podobni ali drugačni od učne množice.

Na testni množici besedil Wikipedije se je najbolje odrezala metoda RECON, ki doseže skupno točnost 89%, sledi ji metoda z mrežo LSTM, ki na množici doseže skupno točnost 82%. Malce slabše se odreže metoda z vložitvami CroSloEngual, ki doseže skupno točnost 76%. Razlog za malce nižjo točnost je bolj pogosta napoved praznih relacij. Veliko slabše se odreže metoda z vložitvami SloBERTa, ki se pretirano prilagodi parim relacijam.

Na testni množici člankov 24ur.com se je najbolje odrezala metoda z vložitvami CroSloEngual s skupno točnostjo 20%, sledi ji metoda RECON z 17%, metoda z vložitvami SloBERTa z 13% in nato metoda z mrežo LSTM, ki doseže skupno točnost 11%.

Metode dosežejo nižjo skupno točnost na testni množici člankov 24ur.com kot na testni množici besedil Wikipedije. Razlog za napake v testni množici člankov 24ur.com je v drugačnem postopku označevanja entitet, kar smo že omenili v podpoglavju 6.2. Poleg tega v člankih 24ur.com metode napovejo relacije za vse možne kombinacije entitet v stavku. Iz tega sledi, da ima testni korpus veliko primerov praznih relacij, ki jih modeli pomotoma označijo z neprazno relacijo. Problem smo do neke mere rešili z dodajanjem praznih relacij v učni korpus v upanju, da bodo modeli bolje zaznali prazne povezave.

Iz rezultatov na testni množici Wikipedije lahko sklepamo, da se modeli na učnih podatkih dobro naučijo napovedovati relacije za podobna besedila. Za boljše rezultate na člankih 24ur.com bi pa verjetno potrebovali učne podatke, ki so po stilu pisanja bolj podobni.

6.3.3 Primerjava rezultatov metod z drugačnim pristopom ekstrakcije relacij

Članek [6] predstavi pristop, ki s pomočjo orodja *google translate* uporabi model ekstrakcije relacij za angleški jezik, da napove relacije za poljuben tuji jezik, ki ga orodje *google translate* podpira. Izbrali so si angleški model Ollie [19], ki se iz ročno označenega korpusa nauči vzorcev drevesa skladenjskega razčlenjevalnika za stavke in iz tega ugotovi v katerem delu stavka je zapisana relacija. Metoda torej nima vnaprej določenih relacij, ki jih napoveduje, ampak le izvleče del stavka, kjer se nahaja relacija. Pristopu, ki ga uporabi model Ollie, pravimo odprta ekstrakcija informacij (angl. open information extraction). Model so v [6] testirali na treh jezikih tako, da so ročno označili 1.000 relacij iz besedil Wikipedije za vsak jezik in primerjali

ročno označene relacije z napovedmi modela. Ollie je tako za francoski jezik dosegel točnost 81.6%, za hindijščino točnost 64.9% in za ruščino točnost 63.5%. S takim pristopom bi verjetno dosegli višjo točnost na našem testnem korpusu člankov 24ur.com. Žal pa zaradi različnega pristopa takšen model ne bi zaznal relacij, ki niso direktno omenjene v besedilu. Na primer relacija sorojenec med entiteto *Anžeta* in entiteto *Maše* v stavku:

Ožbej je oče od Maše in Anžeta.

7 Zaključek

V našem delu smo predstavili problem ekstrakcije relacij in naučili metode, ki napovedujejo relacije za slovenski jezik. V uvodu smo predstavili problem ekstrakcije relacij in uporabnost ekstrakcije relacij za procesiranje naravnega jezika ter tekstovno rudarjenje.

Nato smo opravili kratek pregled področja in se osredotočili na sodobne pristope z globokim učenjem.

Za ekstrakcijo relacij za slovenski jezik smo se odločili, da bomo uporabili tri metode, ki uporabijo različne pristope globokega učenja. Enega, ki uporabi povratno nevronsko mrežo z dolgim kratkoročnim spominom, drugega, ki uporabi vektorske vložitve BERT in še metodo RECON, ki uporabi grafovske nevronske mreže, ki v zadnjih časih dosegajo najboljše rezultate na področju ekstrakcije relacij.

Poleg metode za učenje modela za ekstrakcijo relacij potrebujemo še učne in testne podatke. Za učni korpus uporabimo pristop polavtomatske gradnje. Na besedilih slovenske Wikipedije označimo relacije s pomočjo baze znanja WikiData. Ta postopek omogoča hitro označevanje velikega števila relacij, vendar pa nam ta postopek prinese tudi nekaj napačno označenih relacij. Iz našega korpusa lahko nekaj napačno označenih relacij odstranimo s filtriranjem korpusa. Za preostale napake pa upamo, da nimajo prevelikega vpliva pri učenju naših modelov. Za testiranje na manj podobnih podatkih uporabimo še članke 24ur.com na katerih označimo entitete in ročno preverimo pravilnost relacij, ki jih jim dodelijo naučeni modeli izbranih metod.

Učni korpus zgrajen iz besedil iz Wikipedije, razdelimo na učno in testno množico. Tri metode učimo na učni množici in jih nato testiramo na testni množici učnega korpusa in člankih 24ur.com z označenimi entitetami. Metoda RECON doseže najvišjo skupno točnost na testni množici Wikipedije. Metoda BERT z vložitvami CroSloEngual pa doseže najvišjo skupno točnost na testni množici člankov 24ur.com.

Metoda RECON, metoda s povratno nevronsko mrežo z dolgim kratkoročnim spominom ter metoda BERT z vložitvami CroSloEngual so se dobro prilagodile učnim podatkom in testni množici iz besedil Wikipedije. Na testni množici iz člankov 24ur.com so pa dosežejo slabše rezultate. Razlog za to je v drugačnem pristopu označevanja entitet v učnem korpusu in člankih 24ur.com in drugačnem slogu pisanja. Če bi želeli izboljšati rezultate metod, ki smo jih uporabili, bi potrebovali učni korpus, ki bi vseboval tudi besedila izven Wikipedije in bi imel vse entitete označene z označevalnikom, ki bi ga uporabili tudi za testiranje. To lahko dosežemo z ročnim označevanjem velikega učnega korpusa, kar bi zahtevalo veliko časa in financiranja. Druga možnost bi zahtevala točen in splošen postopek povezovanja entitet, ki bi jih v besedilu označili. Povezovanje entitet, ki smo ga uporabili učnem korpusu, je bil sicer natančen, vendar je zahteval uporabo notranjih povezav Wikipedije in smo ga zato lahko uporabili le na besedilih Wikipedije. Za članke 24ur.com pa smo uporabili splošen postopek, ki pa ni dosegel visoke točnosti in bi zato uvedel veliko dodatnih napačno označenih relacij.

Literatura

[1] A. Bastos in dr., *Recon: Relation extraction using knowledge graph context in a graph neural network*, v: Proceedings of the Web Conference 2021, 2021, str. 1673–1685.

[2] R. Bunescu in R. J. Mooney, *Subsequence kernels for relation extraction*, v: NIPS, Citeseer, 2005, str. 171–178.

[3] A. Culotta in J. Sorensen, *Dependency tree kernels for relation extraction*, v: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), 2004, str. 423–429.

[4] A. Delpeuch, *Opentapioca: Lightweight entity linking for wikidata*, arXiv preprint arXiv:1904.09131 (2019).

[5] J. Devlin in dr., *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).

[6] M. Faruqui in S. Kumar, *Multilingual open relation extraction using cross-lingual projection*, arXiv preprint arXiv:1503.06450 (2015).

[7] T. Hasegawa, S. Sekine in R. Grishman, *Discovering relations among named entities from large corpora*, v: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), 2004, str. 415–422.

[8] A. Hogan in dr., *Knowledge graphs*, ACM Computing Surveys (CSUR) **54**(4) (2021) 1–37.

[9] A. Joulin in dr., *Bag of tricks for efficient text classification*, v: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Association for Computational Linguistics, 2017, str. 427–431.

[10] N. Kambhatla, *Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction*, v: Proceedings of the ACL Interactive Poster and Demonstration Sessions, 2004, str. 178–181.

[11] L. P. Lebedev in M. J. Cloud, *Introduction to Mathematical Elasticity*, World Scientific, Singapur, 2009.

[12] N. Ljubešić in K. Dobrovoljc, *What does neural bring? analysing improvements in morphosyntactic annotation and lemmatisation of Slovenian, Croatian and Serbian*, v: Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing, Association for Computational Linguistics, Florence, Italy, 2019, str. 29–34, doi: 10.18653/v1/W19-3704.

[13] A. Moro, A. Raganato in R. Navigli, *Entity linking meets word sense disambiguation: a unified approach*, Transactions of the Association for Computational Linguistics **2** (2014) 231–244.

[14] D. Nathani in dr., *Learning attention-based embeddings for relation prediction in knowledge graphs*, arXiv preprint arXiv:1906.01195 (2019).

[15] D. P. Nguyen, Y. Matsuo in M. Ishizuka, *Relation extraction from wikipedia using subtree mining*, v: Proceedings of the National Conference on Artificial Intelligence, **22**, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, str. 1414.

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

[17] S. Riedel, L. Yao in A. McCallum, *Modeling relations and their mentions without labeled text*, v: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2010, str. 148–163.

[18] A. Sakor in dr., *Falcon 2.0: An entity and relation linking tool over wikidata*, v: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, str. 3141–3148.

[19] M. Schmitz in dr., *Open language learning for information extraction*, v: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, 2012, str. 523–534.

[20] D. Sorokin in I. Gurevych, *Context-aware representations for knowledge base relation extraction*, v: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, str. 1784–1789.

[21] J. Strötgen in M. Gertz, *Multilingual and cross-domain temporal tagging*, Language Resources and Evaluation **47**(2) (2013) 269–298.

[22] M. Ulčar in M. Robnik-Šikonja, *CroSloEngual BERT 1.1*, 2020, slovenian language resource repository CLARIN.SI.

[23] M. Ulčar in M. Robnik-Šikonja, *Sloberta: Slovene monolingual large pretrained masked language model* (2021).

[24] A. Vaswani in dr., *Attention is all you need*, v: Advances in neural information processing systems, 2017, str. 5998–6008.

[25] P. Veličković in dr., *Graph attention networks*, arXiv preprint arXiv:1710.10903 (2017).

[26] S. Wu in Y. He, *Enriching pre-trained language model with entity information for relation classification*, v: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, str. 2361–2364.

[27] D. Zeng in dr., *Relation classification via convolutional deep neural network*, v: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, str. 2335–2344.

[28] D. Zhang in D. Wang, *Relation classification via recurrent neural network*, arXiv preprint arXiv:1508.01006 (2015).

[29] G. Zhou in dr., *Exploring various knowledge in relation extraction*, v: Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05), 2005, str. 427–434.

[30] H. Zhu in dr., *Graph neural networks with generated parameters for relation extraction*, arXiv preprint arXiv:1902.00756 (2019).