SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Domagoj Šebalj

ANALIZA TEKSTNIH DOKUMENATA NA HRVATSKOM JEZIKU KORIŠTENJEM METODA DUBOKOG UČENJA

DIPLOMSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Domagoj Šebalj

Matični broj: 44856/16–R

Studij: Informacijsko i programsko inženjerstvo

ANALIZA TEKSTNIH DOKUMENATA NA HRVATSKOM JEZIKU KORIŠTENJEM METODA DUBOKOG UČENJA

DIPLOMSKI RAD

Mentorica :

Prof. dr. sc. Jasminka Dobša

Varaždin, rujan 2022.

Domagoj Šebalj

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Najbolji predtrenirani modeli u području obrade prirodnog jezika dolaze od nekolicine velikih tvrtki koje nemaju uvijek interes ili motivaciju za prilagođavanje modela za slabo zastupljene jezike. Tako je cilj ovoga rada prikazati način treniranja vlastitog predtreniranog modela na već postojećoj GPT-2 arhitekturi korištenjem slobodno dostupnih ili jeftinih alata, biblioteka i programskog okružja, a kao teorijska podloga je dan pregled povijesnog razvoja predtreniranih modela, njihov način rada i vodeće arhitekture. Tehnike kojima se zaobilaze ograničenja alata, biblioteka i programskih okružja je potrebno dodatno razraditi i pronaći optimalni način treniranja, ali kreirani model za hrvatski jezik svejedno pokazuje zadovoljavajuće rezultate na zadacima jezičnog modeliranja, analize sentimenta te strojnog prevođenja.

Ključne riječi: Obrada prirodnog jezika, Duboko učenje, Python, Tensorflow, Transformeri, GPT-2

Sadržaj

1. Uvod 1

2. Pozadina i povezani radovi 2

2.1. Predtrenirani jezični modeli (engl. *Pretrained language models*, PLMs) 2

2.2. Sekvenca u sekvencu (engl. *Sequence to sequence*, Seq2Seq) 4

2.2.1. Rekurentne neuronske mreže (engl. *Recurrent neural network*, RNN) . . 4

2.2.2. RNN enkoder - dekoder 5

2.2.3. Primjer Seq2Seq-a 6

2.2.4. Mehanizam pozornosti (engl. *Attention Mechanism*) 7

2.2.5. Primjer Seq2Seq-a s mehanizmom pozornosti 8

2.3. Transformer 10

2.3.1. Arhitektura 10

2.3.2. Izvedba mehanizma pozornosti 12

2.3.3. Unaprijedna neuronska mreža i rezidualne konekcije 14

2.3.4. Primjena mehanizma pozornosti 14

2.4. Generativni Predtrenirani Transformer (engl. *Generative Pretrained Transformer*, GPT) 15

2.4.1. Arhitektura 15

2.4.2. GPT-2 17

2.4.3. GPT-3 18

2.5. BERT (engl. *Bidirectional Encoder Representations from Transformers*) 19

2.5.1. Predtreniranje 19

2.5.2. Verzije BERT-a za hrvatski jezik 21

2.6. Suvremeni modeli 22

3. Izrada vlastitog predtreniranog modela 24

3.1. Korišteni alati i biblioteke 24

3.2. Podaci za treniranje 25

3.2.1. Pretprocesuiranje 27

3.3. Kreiranje i treniranje modela 27

3.3.1. Enkodiranje parova bajtova (engl. *Byte pair encoding*, BPE) 27

3.3.2. Tokenizacija 29

3.3.3. Kreiranje modela 31

3.3.4. Treniranje 32

4. Evaluacija 36

4.1. Jezično modeliranje 36

4.2. Prevođenje 38
4.3. Analiza sentimenta 40

5. Budući rad 43

6. Zaključak 44

Popis literature 50

Popis slika 51

Popis tablica 52

1. Prilog 1: Tablični prikaz vrijednosti funkcije gubitka po skupovima podataka i epohama 53

1. Uvod

Jezični neuronski modeli od začetka nastoje riješiti probleme automatizacije povezane s prirodnim, odnosno govornim, jezikom. Na napredak koji su u 1990-ima i 2000-ima postigle statističke i metode bazirane na pravilima prvo nadograđuju duboke neuronske mreže. Tako je prvi jezični model temeljen na neuronskim mrežama predstavljen 2001. godine u [1]. 2000-e su započele digitalizaciju društva kao i usvajanje svakodnevnog korištenja interneta koje je uzrokovalo generiranje velikih količina podataka. To, uz naravno veliki rast računalnih mogućnosti hardvera, omogućilo je zatim razvoj dubokih neuronskih mreža (engl. *Deep neural networks*) poput konvolucijskih neuronskih mreža (engl. *Convolutional neural networks*, CNNs) [2] [3] [4] ili rekurentnih neuronskih mreža (engl. *Recurrent neural networks*, RNNs) [5] [6] [7] na dosad neviđenoj razini.

Usporedno s time se pojavljuje i ideja o učenju jedne mreže za više zadataka (engl. *Multi-task learning*) koje je prvo, još 1993., godine predložio Caruana u [8], a tek su Collobert i Weston 2008. godine primjenili tu ideju za modele u području obrade prirodnog jezika (engl. *Natural language processing*, NLP) [9] [10]. Njihova je ideja o dijeljenju matrica riječnih reprezentacija (engl. *Word embedding matrices*), koje omogućuju različitim modelima da surađuju i dijele neko općenito znanje na nižoj razini, utjecala na puno više od samog treniranja modela za više zadataka. Ona je započela ideju o korištenju predtreniranih reprezentacija riječi koje su, uz korištenje s konvolucijskim mrežama, bile u širokoj uporabi do nedavno. Učenje za više zadataka tako je postao standard prilikom treniranja novih jezičnih modela. Sljedeći korak je došao 2014. godine pojavom koncepta transformiranja rečenice u rečenicu (engl. *Sequence to Sequence*, Seq2Seq) [5]. Ovaj je koncept umjesto riječi uzimao u obzir čitave rečenice, odnosno sekvence te je uskoro nadograđen uvođenjem mehanizma pozornosti (engl. *Attention mechanism* [11] koji je mogao bolje modelirati kontekst tih rečenica ili sekvenci.

Sav je taj napredak doveo do razvoja suvremenih predtreniranih modela (engl. *Pretrained models*) koji će biti fokus ovoga rada, a biti će i dan pregled svih dostignuća potrebnih za njihovo razumijevanje. Ti suvremeni modeli neovisni o kontekstu se treniraju za velik broj zadataka, za neke od njih ostvaruju vrhunske rezultate bez dodatnog prilagođavanja (engl. *zero-shot learning*), a za učenje uspješno koriste velike količine neoznačenih podataka. No njihov je glavni nedostatak potreba za velikom količinom računalnih resursa za predtreniranje pa unatoč činjenici da je većina takvih modela slobodna za korištenje zajednici, njihovo je predtreniranje pod kontrolom nekolicine tehnoloških tvrtki. Slijedom toga, jezici koji nisu jako zastupljeni u globalnom pogledu ostaju izostavljeni jer predtreniranje modela za jezike je, tim tvrtkama, uvjetovano isplativnošću.

U takvom kontekstu postavlja se pitanje: je li uopće moguće izraditi predtrenirani model za na internetu slabo zastupljeni jezik poput hrvatskog jezika koristeći besplatne ili jako jeftine metode? A to dalje povlači pitanje: je li takav model uopće prihvatljiv za rješavanje zadataka iz područja NLP-a? Tako će cilj ovoga rada biti pokušaj predtreniranja modela za hrvatski jezik koristeći potpuno besplatne ili jako jeftine alate za njegovu izradu te će taj model biti evaluiran na zadacima za koje postoje skupovi podataka za hrvatski jezik.

2. Pozadina i povezani radovi

Ovo će poglavlje dati pregled razvoja suvremenih predtreniranih modela kako bi se jasno predočio njihov način funkcioniranja. Prvo će riječ biti o samim predtreniranim modelima i konceptima o strojnom učenju koji su prethodili njihovom razvoju. Zatim će biti objašnjeni ključni mehanizmi o kojima direktno ovise suvremeni predtrenirani modeli, a nakon njih će biti predstavljene i temeljne arhitekture koje su osnova gotovo svih najboljih predtreniranih modela.

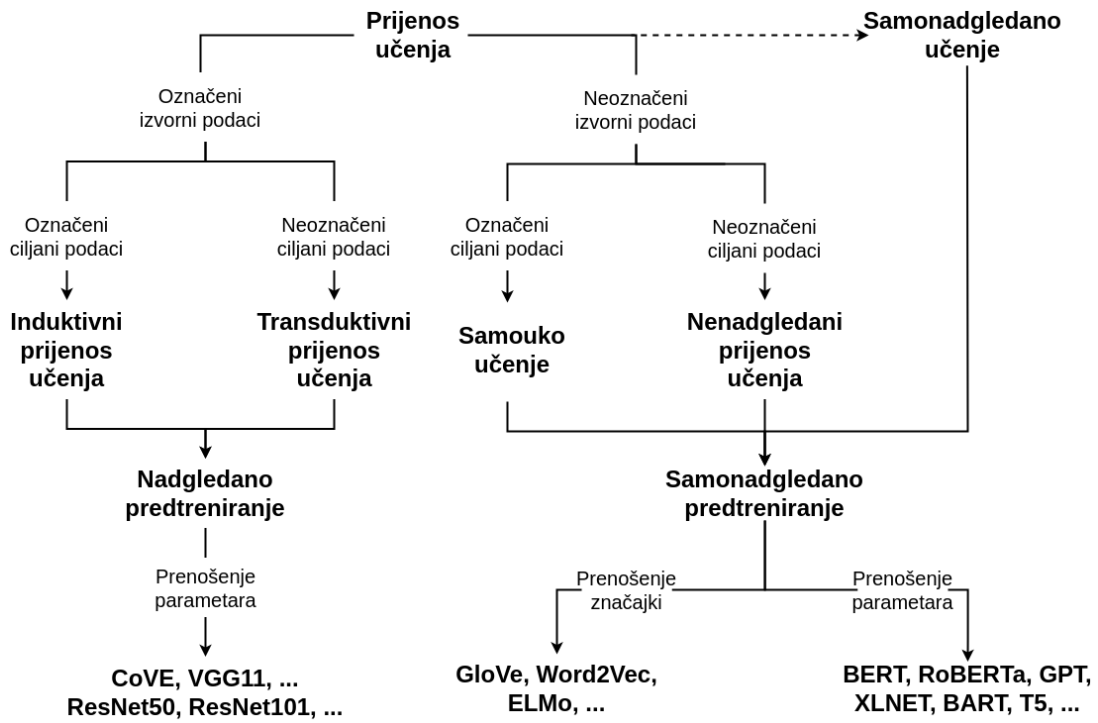
2.1. Predtrenirani jezični modeli (engl. *Pretrained language models*, PLMs)

Razvoj predtreniranih modela započinje pojavom ideje o prijenosu učenja u [12] na što se Han et al. u [13] nadovezuju navodeći da je motivacija te ideje veoma intuitivna, a polazi od jednostavne činjenice da se ljudi mogu oslanjati na prethodno stečeno znanje kako bi lakše riješili novi problem. Odnosno, da je cilj prijenosa učenja obuhvatiti važno znanje iz više izvornih zadataka s ciljem njegove primjene u rješavanju novog zadatka.

Uzimajući takvo intuitivno shvaćanje predtreniranih modela u obzir, razumijemo da jednako kako i osobu koja razumije neki jezik je lakše naučiti rješavati neki problem na tom jeziku, tako je i jezični model koji je treniran na nekom jeziku jednostavnije prilagoditi za rješavanje nekog problema na tom jeziku u odnosu na treniranje iz nule. Stoga je razlog kreiranja takvih modela intuitivno postavljen kao stvaranje crne kutije s razumijevanjem jezika, odnosno imitiranje dobro načitane osobe, sa svrhom rješavanja problema za koji nije dostupan širok spektar podataka za treniranje ili u slučajevima nedostupnosti veće količine računalnih resursa.

Han, Zhang, Ding i dr. u [13] dalje opisuju dva glavna smjera kod prijenosa učenja: prenošenje značajki (engl. *feature transfer*) i prenošenje parametara (engl. *parameter transfer*). Referencirajući [14], [15], [16] i [17] definiraju da metode prijenosa značajki unaprijed uče učinkovite reprezentacije značajki za pred-enkodiranje (engl. *pre-encoding*) znanja kroz različite domene i zadatke, a da ubacivanje tih unaprijed naučenih reprezentacija značajki u ciljane zadatke postiže značajno unaprjeđenje performansi modela. Metode prenošenja parametara u [13] objašnjavaju, pozivajući se na [18], [19], [20] i [21], intuitivnom pretpostavkom da izvorni i ciljani zadaci mogu dijeliti parametre modela ili prijašnje distribucije hiperparametara, a zatim prenijeti znanje finim prilagođavanjem (engl. *fine-tuning*) predtreniranih parametara podacima ciljanog zadatka.

U području obrade prirodnog jezika Han, Zhang, Ding i dr. [13] tumače da se kao ulazi često koriste riječne reprezentacije (engl. *Word embeddings*) poput GloVE-a [22]. Zasada ih uzmimo samo kao primjer prenošenja značajki, a kasnije će biti detaljnije objašnjene. S druge strane, kod prenošenja parametara, predtrenirani modeli kao što su ResNet [23] i VGG11 [24] postižu velike uspjehe u području računalnog vida (engl. *Computer vision*, CV). Taj je uspjeh djelovao poticajno na razvoj nadgledanog (engl. *Supervised*) predtreniranja u području obrade prirodnog jezika od kojih je najreprezentativniji rad zasigurno CoVE [25].



Slika 1: Podjela načina učenja predtreniranih modela (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021)

Na slici 1 jasno je prikazana podjela načina učenja predtreniranih modela. Prvo su razvijeni modeli nadgledanog predtreniranja (engl. *Supervised pre-training*) koje, unatoč brojnim prednostima, ima i veliki nedostatak. Glavni problem takvog načina predtreniranja je potreba za prethodnim anotiranjem podataka koje se često odrađuje ručno. Nenadgledano učenje (engl. *Unsupervised learning*), s druge strane, je u velikoj prednosti kada su u pitanju velike količine nestrukturiranih podataka s kojima se istraživači često susreću. No takvo učenje također ima očite nedostatke kada je u pitanju usmjeravanje modela u željenom smjeru. Stoga, kako bi se odgovorilo na nedostate nadgledanog i nenadgledanog učenja, uveden je pojam samonadgledanog učenja(eng. *Self-supervised learning*) koje samo izvlači znanje iz velikog neoznačenog skupa podataka dok koristi same ulazne podatke za nadgledanje. [13]

Rane verzije predtreniranih jezičnih modela postojale su u obliku već ranije spomenutih riječnih reprezentacija poput Word2Vec[26], GloVE [22] ili ELMo[27]. One su, kako Han, Zhang, Ding i dr. [13] navode, primjenjivale samonadgledane metode za transformaciju riječi u distribuirane reprezentacije. U [28] Turian, Ratinov i Bengio objašnjavaju da se riječne reprezentacije često koriste kao ulazne reprezentacije (engl. *Input embeddings*) i parametri inicijalizacije (engl. *Initialization parameters*) za NLP modele jer hvataju sintaktičke i semantičke informacije u tekstu te time omogućuju značajna poboljšanja u odnosu na nasumičnu inicijalizaciju parametara. A Han, Zhang, Ding i dr. [13] dalje navode da zbog polisemije, odnosno višeznačnosti, nekih riječi, Peters, Neumann, Iyyer i dr. u [29] primjenjuju neuronski model na razini sekvence kako bi uhvatili kompleksne jezične značajke kroz različite lingvističke kontekste i generirali kontekstualno zavisne riječne reprezentacije.

Zadnji je veliki korak u predtreniranim jezičnim modelima postignut predstavljanjem Transformera (engl. *Transformer*) 2017. godine u [30]. Na slici 1 može se uočiti da se tu radi o metodi prenošenja parametara samonadgledanim učenjem što nam govori da, za razliku od modela koji uče nadgledano, bolje iskorištava sve one lako dostupne nestrukturirane podatke, a za razliku od samih riječnih reprezentacija, omogućuje dublje razumijevanje konteksta kao i prilagođavanje specifičnim zadacima. Han, Zhang, Ding i dr. [13] objašnjavaju da, po završetku predtreniranja predtreniranih modela temeljenih na transformerima (engl. *Transformer-based pre-trained models*) na velikom skupu tekstnih podataka, i arhitektura i parametri modela mogu biti korišteni kao polazna točka za neki specifični NLP zadatak, odnosno da se finim prilagođavanjem parametara predtreniranih modela za specifični NLP zadatak postižu konkurentne performanse. Dosada predložene modele, kao što su: XLNET [31], RoBERTa [32], BART [33] i T5 [34], uglavnom su inspirirali BERT [35] i GPT [36] te postižu vrhunske rezultate na skoro svim NLP zadacima.

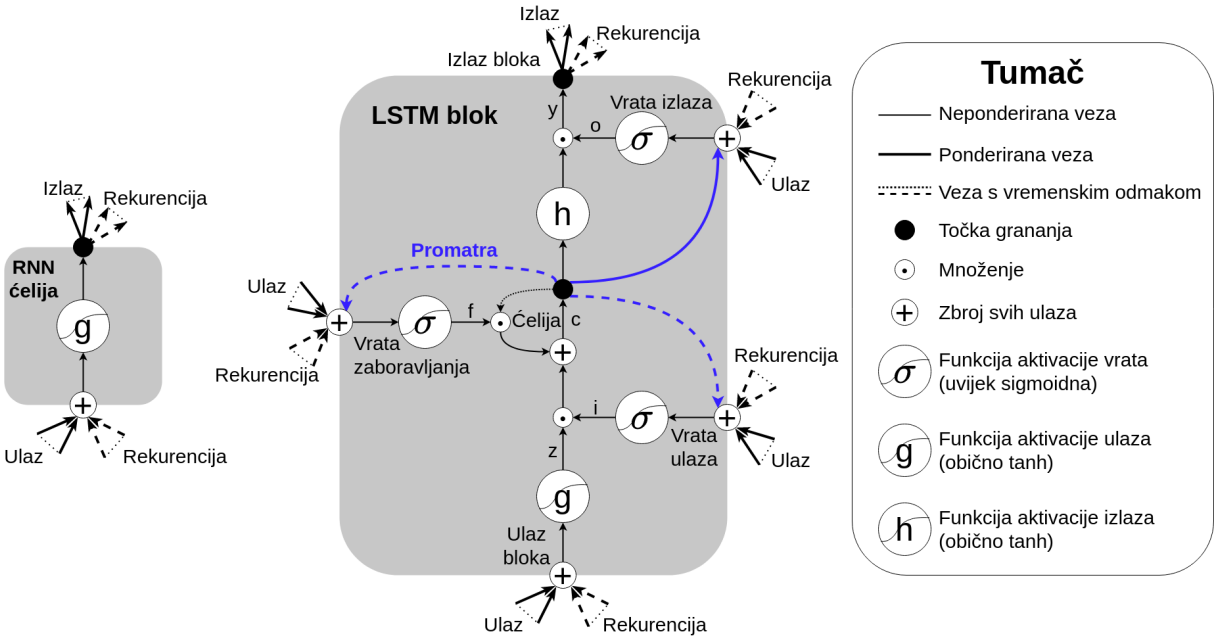
2.2. Sekvenca u sekvencu (engl. *Sequence to sequence*, Seq2Seq)

Kako je prirodni jezik nastao ljudskim djelovanjem, izravno možemo zaključiti da je za strojno razumijevanje jezika najpraktičniji pristup onaj koji imitira ljudsku percepciju jezika. Takva je logika dovela do predstavljanja modela strojnog prevođenja koji umjesto riječi koriste čitave sekvence za prevođenje u [5] i [37] što ima smisla, uzmemo li u obzir da za prevođenje jezika nije dovoljno samo prevesti značenje svake riječ koje često ovisi o kontekstu, nego prenijeti smisao čitave rečenice zadržavajući njenu koheziju. Nedugo nakon tih radova, krajem 2016., i Google translate kao najpoznatiji alat za strojno prevođenje je počeo koristiti takve modele u produkciji.

2.2.1. Rekurentne neuronske mreže (engl. *Recurrent neural network*, RNN)

Polazišna točka razmijevanja Seq2Seq modela su RNN modeli. Minaee, Kalchbrenner, Cambria i dr. u [38] navode da RNN modeli vide tekst kao sekvencu riječi te da su namijenjeni za hvatanje zavisnosti među riječima i tekstualnim strukturama u svrhu klasifikacije teksta. Ali, osnovni RNN ne radi najbolje i često ostvaruje lošije rezultate od unaprijedne neuronske mreže (engl. *feedforward neural network*). Među ostalim varijantama RNN-a, duga kratkoročna memorija (engl. *Long Short-Term Memory*, LSTM) [39] je najpopularnija arhitektura, a dizajnirana je da bolje hvata dugoročne zavisnosti. [38]

Na slici 2 se može vidjeti usporedba arhitekture jednostavne RNN ćelije i LSTM bloka. Važno je primijetiti da RNN ćelija ima takozvani rekurentni (engl. *recurrent*) ulaz i izlaz što znači da nakon obrade jednog dijela sekvence, uz izlazni podatak se generira i takozvano skriveno stanje (engl. *Hidden state*) koje dalje služi kao ulazni podatak za sljedeći dio početne sekvence. Time se osigurava hvatanje dugoročnih zavisnosti. LSTM arhitektura još bolje rješava taj problem što je postignuto kompleksnijom arhitekturom kao što je prikazano na slici 2.



Slika 2: Usporedba RNN i LSTM arhitektura (Vlastita izrada prema: Greff, Srivastava, Koutník i dr., 2016)

2.2.2. RNN enkoder - dekoder

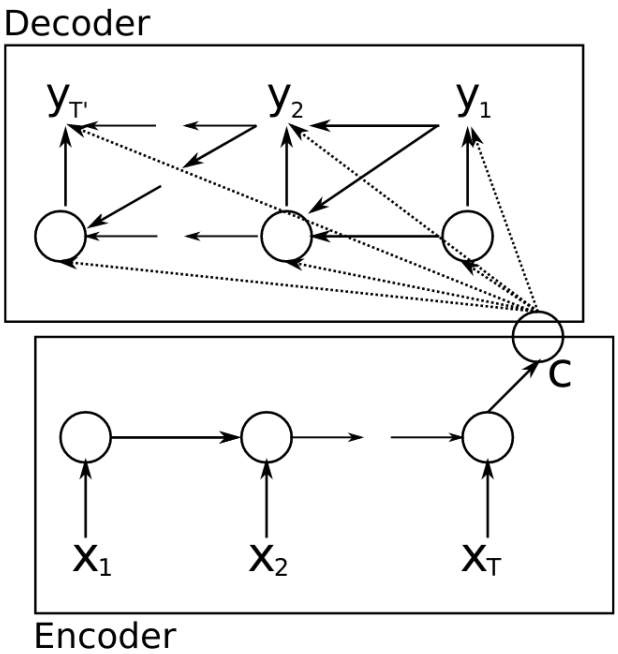
Cho, Van Merriënboer, Gulcehre i dr. su iznijeli ideju o enkoder-dekoder arhitekturi u [37] nadovezujući se na prije spomenute RNN modele. Sutskever, Vinyals i Le komentiraju taj rad u [5] navodeći da je motivacija za enkoder-dekoder arhitekturom došla od problema stvorenih različitim duljinama ulaznih i izlaznih sekvenci koje su pritom kompleksno i nemonotono povezane. Taj je problem u [37] zapravo riješen uvođenjem drugog RNN sloja tako da prvi enkoder sloj enkodira rečenicu na jednom jeziku i tako kreira skriveno stanja (engl. *Hidden state*) koje zatim kao ulazni podatak dolazi dekoderu koji ga pretvara u prevedenu rečenicu u ciljanom jeziku. Na slici 3 se nalazi prikaz enkoder - dekoder arhitekture kakva je zamišljena u [37].

Enkoder je dakle RNN koji sekvencijalno čita svaki simbol ulazne sekvence **x**. Čitanjem svakog simbola, skriveno se stanje RNN mijenja sukladno izrazu:

$$\mathbf{h}_{\langle t \rangle} = f(\mathbf{h}_{\langle t-1 \rangle}, x_t), \tag{2.1}$$

Taj izraz predstavlja RNN koja se sastoji od skrivenog stanja **h** i neobaveznog izraza **y** koji djeluje na sekvenci varijabilne duljine $x = (x_1, \ldots, x_T)$, a f predstavlja nelinearnu funkciju aktivacije. Ta funkcija može biti jednostavna kao što je hiperbolna tangens funkcija (*tanh*), no u posljednje se vrijeme uvriježilo koristiti ranije spomenute sofisticiranije metode kao što su LSTM [39] ili GRU [40]. Dolaskom do kraja sekvence, označenog simbolom za kraj sekvence (engl. *end-of-sequence symbol*, eos simbol), skriveno stanje RNN-a je sažetak (kontekst) **c** čitave ulazne sekvence. [37]

Cho, Van Merriënboer, Gulcehre i dr. [37] dalje definiraju dekoder kao RNN koji je uvježban da, predviđajući sljedeći simbol y_t uz dano skriveno stanje $\mathbf{h}_{\langle t \rangle}$, generira izlaznu sekvencu.



Slika 3: Enkoder - dekoder arhitektura (Izvor: Cho, Van Merriënboer, Gulcehre i dr., 2014)

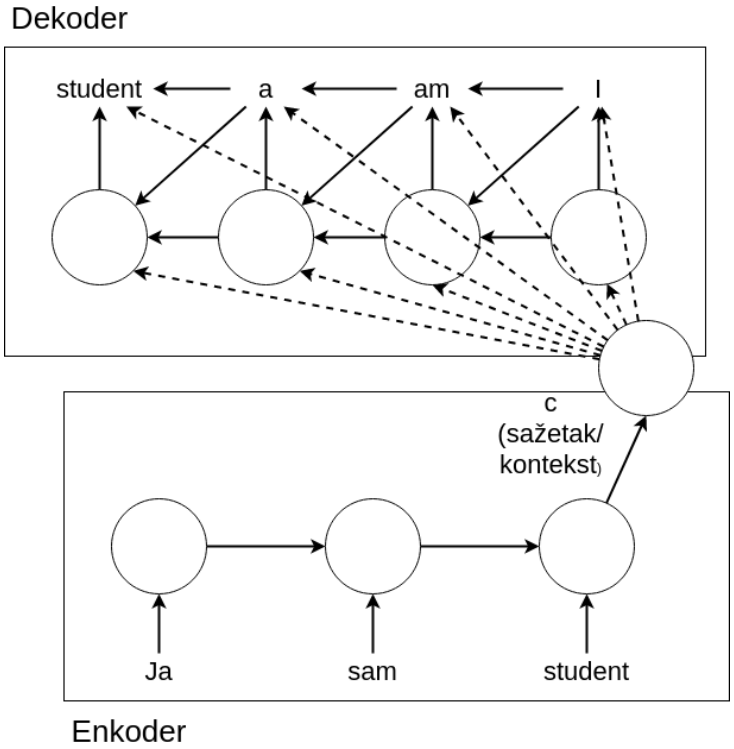
Ali, za razliku od osnovnog RNN-a, i y_t i $\mathbf{h}_{\langle t \rangle}$ ovise o y_{t-1} i o sažetku (kontekstu) **c** ulazne sekvence. Tako je, na kraju, u [37] definirano da se skriveno stanje dekodera u trenutku t računa kao:

$$\mathbf{h}_{\langle t \rangle} = (\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}). \tag{2.2}$$

2.2.3. Primjer Seq2Seq-a

Kako bi se jasnije predočio način rada enkoder - dekoder arhitekture prilikom prevođenja teksta, kreiran je jednostavan primjer po ilustraciji na slici 3. Na slici 4 jasno se može vidjeti kako bi funkcioniralo prevođenje rečenice sa hrvatskog na engleski jezik u 7 vremenskih koraka. U prvome se koraku čita riječ "Ja" te generira stanje h_1. Dalje se čita riječ "sam" i generira stanje h_2, a na kraju riječ "student" i generira stanje h_3. To se zadnje stanje prilikom prosljeđivanja dekoderu smatra kontekstom ili sažetkom rečenice, a u ranijim se matematičkim izrazima definirao kao **c**.

Slijedi 4. korak u kojemu dekoder čita proslijeđeni kontekst, generira prvu riječ "I", odnosno y_1, generira stanje h_4 te ga prosljeđuje dalje. Ranije je već bilo objašnjeno da generiranje riječi ne ovisi samo o kontekstu već i prethodno generiranoj riječi, odnosno y_{t-1}, tako da se prilikom generiranja iduće riječi y_2 uzimaju u obzir prethodna riječ y_1, stanje prethodnog koraka h_4 i kontekst **c** kako je i prikazano na slici 4. To omogućuje da se u šestom koraku generira riječ "a" koja se ne nalazi u izvornoj rečenici, a čak i njena uloga ne postoji u hrvatskom jeziku. Tako se ostvaruje prijevod u duhu jezika što je glavna prednost Seq2Seq modela. Na kraju se još u zadnjem koraku generira zadnje stanja h_7, odnosno zadnja riječ y_4 čime se završava zadaća modela.



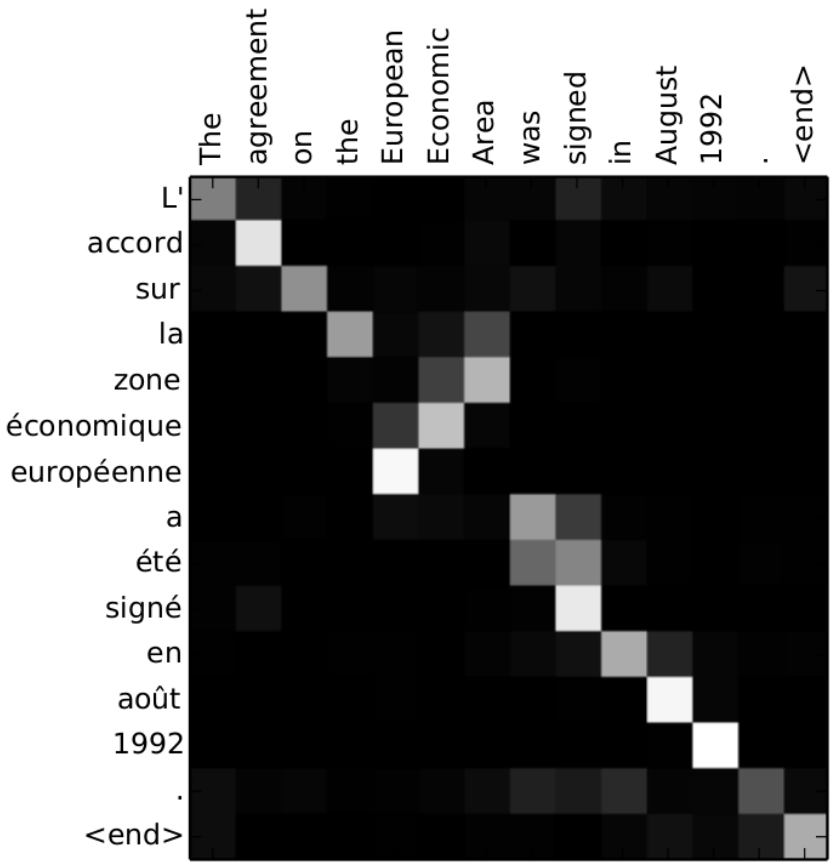
Slika 4: Enkoder - dekoder primjer (Vlastita izrada prema: Cho, Van Merriënboer, Gulcehre i dr., 2014)

2.2.4. Mehanizam pozornosti (engl. *Attention Mechanism*)

Minaee, Kalchbrenner, Cambria i dr. u [38] navode da motivacija za pojavu mehanizma pozornosti proizlazi iz intuitivnog razumijevanja ljudskog korištenja osjeta vida. Odnosno, da je motiviran načinom na koji ljudi obraćaju pažnju na različite dijelove slike ili korelaciju riječi u jednoj rečenici. Uglavnom, pozornost u jezičnim modelima se može interpretirati kao vektor težine značaja. Tako se u slučaju prevođenja, takav vektor koristi da bi se procijenila korelacija pojedinih s ostalim riječima u rečenici.

Mehanizam pozornosti se prvo javio u [11], a zatim proširuje u [41]. Bahdanau, Cho i Bengio u [11] prvo koriste dvosmjerni RNN (engl. *Bidirectional Recurrent Neural Network*, BiRNN) čime ostvaruju, kako je kasnije u [41] to definirano, globalnu pozornost. Globalna pozornost uzima u obzir sve riječi sa strane ulaznog tekstualnog zapisa prilikom generiranja nove riječi izlazne sekvence što ju čini nepraktičnom prilikom prevođenja duljih sekvenci. Za rješavanje toga problema Luong, Pham i Manning u [41] predlažu korištenje mehanizma lokalne pozornosti kojoj je cilj, prilikom generiranja sljedeće riječi izlazne sekvence, voditi računa samo o užem podskupu riječi iz izvorne sekvence. Drugim riječima, mehanizam lokalne pozornosti se selektivno fokusira na manje kontekstualne okvire te time ostvaruje veće brzine i lakše treniranje.

Bahdanau, Cho i Bengio [11] su na slici 5 prikazali kako riječi prevedene sekvence ovise o riječima izvorne sekvence kod prevođenja s engleskog na francuski jezik. Kontekstualna je ovisnost najuočljivija prilikom prevođenja naziva institucije "European Economic Area",



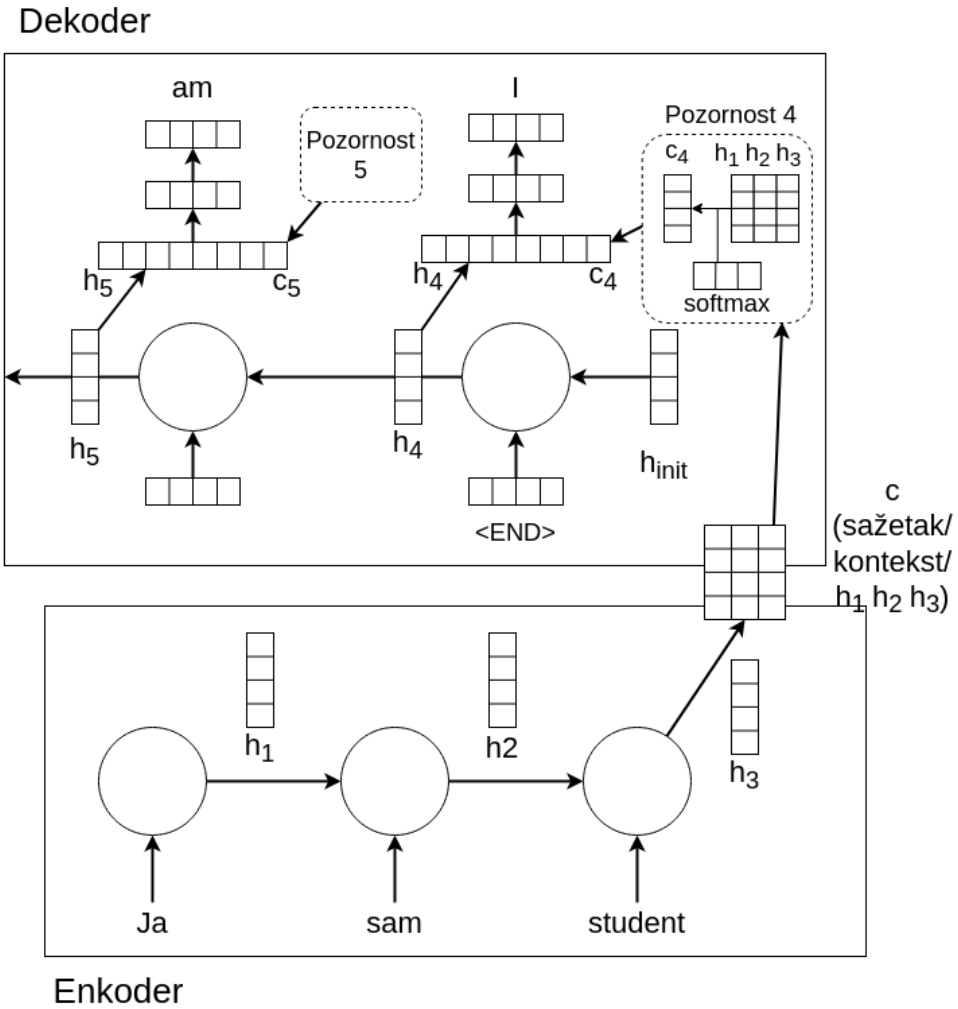
Slika 5: Konteksutalna ovisnost prilikom korištenja mehanizma pozornosti (Izvor: Bahdanau, Cho i Bengio, 2014)

odnosno Europskog Ekonomskog Prostora te se koristi kao čest primjer vizualizacije mehanizma pozornosti upravo zbog obrnutog poretka riječi u nazivima.

2.2.5. Primjer Seq2Seq-a s mehanizmom pozornosti

Mehanizam pozornosti prilikom prevođenja rečenice će biti prikazan pomoću ranije prikazanog primjera na slici 4 kako bi se bolje pokazala razlika njegovog korištenja i ne korištenja. Glavna razlika koju je potrebno odmah na početku uzeti u obzir jest količina podataka koju enkoder šalje dekoderu. U primjeru bez mehanizma pozornosti enkoder dekoderu šalje kontekst, odnosno zadnje skriveno stanje, u obliku jednog vektora. U ovom primjeru, zbog korištenja mehanizma pozornosti iz [11] i [41], dekoderu se dostavljaju sva skrivena stanja koja enkoder generira.

Dekoder s mehanizmom pozornosti, u usporedbi s ranijim primjerom, prima sva skrivena stanja koja enkoder generira, a od kojih je svaki povezan s određenom riječi iz izvorne sekvence. Svako se skriveno stanje boduje te se svaki vektor množi s normaliziranom vrijednošću dodijeljenih bodova. Time se pojačava utjecaj skrivenih stanja s visokim bodovima na izlaznu sekvencu, a smanjuje onih sa nižim. Bodovani se vektori zbrajaju te se time dobiva vektor konteksta za neki vremenski korak dekodera. Na slici 6 je ilustriran način dobivanja vektora konteksta za vremenski korak 4, c_4, u kvadratu označenom nazivom "Pozornost 4".



Slika 6: Seq2Seq primjer s mehanizmom pozornosti (Vlastita izrada prema: Cho, Van Merriënboer, Gulcehre i dr., 2014)

Uz vektor konteksta dekodera, za generiranje riječi potrebno je još, kao što je prikazano na slici 6, dobiti skriveno stanje u tom vremenskom koraku. Na prikazanom se primjeru generira prva riječ "I". Stoga u prvom koraku, odnosno vremenskom koraku 4, dekoder RNN prima reprezentaciju <END> simbola i inicijalno skriveno stanje dekodera, h_{init}. Time se dobiva skriveno stanje h_4, a višak ili y_4 se zanemaruje. Zatim slijedi ranije opisani korak dobivanja vektora konteksta za ovaj vremenski korak, a nakon njega se skriveno stanje, h_4, i dobiveni vektor konteksta, c_4, spajaju kako bi se dobio vektor dimenzije jednakoj zbroju dimenzija skrivenog stanja i vektora konteksta. Takav se spojeni vektor prosljeđuje unaprijednoj (engl. *feedforward*, FFNN) neuronskoj mreži koja je uvježbana zajedno s ovim modelom te čiji izlazni podaci indiciraju riječ generiranu u ovom vremenskom koraku. Na primjeru na slici 6 tako se generira prva riječ "I", a nakon nje se postupak ponavlja za sve vremenske korake dekodera dok se ne dobije ciljana sekvenca.

2.3. Transformer

Ranije su spomenuti modeli bili dobro rješenje za sekvencijalne podatke. Minaee, Kalchbrenner, Cambria i dr. [38] pišu o više različitih pristupa mehanizmu pozornosti koji su se pojavili nakon objave originalnih radova. Pa su tako Yang, Yang, Dyer i dr. [42] predložili mehanizam hijerarhijske mreže pozornosti za klasifikaciju teksta, a Santos, Tan, Xiang i dr. [43] dvosmjerni mehanizam pozornosti poznat i kao *Attentive Pooling*. Svi su ti modeli pridodali razvijanju mehanizma pozornosti i sekvencijalnog pristupa razumijevanju tekstnih podataka. S druge strane, Han, Zhang, Ding i dr. [13] opisuju tu sekvencijalnu prirodu kao problem jer onemogućuje iskorištavanje mogućnosti paralelne obrade koju nudi hardver poput GPU-a i TPU-a.

Rješenje problema neiskorištavanja mogućnosti hardvera dolazi pojavom Transformera. Vaswani, Shazeer, Parmar i dr. [30] opisuju Transformer kao arhitekturu koja izbjegava ponavljanje tipično za RNN modele i umjesto toga se u potpunosti oslanja na mehanizam pozornosti za izvlačenje globalnih zavisnosti između ulaznih i izlaznih podatka. Time se omogućuje značajno više iskorištavanja mogućnosti paralelizacije i postižu vrhunski rezultati u prevođenju prirodnog jezika.

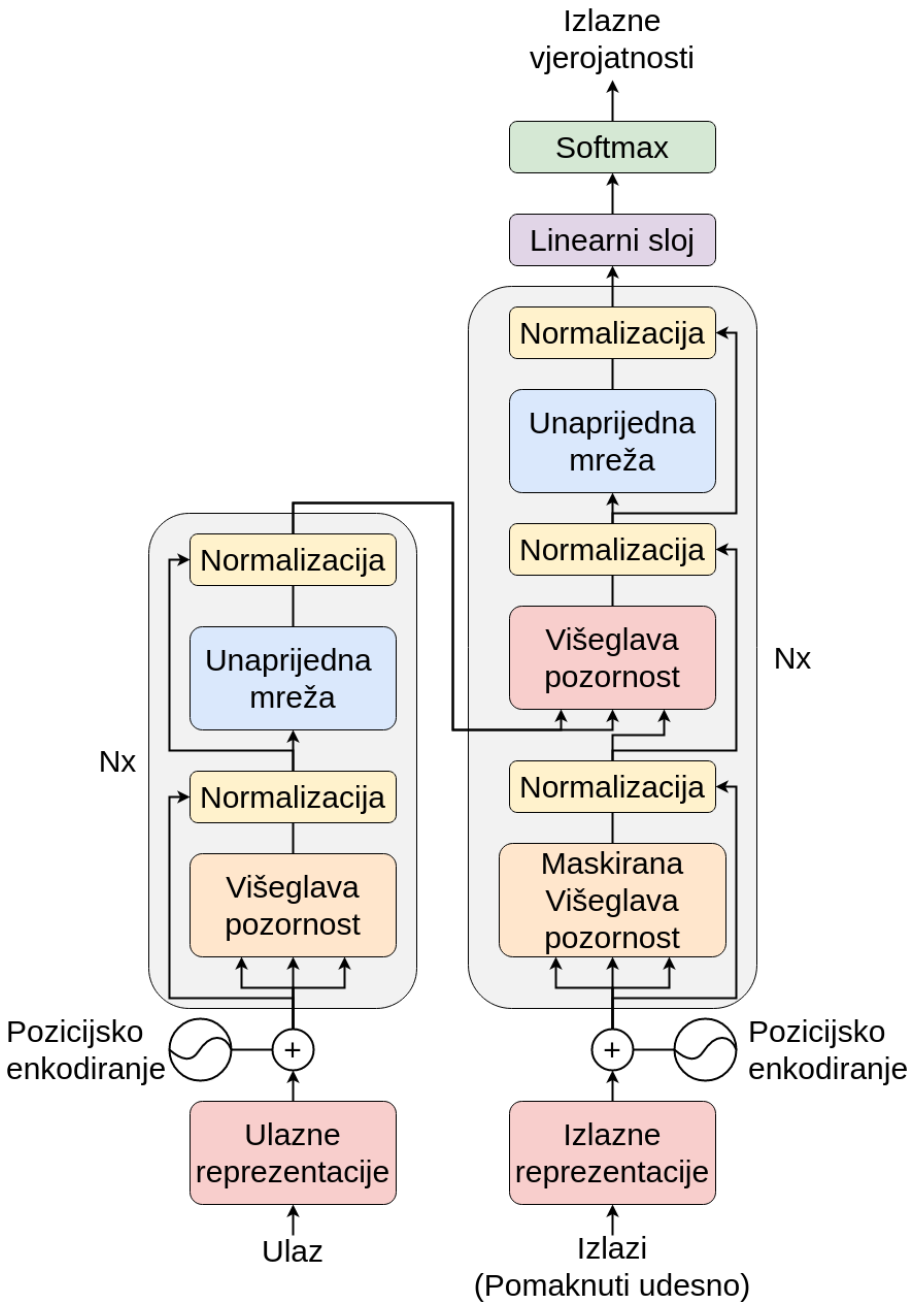
2.3.1. Arhitektura

Vaswani, Shazeer, Parmar i dr. [30] pišu da Transformer uglavnom prati ranije obrađenu enkoder-dekoder arhitekturu koristeći naslagani mehanizam samopozornosti (engl. *stacked self-attention*) i potpuno povezane slojeve na svakom koraku enkodera i dekodera kao što je prikazano na slici 7.

Enkoder se sastoji od stoga $N = 6$ identičnih slojeva, a svaki sloj od dva podsloja. Prvi od ta dva je višeglavi mehanizam samopozornosti (engl. *multi-head self-attention mechanism*), tj. modul koji paralelno izvršava nekoliko mehanizama pozornosti, a drugi jednostavna potpuno povezana unaprijedna neuronska mreža (engl. *position-wise fully connected feed-forward network*). Rezidualna konekcija [23] je kreirana oko svakog od dva podsloja, a zatim slijedi normalizacija sloja [44]. Normalizacijom sloja se ulazne vrijednosti za sve neurone istog sloja normaliziraju za svaki uzorak podataka te tako omogućuju brže treniranje i veću točnost generalizacije. Vaswani, Shazeer, Parmar i dr. [30] nastavljaju da iz toga slijedi da je izlaz svakog podsloja: $LayerNorm(x + Sublayer(x))$, gdje je $Sublayer(x)$ funkcija koju implementira podsloj. A kako bi se rezidualne konekcije olakšale, izlazi svih podslojeva modela, uključujući *embedding* slojeve, su dimenzija $d_{model} = 512$.

Dekoder se također sastoji od stoga $N = 6$ identičnih slojeva, ali uz ranije spomenuta dva podsloja on dodaje i treći koji izvršava višeglavu pozornost (engl. *multi-head attention*) nad izlazom stoga dekodera. Isto kao i kod enkodera, koriste se rezidualne konekcije oko svakog podsloja nakon čega slijedi normalizacija sloja. Podsloj samopozornosti je modificiran kako bi se u nekoj poziciji spriječilo obraćanje pozornosti na sljedeću poziciju. To se naziva maskiranje te uz činjenicu da su izlazi pomaknuti za jednu poziciju, osigurava da generiranje sekvence na poziciji i ovisi samo o poznatim generiranim sekvencama na pozicijama prije i. [30]

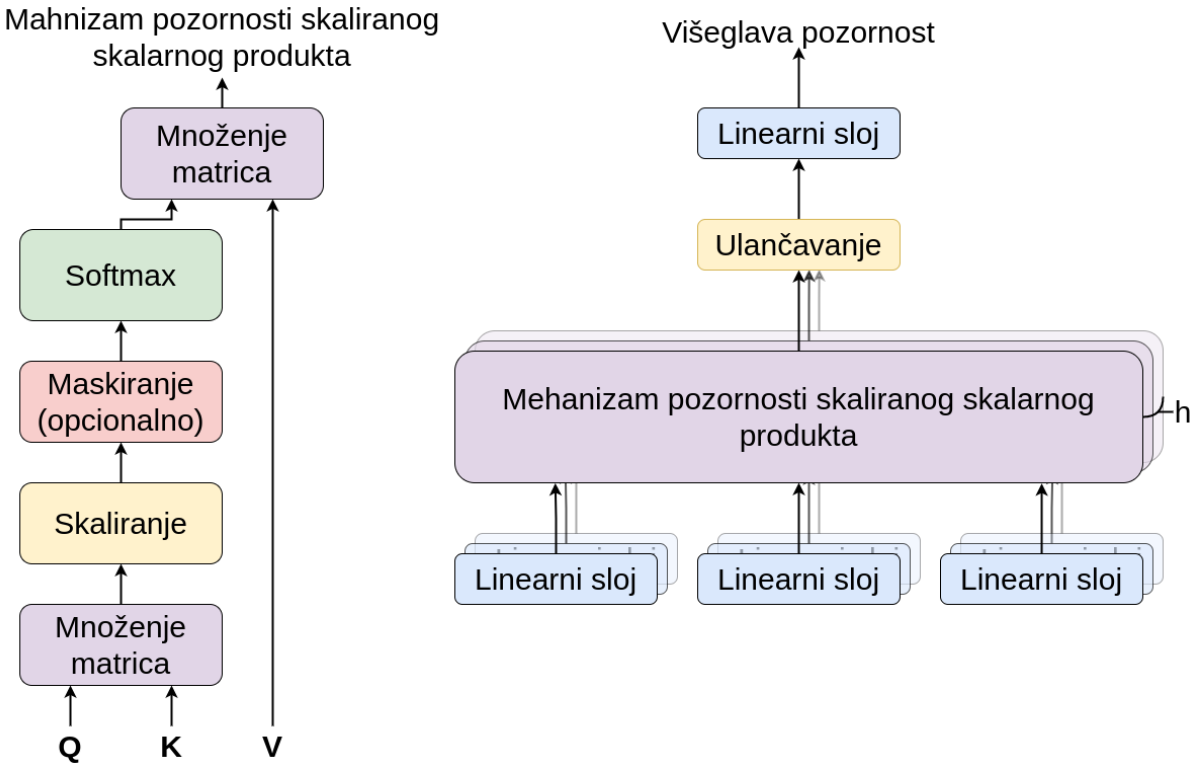
Na kraju još linearni sloj, koji je zapravo potpuno povezana neuronska mreža, projicira vektor dobiven od stoga dekodera u puno veći, takozvani *logits* vektor. U slučaju da model zna n jedinstvenih riječi, dimenzija bi takvog vektora bila jednaka n. Zadnji je sloj Softmax funkcije. To je normalizirana eksponencijalna funkcija koje pretvara vektor od n realnih brojeva u distribuciju vjerojatnosti od n mogućih ishoda. Tako se *logits* vektor pretvara u niz od n vjerojatnosti u rasponu 0 do 1 gdje svaka odgovara točno jednoj jedinstvenoj riječi iz vokabulara modela te gdje je zbroj svih vjerojatnosti jednak 1. Zatim se odabire polje s najvećom vjerojatnošću, odnosno pripadajuća riječ, što predstavlja izlaz za jedan vremenski korak nakon čega se postupak ponavlja.



Slika 7: Arhitektura modela Transformera (Vlastita izrada prema: Vaswani, Shazeer, Parmar i dr., 2017)

2.3.2. Izvedba mehanizma pozornosti

Vaswani, Shazeer, Parmar i dr. u [30] predstavljaju novi pristup mehanizmu pozornosti. Po njihovom shvaćanju funkcija pozornosti se može definirati kao preslikavanje upita (engl. *Query*) te parova ključ - vrijednost (engl. *Key - value pairs*) nekom izlazu, gdje su upit, ključ i vrijednost vektori. Imena tih vektora su analogna tradicionalnim sustavima za pronalaženje, a razlog takvog imenovanja će biti jasniji upoznavanjem njihovih namjena.



Slika 8: Prikaz mehanizma pozornosti skaliranog skalarnog produkta i višeglave pozornosti (Vlastita izrada prema: Vaswani, Shazeer, Parmar i dr., 2017)

Vaswani, Shazeer, Parmar i dr. [30] svoju izvedbu pozornosti nazivaju mehanizmom pozornosti skaliranog skalarnog produkta (engl. *Scaled dot-product attention*) te je ilustrativno prikazan na slici 8 lijevo. Osnova takvog mehanizma su ranije spomenuti skup upita $Q = \{q_1, ..., q_n\}$, skup ključeva $K = \{k_1, ..., k_m\}$ i skup vrijednosti $V = \{v_1, ..., v_m\}$, gdje je svaki vektor upita $q_i \in \mathbb{R}^{d_k}$, svaki vektor ključa $k_i \in \mathbb{R}^{d_k}$, te vrijednosti $v_i \in \mathbb{R}^{d_v}$. Drugim riječima, vektori ključa i upita su elementi vektorskog prostora dimenzije jednake dimenziji vektora ključa, dok je vektor vrijednosti element prostora svoje dimenzije. Takav se mehanizam pozornosti u [13] zatim definira kao:

$$\{\mathbf{h}_1, ..., \mathbf{h}_n\} = ATT(Q, K, V), \tag{2.3}$$

$$\mathbf{h}_i = \sum_{j=1}^{m} a_{ij} v_j$$

$$a_{ij} = \frac{exp(ATT - Mask(\frac{q_i \cdot k_j}{\sqrt{d_k}}))}{\sum_{l=1}^{m} exp(ATT - Mask(\frac{q_i \cdot k_j}{\sqrt{d_k}}))}.$$

Han, Zhang, Ding i dr. [13] dalje objašnjavaju da je intuitivno Q skup vektora upita za koji se računa pozornost, a K skup vektora ključeva u usporedbi s kojima se računa pozornost. Kao rezultat skalarnog produkta, dobija se ponder a_{ij} koji pokazuje kolika je zavisnost između vektora upita q_i i vektora ključa k_j. Na kraju je još potrebno izračunati ponderiranu srednju vrijednost vektora vrijednosti kao zadnji korak sloja pozornosti. Također, upozoravaju da treba uzeti u obzir da se $ATT - Mask$ koristi kao funkcija maskiranja kako bi se moglo ograničiti koji ključ - vrijednost par može utjecati na koji vektor upita. Ako se želi izbjeći obraćanje pozornosti vektora q_i na vektor k_j, $ATT - Mask$ se postavlja na $-\infty$, a inače je $ATT - Mask(x) = x$.

Time je završen izračun pozornosti za jedan element sekvence. Vektor dobiven kao rezultat se dalje može poslati unaprijednoj neuronskoj mreži. No u stvarnosti, kako bi se zapravo ostvarila ranije spomenuta mogućnost iskorištavanja svih računalnih resursa, od ovih se vektora kreiraju matrične reprezentacije $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, $\mathbf{K} \in \mathbb{R}^{n \times d_k}$ i $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ pa se pozornost može pojednostaviti na:

$$\mathbf{H} = ATT(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{AV}, \tag{2.4}$$

$$\mathbf{A} = Softmax(ATT - Mask(\frac{\mathbf{QK}^T}{\sqrt{d_k}})),$$

gdje se $Softmax$, odnosno normalizirana eksponencijalna funkcija koja svaki red od n realnih brojeva pretvara u distribuciju od n mogućih ishoda, primjenjuje nad svakim redom, $\mathbf{A} \in \mathbb{R}^{n \times m}$ je matrica pozornosti, a $\mathbf{H} \in \mathbb{R}^{n \times d_v}$ rezultat. [13]

Cijeli se dosad objašnjeni postupak odnosi na standardni mehanizam pozornosti skaliranog skalarnog produkta. No umjesto korištenja standardnog pristupa, u [30] je objašnjeno da Transformer primjenjuje višeglavi sloj pozornosti (engl. *multi-head attention layer*), prikazan na slici 8 desno. Takav je pristup u [13] definiran kao:

$$\mathbf{H} = MH - ATT(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \tag{2.5}$$

$$= Concat(\mathbf{H}_1, ..., \mathbf{H}_h)\mathbf{W}^0,$$

$$\mathbf{H}_i = ATT(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V),$$

gdje je h broj glave. $\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V$ se koriste za projiciranje ulaza $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ u prostor svojstva (engl. *Feature space*) i-te glave pozornosti. Nakon ulančavanja svih izlaza glavi koristeći $Concat$, višeglava pozornost ih projicira u završni prostor (engl. *output space*) primjenjujući $\mathbf{W}^0$.

2.3.3. Unaprijedna neuronska mreža i rezidualne konekcije

Osim podslojeva pozornosti, Vaswani, Shazeer, Parmar i dr. [30] pišu da se svaki enkoder i dekoder sloj sastoji i od potpuno povezane unaprijedne neuronske mreže kako je prikazano na slici 7. U [13] je objašnjeno da se ta unaprijedna mreža, ako primi ulaznu matricu $\mathbf{X} \in \mathbb{R}^{n \times d_i}$ koja predstavlja skup ulaznih vektora gdje je d_i vektor dimenzije, može definirati kao:

$$\mathbf{H} = FFN(\mathbf{X}) = \sigma(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.6}$$

gdje je σ funkcija aktivacije, najčešće ReLU funkcija, a $\mathbf{W}_1 \in \mathbb{R}^{d_i \times d_f}$, $\mathbf{b}_1 \in \mathbb{R}^{d_f}$, $\mathbf{W}_2 \in \mathbb{R}^{d_f \times d_0}$ i $\mathbf{b}_2 \in \mathbb{R}^{d_0}$ sve parametri koji se mogu naučiti (engl. *Learnable parameters*). $\mathbf{H} \in \mathbb{R}^{n \times d_0}$ je konačan rezultat unaprijednog sloja. Han, Zhang, Ding i dr. [13] još dodaju da je empirijski d_i postavljen da je jednak d_0, a d_f je postavljen kao mnogo veći od d_i i d_0.

Na kraju je još ostalo spomenuti ranije spomenute rezidualne konekcije i normalizaciju slojeva. Svrha njihove uporabe jest omogućavanje da arhitektura Transformera bude duboka. Han, Zhang, Ding i dr. [13] da objašnjavaju da ako postoji neuronski sloj f, rezidualna konekcija i normalizacijski sloj se definiraju kao:

$$\mathbf{H} = A\&N(\mathbf{X}) = LayerNorm(f(\mathbf{X}) + \mathbf{X}), \tag{2.7}$$

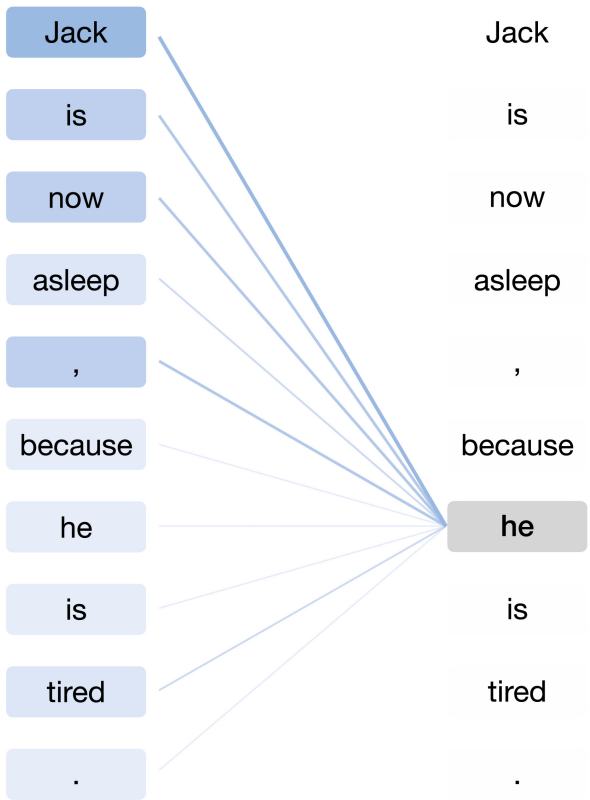
gdje $LayerNorm$ označava funkciju normalizacijskog sloja.

2.3.4. Primjena mehanizma pozornosti

Vaswani, Shazeer, Parmar i dr. [30] navode da postoje tri različita načina na koji se ranije objašnjen mehanizam višeglave pozornosti primjenjuje kod Transformera:

Samopozornost Ovi su slojevi sadržani u enkoderu. U slojevima samopozornosti svi ključevi (**K**), vrijednosti (**V**) i upiti (**Q**) dolaze iz izlaza prethodnog sloja enkodera. Tako svaka pozicija u enkoderu može obratiti pozornost na sve pozicije prethodnog sloja enkodera. Primjer ovoga načina primjene su izradili Han, Zhang, Ding i dr. [13] na slici 9, a prikazuje kako samopozornost precizno hvata zavisnost između riječi "Jack" kao subjekta i "he" kao zamjenice koja se odnosi na subjekt. Na ilustraciji tamnija boja okolnog kvadrata i poveznice označava jaču zavisnost između riječi.

Maskirana samopozornost Ovi slojevi u dekoderu omogućuju svakoj poziciji u dekoderu da obrati pažnju na sve prethodne i tu poziciju. Ono što je bitno jest spriječiti tok podataka ulijevo kako bi se zadržala autoregresivnost.Način implementacije je već ranije bio objašnjen kod mehanizma pozornosti skaliranog skalarnog produkta , a odnosi se na maskiranje, odnosno postavljanje na $-\infty$, svih vrijednosti u ulazu $Softmax$-a koje predstavljaju nedozvoljene konekcije.



Slika 9: Prikaz mehanizma samopozornosti (izvor: Han, Zhang, Ding i dr., 2021)

Unakrsna pozornost U enkoder-dekoder mehanizmu pozornosti upiti (**Q**) dolaze iz prethodnog dekoder sloja, a ključevi (**K**) i vrijednosti (**V**) iz izlaza enkodera. Time se omogućuje da svaka pozicija u dekoderu obraća pozornost na sve pozicije ulazne sekvence. To imitira tipični mehanizam pozornosti enkoder-dekoder arhitekture u Seq2Seq modelima kao što je to slučaj u [11].

2.4. Generativni Predtrenirani Transformer (engl. *Generative Pretrained Transformer*, GPT)

Uzimajući u obzir pristup samonadgledanog učenja spomenutog u poglavlju 2.1, Radford, Narasimhan, Salimans i dr. [36] su istraživali mogućnosti nenadgledanog predtreniranja i nadgledanog finog prilagođavanja modela. Tako su s ciljem kreiranja modela koji bi se mogao primijeniti za širok spektar zadataka uz vrlo malo ili uopće bez prilagodbe predstavili Generativni Predtrenirani Transformer, GPT.

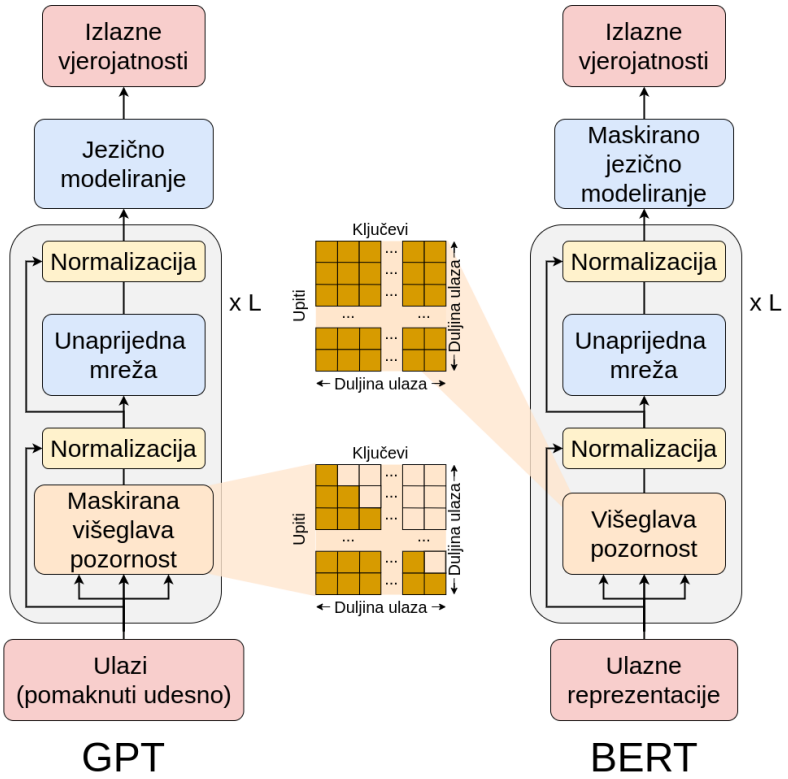
2.4.1. Arhitektura

Osnova ovog modela je Transformer-dekoder blok [45] koji odbacuje enkoder iz Transformer arhitekture prikazane u [30] čime se količina parametara modela smanjuje za skoro pola, prikazano na slici 10. Ulazna i izlazna sekvenca se spajaju u jednu "rečenicu" te se model trenira kao standardni jezični model. Odnosno, umjesto korištenja transdukcije rečenice

$(m^1, ...m^n) \mapsto (y^1, ...y^n)$, koristi se jedna rečenica $(w^1, ...w^{n+\eta+1}) = (m^1, ..., m^n, \delta, y^1, ...y^\eta)$, gdje je δ posebni separator. Takav se model trenira za predviđanje sljedeće riječi u rečenici, ako su mu poznate prethodne riječi:

$$p(w^1, ...w^{n+\eta}) = \prod_{j=1}^{n+\eta} p(w^i | w^1, ...w^{j-1}). \tag{2.8}$$

Drugim riječima, umjesto pretvaranja jedne rečenice koju čine riječi, odnosno tokeni, $m^1, ...m^n$ u drugu rečenicu sastavljenu od tokena $y^1, ...y^\eta$, od te se dvije rečenice kreira jedna koju čine tokeni $w^1, ..., w^{n+\eta+1}$, a odijeljene su δ kao posebnim separatorom. Tako se model, umjesto za funkciju pretvaranja iz jednog oblika u drugi, koristi za predviđanje sljedeće riječi u rečenici. Zbog toga se može reći da GPT primjenjuje generativno predtreniranje i diskriminativno fino prilagođavanje. Han, Zhang, Ding i dr. [13] navode da je GPT prvi model koji je upotrijebio modernu Transformer arhitekturu u svrhu samonadgledanog predtreniranja te da je empirijski pokazao znatne uspjehe na skoro svim NLP zadacima uključujući jezično zaključivanje, odgovaranje na pitanja, zdravorazumno zaključivanje, semantičku sličnost i klasifikaciju.



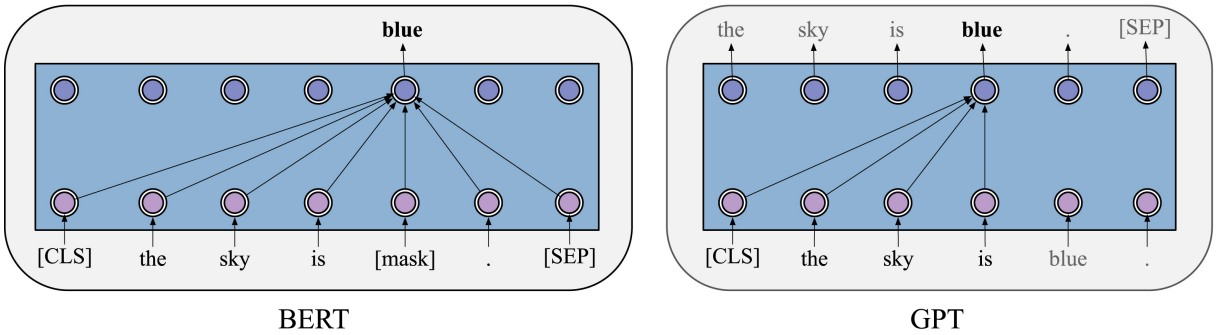
Slika 10: Usporedba arhitekture GPT-a i BERT-a (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021)

Dakle GPT će, uz dan veliki skup tekstualnih podataka bez oznaka, optimizirati standardno autoregresivno modeliranje jezika tako da maksimizira uvjetnu vjerojatnost svih riječi uzimajući prethodne riječi kao kontekst kao što je prikazano na slici 11. U predtreniranju uvjetnu vjerojatnost svake riječi modelira Transformer. Han, Zhang, Ding i dr. [13] dalje navode da GPT računa svoju distribuciju vjerojatnosti primjenjujući operaciju maskirane višeglave samopozor-

nosti (engl. *Masked multi-head self-attention*) nad prethodnim riječima. Odnosno, ako postoji skup simbola $X = \{x_0, x_1, ..., x_n, x_{n+1}\}$, GPT ima standardni cilj jezičnog modeliranja maksimizirajući vjerojatnost:

$$L(X) = \sum_{i=1}^{n+1} logP(x_i|x_{i-k}, ..., x_{i-1}; \Theta), \tag{2.9}$$

gdje je k raspon konteksta uzetog u obzir, vjerojatnost P se modelira Transformer-dekoderom s parametrima Θ, x_0 je posebni simbol [CLS], a x_{n+1} posebni simbol [SEP].



Slika 11: Razlika samopozornosti GPT i BERT arhitekture (izvor: Han, Zhang, Ding i dr., 2021)

2.4.2. GPT-2

Gradeći na uspjehu GPT-a, uskoro nakon je objavljen GPT-2 koji je donio veću raznolikost modela i nekoliko promjena u odnosu na svog prethodnika. GPT-2 je treniran u 4 arhitekture[1] koje se razlikuju u broju slojeva, parametara i dimenzionalnosti modela. Najmanji od ta 4 je jednak originalnom GPT-u sa 12 Transformer-dekoder slojeva i 124 milijuna parametara[2], ali uz nekoliko promjena. Prvo je normalizacija sloja premještena na ulaz svakog pod-bloka i dodana je normalizacija nakon zadnjeg bloka samopozornosti. Pri inicijalizaciji su ponderi rezidualnih slojeva skalirani s $1/\sqrt{N}$, gdje je N broj rezidualnih slojeva. Vokabular je povećan na 50.257, a širina konteksta s 512 na 1024.

Dimenzionalnost modela se odnosi na dimenzije vektora upita, ključeva i vrijednosti te također ima utjecaja na broj neurona u potpuno povezanom sloju. Tako na primjer najmanji GPT-2 model dimenzionalnosti 768 u prvom potpuno povezanom sloju ima 3072 neurona, odnosno 4*768 neurona. Brojka 4 ovdje dolazi iz originalne Transformer arhitekture te nije postojala potreba mijenjati taj dio arhitekture.

Mreža je uvježbana na 8 milijuna dokumenata, odnosno 40 GB tekstualnih podatka, preuzetih samo sa web stranica koje moderiraju ili filtriraju ljudi. Primjer koji je dan u [46] navodi da su preuzeli sadržaj sa svih vanjskih poveznica na društvenoj mreži "Reddit" koje su pozitivno

[1] Istraživači navode da se samo najveći model zove GPT-2, no taj se naziv nerijetko koristi kao grupni naziv za sve 4 verzije arhitekture popraćen oznakom veličine, npr. S (engl. *small*) za najmanju ili XL (engl. *Extra Large*) za najveću, pa će tako biti korišten i u ovom radu

[2] U [46] je navedena brojka od 117 milijuna parametara za model od 12 i 345 milijuna za model od 24 sloja no radi se o grešci koja je ispravljena na github repozitoriju projekta (`https://github.com/openai/gpt-2`).

ocijenjene sa strane barem 3 korisnika te mreže. Na kraju je taj skup podataka sadržavao 45 milijuna poveznica, podaci su preuzeti te su uklonjeni duplikati kao i sve poveznice na sadržaj na Wikipediji jer se on često koristi za druge skupove podatka pa se time izbjegava preklapanje skupova.

Tablica 1: Hiperparametri arhitekture za 4 različite veličine GPT-2 modela.

Parametara	Slojeva	d_{model}
124 Mil.	12	768
355 Mil.	24	1024
762 Mil.	36	1280
1542 Mil.	48	1600

(Izvor: Radford, Wu, Child i dr., 2019)

2.4.3. GPT-3

Daljni razvoj GPT modela prate male promjene u samoj arhitekturi modela iako se povećava broj slojeva, parametara i podataka za treniranje. U [47] navode da je GPT-3 treniran u 8 verzija[3], od 125 milijuna do 175 milijardi parametara, odnosno od 12 do 96 slojeva te dimenzijama u rasponu od 768 do 12.288, prikazano u tablici 2. U trenutku pisanja ovog rada, GPT-3 je dostupan samo po kontroliranoj proceduri slanja zahtjeva istraživačima u OpenAI te preko njihovog API-ja, a kao razlog navode sprječavanje mogućih zloporaba.

Zbog velike količine podataka za treniranje, njihove raznolikosti kao i velikog broja parametara modela, GPT-3 ostvaruje odlične rezultate na zadacima za koje uopće nije imao ili je imao jako malo primjera. Također je bio uspješan u odrađivanja zadataka u hodu za koje nikada nije bio treniran kao što su: pisanje SQL upita i koda, pisanje React i Javascript koda ako mu je bilo dano objašnjenje zadatka u prirodnom jeziku, ispravljanje rečenica s pomiješanim redoslijedom riječi, itd.

[3] Isto kao i kod GPT-2, istraživači u originalnom radu koriste naziv GPT-3 samo za najveću verziju modela, no uvriježeno je koristiti taj naziv kao grupni naziv za svih 8 verzija popraćen brojem slojeva ili drugom oznakom (npr. GPT3-12 ili GPT-3 S)

Tablica 2: Hiperparametri arhitekture za 8 različitih veličina GPT-3 modela.

Parametara	Slojeva	d_{model}
125 Mil.	12	768
350 Mil.	24	1024
760 Mil.	24	1536
1.3 Mlrd.	24	2048
2.7 Mlrd.	32	2560
6.7 Mlrd.	32	4096
13.0 Mlrd.	40	5140
175.0 Mlrd.	96	12288

(Izvor: Brown, Mann, Ryder i dr., 2020)

2.5. BERT (engl. *Bidirectional Encoder Representations from Transformers*)

Kao najveći problem dotadašnjih modela, uključujući i GPT model, Devlin, Chang, Lee i dr. [35] navode njihovu jednosmjernost te tvrde da to ograničava mogućnosti predtreniranih reprezentacija, pogotovo prilikom finog prilagođavanja. Na primjeru ranije opisanog GPT modela to se odnosi na ograničenje maskirane samopozornosti gdje svaka pozicija može uzimati u obzir samo one pozicije, odnosno riječi, koje se nalaze lijevo od nje. Drugim riječima, uzimaju se u obzir samo prethodne riječi u rečenici. Devlin, Chang, Lee i dr. [35] dalje navode da to predstavlja veliki problem prilikom finog prilagođavanja za zadatke poput odgovaranja na pitanja gdje je nužno inkorporirati kontekst s obje strane. Iz tog su razloga 2018. godine predstavili model dvosmjernih enkoder reprezentacija transformera (engl. *Bidirectional Encoder Representations from Transformers*, BERT).

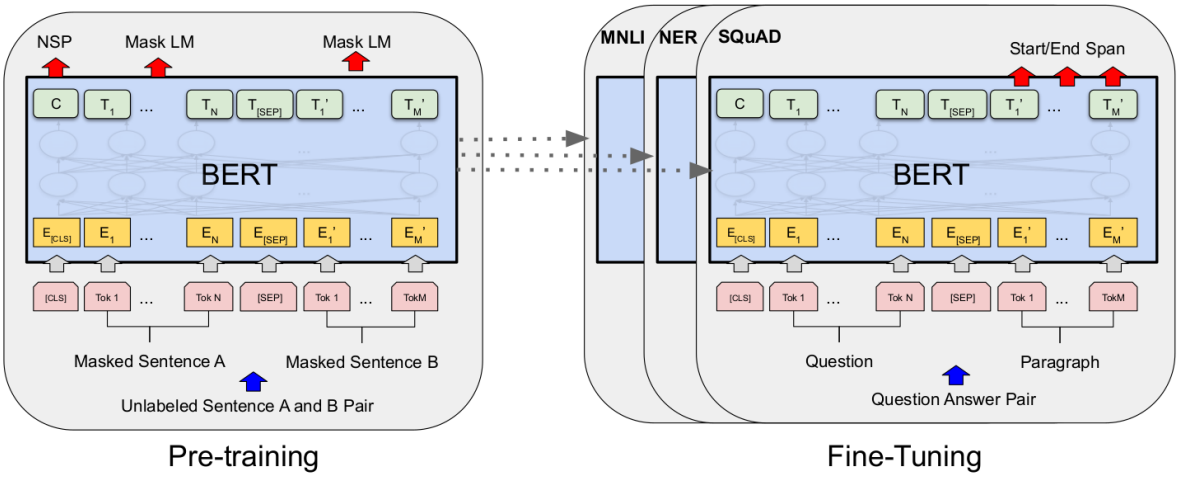
2.5.1. Predtreniranje

Isto kao i GPT, BERT ima dva stadija učenja, predtreniranje i fino prilagođavanje. Usporedba oba stadija je prikazana na slici 12. Prilikom predtreniranja BERT primjenjuje autoenkodirajuće jezično modeliranje (engl. *Autoencoding language modeling*) umjesto autoregresivnog koje primjenjuje GPT, prikazano na slici 11. Odnosno, kako Devlin, Chang, Lee i dr. [35] navode, za predtreniranje je dizajniran zadatak maskiranog jezičnog modeliranja (engl. *Masked language modeling*, MLM) po uzoru na Cloze [48] proceduru. MLM nasumično maskira neke tokene ulazne sekvence, a cilj modela je predvidjeti identifikacijski broj originalne riječi iz vokabulara. Za razliku od modeliranja s lijeva na desno, MLM omogućuje spajanje lijevog i desnog konteksta što zauzvrat omogućuje predtreniranje dubokog dvosmjernog Transformera što je prikazano na slici 12 lijevo.

Han, Zhang, Ding i dr. [13] MLM objašnjavaju na način da ako postoji skup tekstualnih podataka koji se sastoji od tokena $X = \{x_0, x_1, ..., x_n, x_{n+1}\}$, BERT nasumično maskira m tokena u X te zatim maksimizira sljedeću logaritamsku vjerojatnost:

$$L(X) = \sum_{i=1}^{m} logP([Mask]_i = y_i | \tilde{X}; \Theta), \tag{2.10}$$

gdje se vjerojatnost P modelira Transformer-enkoderom s parametrima Θ, $\tilde{X}$ je rezultat nakon maskiranja nekih tokena u X, $[Mask]_i$ je i-ta maskirana pozicija, a y_i originalni token i-te pozicije.



Slika 12: BERT predtreniranje i fino prilagođavanje (izvor: Devlin, Chang, Lee i dr., 2018)

Osim MLM-a, na slici 12 se može uočiti da se BERT predtrenira na još jednom zadatku. Taj zadatak se odnosi na predviđanje sljedeće rečenice (engl. *Next sentence prediction*, NSP), a izvršava se s ciljem modeliranja odnosa između čitavih rečenica kako bi mreža bila uspješnija na specifičnim zadacima s većim brojem rečenica kao što su jezično zaključivanje ili odgovaranje na pitanja. Taj zadatak, kako Han, Zhang, Ding i dr. [13] navode, koristi binarni klasifikator koji predviđa jesu li dvije rečenice skladne, odnosno koherentne. Dakle, MLM i NSP se zajedno koriste u fazi predtreniranja kako bi optimizirali parametre BERT-a.

Nakon predtreniranja, na slici 12 se može primjetiti da se BERT dalje može uspješno prilagoditi za zadatke kao što su:

MNLI (engl. *Multi-Genre Natural Language Inference Corpus*) [49] u kojemu model na osnovu rečenice premise i rečenice hipoteze predviđa povlači li premisa hipotezu, proturiječi li hipotezi ili je njihov odnos neutralan.

NER (engl. *Named entity recognition*) u kojemu cilj modela pronaći i klasificirati imenovane entitete spomenute u nestrukturiranom tekstu u predodređene kategorije.

SQuAD (engl. *Stanford Question Answering Dataset*) [50] u kojemu je cilj modela odgovoriti na pitanje o priloženom članku s Wikipedije, a odgovor je segment teskta iz samog članka.

Također, za razliku od GPT-a, model BERT-a se arhitekturalno ne razlikuje puno od originalne implementacije Transformera u [30] što je ranije prikazano na slici 10. Originalni je model napravljen u dvije verzije: $BERT_{BASE}$ s 12 slojeva, dimenzionalnosti 768 i 110 milijuna parametara što ga čini usporedivim s originalnim GPT modelom ili najmanjom verzijom GPT-2 te $BERT_{LARGE}$ s 24 sloja, dimenzionalnosti 1024 i 340 milijuna parametara.

2.5.2. Verzije BERT-a za hrvatski jezik

Za razliku od GPT-a, BERT ima izvedbu za hrvatski jeziku u obliku BERTić-a [51] i CroSloEngular BERT-a [52]. Ovi se modeli razlikuju po tome što je CroSloEngular višejezični model uvježban na hrvatskom, slovenskom i engleskom jeziku, dok je BERTić uvježban za hrvatski, bosanski, srpski i crnogorski jezik. Oba modela koriste $BERT_{BASE}$ arhitekturu s 12 slojeva.

CroSloEngular je uvježban na skupu 5.9 milijardi riječi u 40 epoha ili 3.96 milijuna koraka s maksimalnom duljinom sekvence od 128 te još 4 epohe s maksimalnom duljinom sekvence od 512 na jednom Google Cloud TPU v2 što je trajalo otprilike 3 tjedna. BERTić je uvježban na skupu od skoro 8.4 milijardi riječi u 50 epoha ili 2 milijuna koraka na 8 TPU v3 uređaja no ne navode vrijeme koje je bilo potrebno za cjelokupno treniranje.

Ljubešić i Lauc su u [51] usporedili ta dva modela sa višejezičnim mBERT [35] modelom te CLASSLA [53] alatom kada je to bilo moguće. Na zadatku morfosintaktičke anotacije, prikazano u tablici 3, BERTić ostvaruje najbolje rezultate za 3 od 4 skupa podataka. Taj se zadatak odnosi na anotiranje gramatičke klase svake riječi u rečenici. To može uključivati ulogu riječi u rečenici, primjerice subjekt ili predikat, gramatički rod riječ, padež, ako je primjenjivo, gramatički broj, odnosno je li riječ u jednini ili množini, itd.

Tablica 3: Prosječni microF1 rezultati za morfosintaktičku anotaciju

skup podataka	jezik	vrsta	CLASSLA	mBERT	cseBERT	BERTić
hr500k	Hrvatski	književni	93.87	94.60	95.74	**95.81**
reldi-hr	Hrvatski	kolokvijalni	-	88.87	91.63	**92.28**
SETimes.SR	Srpski	književni	95.00	95.50	**96.41**	96.31
reldi-sr	Srpski	kolokvijalni	-	91.26	93.54	**93.90**

(Izvor: Ljubešić i Lauc, 2021)

Na tablici 4 Ljubešić i Lauc [51] su usporedili rezultate za zadatak prepoznavanja imenovanih entiteta (engl. *Named entity recognition*). Cilj zadatka je u rečenici pronaći sve riječi koje predstavljaju primjerice ime osobe, godinu, naziv nekog proizvoda, adresu i slično te ih odgovarajuće klasificirati. Tako bi u rečenici: "Ivan je kupio Jamnicu.", mreža označila riječ "Ivan" kao ime osobe, a "Jamnicu" kao naziv robne marke. Rezultati na tablici 4 pokazuju kako BERTić opet za 3 od 4 skupa podatka postiže najbolje rezultate, a za 4. je to u ovom slučaju mBERT [35].

Tablica 4: Prosječni microF1 rezultati za zadatak prepoznavanja imenovanih entiteta

skup podataka	jezik	vrsta	CLASSLA	mBERT	cseBERT	BERTić
hr500k	Hrvatski	književni	80.13	85.67	88.98	**89.21**
reldi-hr	Hrvatski	kolokvijalni	-	76.06	81.38	**83.05**
SETimes.SR	Srpski	književni	84.64	**92.41**	92.28	92.02
reldi-sr	Srpski	kolokvijalni	-	81.29	82.76	**87.92**

(Izvor: Ljubešić i Lauc, 2021)

2.6. Suvremeni modeli

Han, Zhang, Ding i dr. [13] navode da su se nakon pojave GPT i BERT modela pojavila i neka poboljšanja od kojih vrijedi spomenuti RoBERTa-u (engl. *Robustly Optimized BERT Pre-training Approach*) [32] i ALBERT (engl. *A Lite BERT for Self-supervised Learning of Language Representations*) [54]. RoBERTa je svoj uspjeh postigla s 4 jednostavne promjene:

- Izbačen je NSP zadatak prilikom predtreniranja.
- Povećan je broj koraka kod predtreniranja, kao i veličina grupa (engl. *batch* i sama količina podataka).
- Predtreniranje se vrši na duljim rečenicama.
- Dinamički se mijenja uzorak postavljanja [MASK] simbola.

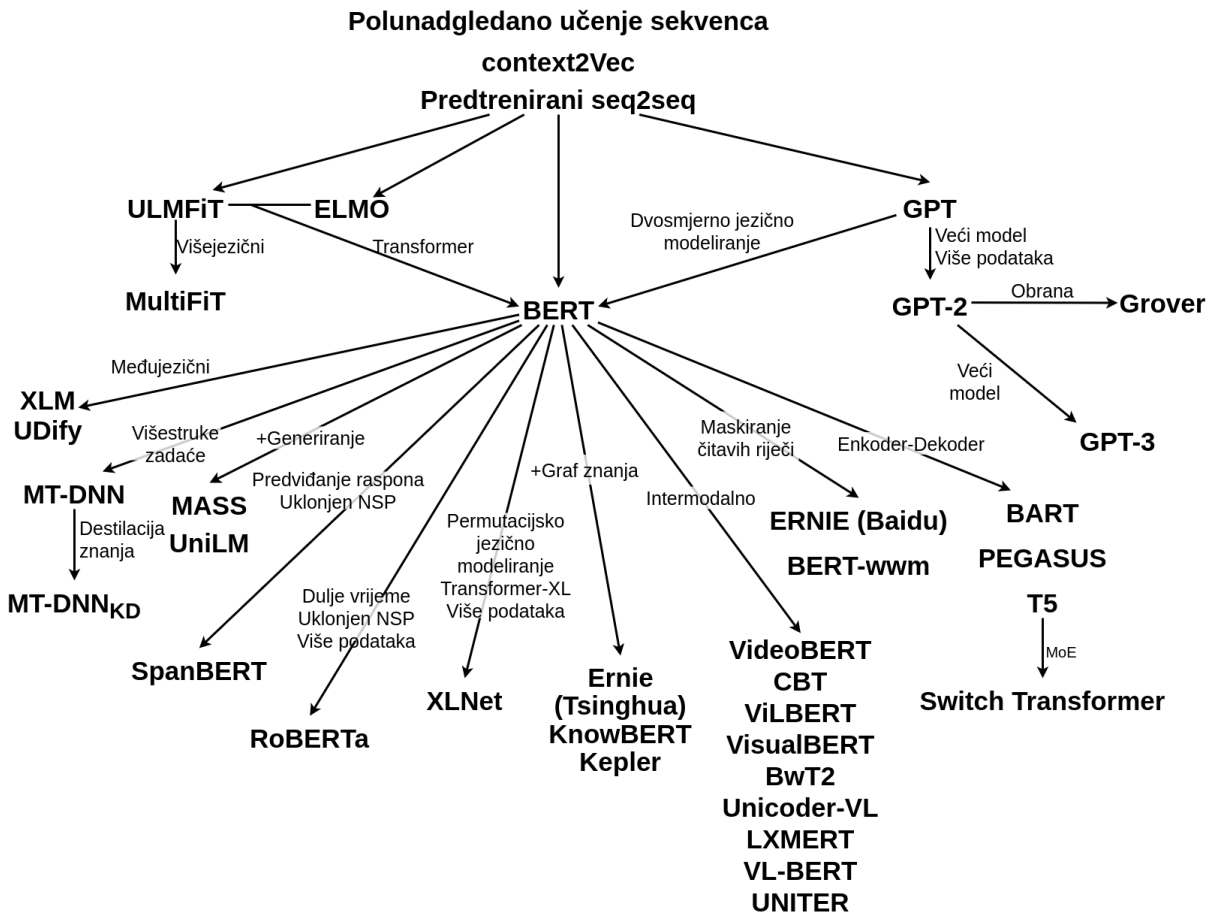
Ostvarivanjem empirijski boljih rezultata u odnosu na BERT, RoBERTa je pokazala da je NSP zadatak relativno beskoristan za predtreniranje BERT-a. ALBERT je, s druge strane, za cilj uzeo smanjenje broja parametara kako bi zahtjevao manje memorije no zauzvrat mu je potrebno više vremena kod finog prilagođavanja i jezičnog razumijevanja. Definiran je s 3 promjene u odnosu na BERT:

- Faktorizira ulaznu matricu riječnih reprezentacija u dvije manje matrice.
- Provodi dijeljenje parametara između svih Transformer slojeva čime značajno smanjuje količinu parametara.
- Predlaže zadatak predviđanja reda rečenica (engl. *Sentence order prediction*, SOP)[4] umjesto NSP zadatka.

Osim RoBERTa-e i ALBERT-a postoji velik broj varijacija BERT modela. Han, Zhang, Ding i dr. [13] još dodaju da ti modeli često bolje uče iz neoznačenih podataka, a neki predlažu novu arhitekturu i isprobavaju drugačije zadatke prilikom predtreniranja. Neki od tih modela

[4] SOP je zadatak koji za cilj ima pronalaženje pravilnog redoslijeda rečenica u tekstu s nasumičnim redoslijedom rečenica

uključuju: XLNet [31], UniLM [55], MASS [56], SpanBERT [57] i ELECTRA [58]. Kako je i veličina modela još uvijek ključan faktor, istraživači su predstavili modele koji imaju stotine milijardi parametara kao ranije spomenute verzije GPT modela, ali i druge poput Switch Transformera [59] ili modela koji pokušavaju povećati računalnu efikasnost toliko velikih modela poput Megatron-lm [60], ZeRO [61] ili ZeRO-Offload [62]. Han, Zhang, Ding i dr. [13] su na kraju još i slikovno prikazali podjele bitnijih modela i njihovih prethodnika, odnosno arhitektura na kojima su temeljeni, na slici 13.



Slika 13: Podjela modela i njihovih prethodnika (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021)

3. Izrada vlastitog predtreniranog modela

U ovom će poglavlju biti prikazan i objašnjen proces izrade vlastitog predtreniranog modela. Prvo će biti spomenuto korišteno okružje, alati i biblioteke, zatim će biti riječ o podacima za treniranje, a na kraju će biti prikazan način treniranja. Za tu je svrhu odabrana GPT-2 arhitektura jer, kako je bilo ranije spomenuto u poglavlju 2.5.2, za razliku od GPT-2 modela verzije BERT-a za hrvatski jezik već postoje. Od verzija GPT-2 modela prikazanih na tablici 1 koristit će se najmanji zbog ograničenih dostupnih računalnih resursa te zbog jednakog broja slojeva osnovnoj verziji BERT-a na kojoj su temeljeni BERTić i CroSloEngular. Time se ostvaruje mogućnost usporedbe preformansi vlastitog s tim modelima.

3.1. Korišteni alati i biblioteke

Google Colaboratory , ili skraćeno Colab, je online okruženje izrađeno na temelju Jupyter bilježnica. Radi se o Google-ovom proizvodu koji se može koristiti besplatno ili uz mjesečnu pretplatu. Radom u ovom okružju postoji mogućnosti korištenja GPU ili TPU uređaja. Model u ovome radu je izrađen koristeći Pro verziju okruženja s kojom je na raspolaganju bilo 35 GB RAM memorije i 24-satno trajanje sesije, no zbog greške Colab-ovog razvojnog tima, koja u trenutku pisanja ovog rada još nije ispravljena[1], sesija se prekidala nakon 12 sati korištenja TPU-a što je bitno utjecalo na način treniranja modela.

Google Drive se koristi kao prostor za trajnu pohranu podataka. Ranije je navedeno da sesije u Google Colab okruženju traju maksimalno 24 sata nakon čega se svi podaci gube. Zbog toga, zbog jednostavnog povezivanja s okružjem te zbog velike propusnosti mreže između okružja i pohrane na oblaku je Google Drive predstavljao optimalno rješenje.

Tensorflow je sveobuhvatna platforma otvorenog koda za strojno učenje. Sastoji se od većeg broja alata i biblioteka koje omogućuju pisanje koda u više razina apstrakcije. Primjerice Keras API omogućuje više razinu apstrakcije primjereniju početnicima, a također postoje i opcije koje omogućuju veću razinu kontrole nad samim kodom. Originalno su ga kreirali istraživači i inženjeri Google Brain tima, a danas zajednica značajno pridodaje održavanju platforme. API postoji za Python s kojim će se Tensorflow koristiti u ovom radu, no također postoji stabilni API za C++ i ne toliko stabilni API za druge programske jezike.

Huggingface je zajednica i platforma za znanost o podacima (engl. *Data science*) koja pruža alate koji korisnicima omogućuju izradu, treniranje i upogonjavanje modela strojnog učenja temeljenih na otvorenom kodu i tehnologijama. U ovom je radu Huggingface korišten kao repozitorij za izrađeni model, modul *transformers* za preuzimanje i prilagođavanje originalnog GPT-2 modela te modul *datasets* za funkcije koje olakšavaju obradu podataka uključujući paralelizaciju tokenizacije.

Pandas je Python biblioteka za analizu i obradu podataka te se često može vidjeti u raznim

[1] Problem je prijavljen na Github repozitoriju Google Colab-a no od 2020. godine rješenje nije pronađeno (`https://github.com/googlecolab/colabtools/issues/1053`)

projektima gdje je potreban jednostavan način za upravljanjem podacima u kodu. U ovom radu se koristi za obradu podataka prilikom početnog učitavanja i pretprocesuiranja.

Pickle je modul koji omogućuje jednostavnu serijalizaciju i deserijalizaciju strukture Python objekta. Odnosno, pretvaranja Python objekta u tok bajtova za spremanje u datotekama ili bazama podataka. U ovom radu se koristi upravo za spremanje podataka nakon što je nad njima izvršena neka operacija obrade kako bi se kasnije jednostavno i u istom obliku mogli ponovo učitati u kod kada je to potrebno.

NLTK (engl. *Natural Language Toolkit library*) je skup biblioteka i programa za simboličko i statističko obrađivanje podataka pomoću Pythona. U ovom se radu koristi zbog BLEU metode koja ocjenjuje kvalitetu modela prilikom prevođenja teksta.

3.2. Podaci za treniranje

Podaci korišteni za treniranje predtreniranog modela rezultat su rada Ljubešić i Klubička [63]. Ovaj je skup napravljen pretraživanjem web stranica .hr domene 2011. godine pa zatim opet 2014. godine. Skup se sastoji od 1.4 milijarde tokena, odnosno oko 1.2 milijarde riječi, a zapisi unutar dokumenata su u sljedećem XML formatu:

```
<p urldomain="splitculture.hr" url ="http://splitculture.hr/onama" lang="hr"
    langdistr="hr:-0.486|sr:-0.514">
    <s>
        Dugo Dugo dugo Rgp
        vremena vremena vrijeme Ncnsg
        smo smo biti Rgp
        razmišljali razmišljali razmišljati Vmp-pm
        kako kako kako Cs
        još još još Rgp
        bolje bolje dobro Rgc
        perdstaviti perdstaviti perdstaviti Vmn
        naš naš naš Ps1msan
        grad grad grad Ncmsan
        kao kao kao Cs
        jedno jedno jedan Mlcnsn
        od od od Sg
        najatraktivnijih najatraktivnijih atraktivan Agsmpgy
        turističkih turističkih turistički Agsmpgy
        <g/>
        , , , Z
        kulturnih kulturnih kulturan Agpfpgy
        i i i Cc
        sportskih sportskih sportski Agpnpgy
        odredišta odredišta odredište Ncnpg
        na na na Sl
        Mediteranu Mediteranu Mediteran Npmsl
        <g/>
        . . . Z
    </s>
</p>
```

Može se uočiti da su u s oznakom <p> označene web stranice, a metapodaci o zapisu se nalaze u zaglavlju same oznake. Ti metapodaci uključuju adresu mrežnog mjesta, adresu do specifične stranice na tom web mjestu, oznaku jezika sadržaja te normalizirane distribucije logaritamske vjerojatnosti za jezike uzete u obzir. Sadržaj web stranice je naznačen oznakama <s> te svaki red predstavlja jednu riječ. Riječ u prvom stupcu predstavlja originalnu riječ sa web stranice, riječ u drugom stupcu je normalizirana putem dijakritičke obnove (engl. *Normalised via diacritic restoration*), u trećem se stupcu nalazi lema riječi, a zadnji stupac predstavlja morfosintaktičke oznake. Tekst iz ranijeg XML zapisa preglednije je prikazan na tablici 5.

Tablica 5: Zapis jedne web stranice u skupu podataka

Original	Normalizacija	Lema	Morfosintaktička oznaka
Dugo	Dugo	dugo	Rgp
vremena	vremena	vrijeme	Ncnsg
smo	smo	biti	Rgp
razmišljali	razmišljali	razmišljati	Vmp-pm
kako	kako	kako	Cs
još	još	još	Rgp
bolje	bolje	dobro	Rgc
perdstaviti[2]	perdstaviti	perdstaviti	Vmn
naš	naš	naš	Ps1msan
grad	grad	grad	Ncmsan
kao	kao	kao	Cs
jedno	jedno	jedan	Mlcnsn
od	od	od	Sg
najatraktivnijih	najatraktivnijih	atraktivan	Agsmpgy
turističkih	turističkih	turistički	Agsmpgy
,	,	,	Z
kulturnih	kulturnih	kulturan	Agpfpgy
i	i	i	Cc
sportskih	sportskih	sportski	Agpnpgy
odredišta	odredišta	odredište	Ncnpg
na	na	na	Sl
Mediteranu	Mediteranu	Mediteran	Npmsl
.	.	.	Z

(Vlastita izrada prema zapisu iz: Ljubešić i Klubička, 2016)

U XML zapisu su se još mogle primijetiti oznake <g/> koje su u tablici 5 izbačene, a predstavljaju praznine koje se pojavljuju na web stranicama. Morfosintaktičko označavanje su Ljubešić i Klubička odradili HunPos-om [64] [3]. Tako oznaka "Ncnsg" za riječ "vremena" prenosi

[2] Riječ je krivo napisana na web stranici, kao takva spremljena u skupu podataka te se takva koristi za treniranje mreže. Ovaj zapis pokazuje probleme s kojima se susreće mreža prilikom treniranja, odnosno jedan problem kod korištenja velikih skupova nepregledanih podataka.

[3] Poveznica na projekt `https://code.google.com/p/hunpos/`

informacije da je riječ imenica (N, engl. *Noun*), da je uobičajena (c, engl. *common*), srednjeg roda (n, engl. *neuter*), u jednini (s, engl. *singular*) te da je u genitivu (g). [4]

3.2.1. Pretprocesuiranje

Kako je bilo objašnjeno u poglavlju 2.4, cilj predtreniranja GPT-2 modela jest optimizacija stadardnog autoregresivnog modeliranja jezika. Tako su tom modelu potrebni tekstualni podaci u obliku rečenica ili odlomaka teksta. Iz tog razloga pretprocesuiranje podataka počinje uklanjanjem ranije spomenutih oznaka praznina </g>. Njihovim se uklanjanjem izbjegavaju greške koje su se događale prilikom učitavanja datoteka kao XML datoteka. Dalje slijedi zanemarivanje svega osim originalnih riječi koje se pojavljuju u tekstu jer nije potrebno za jezično modeliranje. Sve se rečenice grupiraju po odlomcima kako bi se mogle bolje modelirati međurečenične zavisnosti. Time se dobiva 28,771,178 odlomaka s 67,403,219 rečenica koje zauzimaju oko 7.5 GB memorijskog prostora. U tim se podacima još uvijek pojavljuju oznake kojima nije mjesto u govornom jeziku. To je posljedica preuzimanja podataka s interneta programskim putem. Iz tog su se razloga svi podaci provjereni za zaostale html oznake, odnosno zadržani su samo ASCII i latinični znakovi, brojevi i razmaci. Oko svih su interpunkcijskih znakova dodani razmaci, a svi su razmaci, ako ih je više uzastopnih, svedeni na samo jedan. Ova obrada, iako nije komplicirana, se izvodi na velikom skupu podataka. Zbog tog razloga, odnosno zbog ograničenja radne memorije koje je spomenuto u opisu programskog okružja u poglavlju 3.1, izvorni se skup podataka morao podijeliti na 7 jednakih dijelova kako bi svaki od njih mogao stati u radnu memoriju i biti obrađen. Na kraju su izbačeni svi zapisi, odnosno odlomci, čija je duljina kraća od 10 znakova pa se tako uklonilo oko 850,000 zapisa.

3.3. Kreiranje i treniranje modela

Cijela obrada podataka kao i treniranje modela su izvršeni Google Colab online okruženju. Glavna prednost tog okruženja je, kako je u poglavlju 3.1 spomenuto, jeftin pristup računalnim resursima koji dolaze uz ograničenje trajanja sjednice. Tako je za treniranje ovog modela bio dostupan TPU i oko 35 GB radne memorije, ali uz maksimalno trajanje sjednice od 12 sati. Takva su ograničenja uzrokovala specifičan način treniranja koji će kasnije biti detaljnije objašnjen. No za izradu upotrebljivog modela nije dovoljno samo njegovo treniranje već i izrada rječnika kojim će se služiti.

3.3.1. Enkodiranje parova bajtova (engl. *Byte pair encoding*, BPE)

Radford, Wu, Child i dr. [46] navode da bi opći jezični model trebao moći računati vjerojatnost za, odnosno generirati, bilo koji niz znakova te da procesuiranje Unicode nizova znakova kao sekvence UTF-8 bajtova dobro odrađuje taj zadatak. No jezični modeli na razini bajtova nisu kompetitivni s onima na razini riječi u slučaju velikih skupova podataka. Isti su problem imali Radford, Wu, Child i dr. prilikom treniranja standardnog jezičnog modela na razini bajtova.

[4] Popis svih oznaka dostupan na: `http://nl.ijs.si/ME/V6/msd/html/msd-hbs.html#msd.msds-hbs`

BPE [65], kako Radford, Wu, Child i dr. [46] navode, se pokazao kao praktični kompromis ove dvije metode te je uspješno izmjenjivao korištenje ulaza na razini riječi za česte te ulaza na razini znakova za rijetke nizove simbola. Jedna od prednosti BPE-a jest da bi uspješna implementacija na razini bajtova rezultirala osnovnim vokabularom od samo 256 znakova. No primjećeno je pojavljivanje varijacija čestih riječi pa se tako riječ *"dog"* još pojavljivala kao: *"dog."*, *"dog!"*, i *"dog?"*. Kako bi izbjegli takva ponavljanja, Radford, Wu, Child i dr. [46] navode da su spriječili spajanje znakova iz različitih Unicode kategorija znakova, no napravljena je iznimka za razmake jer se tom iznimkom postiglo znatno poboljšanje efikasnosti kompresije uz minimalnu fragmentaciju riječi.

Ovakve su ulazne reprezentacije omogućile empirijski dokazane benefite korištenja reprezentacija na razini riječi sa općenitosti reprezentacija na razini bajtova. Na kraju, Radford, Wu, Child i dr. [46] još navode da se ovakvim pristupom omogućilo određivanje za bilo koji Unicode niz znakova pa se model može evaluirati na bilo kojem skupu podataka neovisno o predprocesuiranju, tokenizaciji ili veličini vokabulara.

Za ovaj je rad, radi osiguravanja maksimalne efikasnosti i efektivnosti na hrvatskom jeziku, po uzoru na ranije opisani način reprezentacije znakova i riječi, odnosno tokena, kreiran tokenizator sa vokabularom za hrvatski jezik. Sam je postupk, zbog korištenja Huggingface-ovog modula *tokenizers*, veoma jednostavan i direktan te je prikazan na sljedećem odlomku programskog koda:

```
1  from tokenizers import Tokenizer
2  from tokenizers.models import BPE
3  from tokenizers.normalizers import NFKC, Sequence
4  from tokenizers.pre_tokenizers import ByteLevel
5  from tokenizers.decoders import ByteLevel as ByteLevelDecoder
6  from tokenizers.trainers import BpeTrainer
7
8  tokenizer = Tokenizer(BPE())
9  tokenizer.normalizer = Sequence([NFKC()])
10 tokenizer.pre_tokenizer = ByteLevel()
11 tokenizer.decoder = ByteLevelDecoder()
12
13 trainer = BpeTrainer(vocab_size=50257, show_progress=True, inital_alphabet=ByteLevel
       .alphabet(), special_tokens=[
14             "<s>",
15             "<pad>",
16             "</s>",
17             "<unk>",
18             "<mask>"
19         ])
20 tokenizer.train(trainer = trainer, files = ["putanja/do/datoteka/s/tekstnim/podacima
       "])
21 tokenizer.model.save("putanja/do/direktorija/za/spremanje/", prefix = None)
```

U kodu se može uočiti da se tokenizator u linijama 8 do 11 kreira, a u linijama 13 do 20 trenira. Kako je ranije navedeno, Radford, Wu, Child i dr. [46] navode da BPE treba osnovni vokabular veličine 256 unosa. Uz osnovne unose, odredili su da nije potreban vokabular veći od 50,000 tokena. Tako je određena ukupna veličina vokabulara od 50,257 jer je dodan po-

sebni token za kraj teksta. Iako je veličina vokabulara hiperparametar koji se može proizvoljno odabrati, veći vokabular dolazi s negativnom stranom jer zahtjeva za veću količinom memorije i vremenski je kompleksniji. Kako je u originalnoj implementaciji definiran broj od 50,257, nema očitog razloga da ova implementacija za hrvatski jezik ne koristi tu istu veličinu.

3.3.2. Tokenizacija

Uz gotov tokenizator, podaci se mogu lako pretvoriti u niz tokena kao ulaza za jezično modeliranje mreže. Kako je objašnjeno u poglavlju 2.4.1, GPT-2 treba ulazne podatke u tri dijela:

- Niz identifikacijskih brojeva tokena originalnog teksta koji označavaju unose u vokabularu tokenizatora,
- Oznake koje mreža, predviđajući sljedeći token u nizu, treba pogoditi u tom vremenskom koraku,
- Maska pozornosti koja jedinicom označava sve valjane tokene u nizu, a nulom sve posebne tokene koji popunjavaju razliku između originalne i maksimalne duljine teksta.

U programskom je kodu definirana funkcija koja obavlja zadaću kreiranja ta tri dijela ulaznih podataka te je prikazana u sljedećem odlomku programskog koda:

```
1  EOS_TOKEN = "</s>"
2  MAX_TOKENS = 512
3
4  def tokenize_function(text, tokenizer=tokenizer):
5      text = [ex + EOS_TOKEN for ex in text["text"]]
6
7      output = tokenizer(
8          text,
9          add_special_tokens = True,
10         max_length = MAX_TOKENS,
11         truncation = True,
12         padding = 'max_length',
13     )
14     output["labels"] = [x[1:] for x in output["input_ids"]]
15     output["labels"] = [
16         [-100 if x == tokenizer.pad_token_id else x for x in y]
17         for y in output["labels"]
18     ]
19
20     output["input_ids"] = [x[:-1] for x in output["input_ids"]]
21     output["attention_mask"] = [x[:-1] for x in output["attention_mask"]]
22
23     return output
```

Odmah se može uočiti da je definirana maksimalna duljina ulaznog niza podataka od 512 tokena. Iako je to kraće od duljine najduljeg odlomka u skupu podataka, dovoljno je za

obradu najvećeg dijela. Kraća maksimalna duljina ulaznog niza također osigurava manje zauzimanje radne memorije kao i brže treniranje. U funkciji je zatim na kraj teksta dodan posebni token za kraj ulaznog teksta. Tokenizator, u 7. liniji programskog odlomka, takav tekst zatim tokenizira, a kao izlaz daje niz identifikacija tokena (u kodu *output["input_ids"]*) i masku pozornosti (u kodu *output["attention_mask"]*).

Nakon samog tokeniziranja, u izlaznom skupu moramo definirati oznake za sljedeću riječ, a to se postiže u linijama 14 do 18. Ovo je jedna bitna, ali loše dokumentirana razlika između Tensorflow i PyTorch implementacije GPT-2 modela. PyTorch samostalno generira oznake za sljedeće tokene u nizu tokom treniranja, no za Tensorflow je to potrebno eksplicitno odraditi. Tako u 14. liniji programskog odlomka definiramo da su oznake (u kodu *output["labels"]*) svi tokeni osim prvog. Time se postiže pomicanje udesno objašnjeno u poglavlju 2.4.1. Sve oznake na čijem se mjestu nalazi posebni token za popunjavanje (u kodu *pad_token*) se postavljaju na vrijednost -100 jer tako ti tokeni neće ulaziti u računanje funkcije gubitka prilikom treniranja. Na kraju se još u nizu identifikacija originalnih tokena i maske pozornosti uklanjaju poslijednji elementi kako bi duljina odgovarala duljini niza oznaka, odnosno 511.

Samo pozivanje funkcije tokenizacije vrši se pomoću metode *map* koja je dio Huggingface-ovog *"Dataset"* objekta. U sljedećem programskom odlomku:

```
1  from datasets import Dataset
2
3  ds = Dataset.from_pandas(df)
4
5  ds = ds.map(
6    tokenize_function,
7    batched = True,
8    batch_size = 512,
9    remove_columns=["text"],
10   load_from_cache_file = True,
11   num_proc = strategy.num_replicas_in_sync,
12 )
```

U 3. se liniji kreira *"Dataset"* objekt sa potrebnim podacima od *"DataFrame"* objekta biblioteke Pandas koji je služio za dosadašnju obradu i pretprocesuiranje podataka. Razlog korištenja *"Dataset"* objekta može se uočiti u liniji 11 koja definira broj paralelnih procesa koji će vršiti funkciju tokenizacije. Huggingface-ova biblioteka dolazi s osiguranom kompatibilnošću sa Google TPU-om pa se tako funkcija tokenizacije odvija paralelno u 8 procesa. Ostali parametri *map* metode definiraju da se podaci obrađuju u skupovima po 512 zapisa jer je to najveća veličina istovremene obrade koju je moguće izvršiti zbog ograničenja radne memorije.

U poglavlju 3.2.1 je bilo spomenuto da se zbog ograničenja radne memorije za čišćenje podataka originalni skup morao podijeliti na 7 jednakih, a slična je situacija i sa tokenizacijom. Ona naime zahtjeva još više radne memorije tako da se svaki od prethodno spomenutih 7 skupova morao podijeliti na još 3 podskupa kako bi tokenizacija bila uspješno izvršena, odnosno kako ne bi dolazilo do rušenja radne okoline. Tako od originalnog skupa podataka dolazimo do brojke od 21 podskupova svaki od kojih, kada su tokenizirani, zauzima 8,25 GB memorijskog prostora.

3.3.3. Kreiranje modela

Kako je arhitektura modela kojeg koristimo istovjetna originalnoj GPT-2 arhitekturi, samo ćemo preuzeti predtrenirani model sa Huggingface repozitorija kao što je prikazano u sljedećem programskog odlomku:

```
1  import tensorflow as tf
2  from transformers import TFGPT2LMHeadModel
3
4  lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
5      initial_learning_rate = 3e-3,
6      decay_steps= 1772,
7      decay_rate=0.85,
8      staircase=True
9      )
10
11 with strategy.scope():
12     model = TFGPT2LMHeadModel.from_pretrained(
13         "gpt2",
14         use_cache=False,
15         bos_token_id = tokenizer.bos_token_id,
16         pad_token_id=tokenizer.pad_token_id,
17         eos_token_id=tokenizer.eos_token_id,
18
19     )
20     metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
21
22     model.resize_token_embeddings(len(tokenizer))
23     optimizer = tf.keras.optimizers.Adam(lr_schedule)
24
25     model.compile(optimizer=optimizer, loss=model.hf_compute_loss, metrics=[metric])
```

U linijama 4 do 9 je definirano opadanje brzine učenja koje počinje od 3e-3 pa se nakon svakih 1772 koraka, što predstavlja 80% koraka svake epohe, eksponencijalno smanjuje, a u liniji 23 se može vidjeti da se to opadanje primjenjuje na *Adam* optimizator. Osim funkcije gubitka još će se pratiti točnost (engl. *accuracy*) modela, definirano u liniji 20. U liniji 22 se veličina ulaznih reprezentacija tokena postavljaju na duljinu tokenizatora te na kraju u liniji 25 se model kompilira. Za računanje funkcije gubitka koristi se funkcija definirana u modelu [5] koja zanemaruje ranije spomenute vrijednosti -100.

[5] Od početka pisanja ovog rada Keras je razvio vlastitu implementaciju *compute_loss* metode koja nadjačava onu koja dolazi s GPT-2 modelom. Kako bi se to izbjelo i osiguralo ispravno izračunanje funkcije mora se koristiti *hf_compute_loss* što se navodi kao privremeno rješenje i podložno je promjenama u budućnosti.

3.3.4. Treniranje

Postupak treniranja je, kako je spomenuto na početku poglavlja 3.3, uvelike definiran ograničenim računalnim i memorijskim resursima. Zbog ograničenog trajanja sjednice od 12 sati, stanje modela u određenim epohama se sprema kako bi se ponovnim pokretanjem sjednice nastavilo. Tako su u sljedećem odlomku programskog koda definirane takozvane *Callback* funkcije koje definiraju rano zaustavljanje treniranja (linije 2 do 4) u slučaju da se rezultat funkcije gubitka ne poboljša dvije epohe zaredom, te spremanje stanja (linije 5 do 11) modela nakon epohe koja je ostvarila poboljšanje kako bi se stanje moglo povratiti.

```
1  callbacks = [
2      tf.keras.callbacks.EarlyStopping(
3          monitor="val_loss", verbose=1, patience=2, restore_best_weights=True
4      ),
5      tf.keras.callbacks.ModelCheckpoint(
6          "putanja/do/direktorija/za/spremanje/modela/GPT2-Model_{epoch:02d}_{val_loss
               :.4f}.h5",
7          monitor="val_loss",
8          save_best_only=True,
9          save_weights_only = True,
10         verbose=1,
11     ),
12 ]
```

Nakon *Callback* funkcija slijedi još jedna prilagodba načina treniranja. Ovoga puta problem nije bio u ograničenim resursima već do treniranja na TPU-u. Google TPU koristi takozvani *Protocol Buffer* za serijalizaciju strukturiranih podataka neovisnu o programskom jeziku ili platformi. Do problema dolazi zbog toga što *Protocol Buffer* dolazi s limitom od 2 GB jer koristi 32-bitni integer (int32) za enkodiranje podataka što nije promijenjeno, i ne zna se hoće li biti promijenjeno, na 64-bitni integer ili unsigned 32-bitni integer (uint32). Problem je prijavljen za Tensorflow [6] i za PyTorch[7] platformu.

Ovaj je problem u kodu izbjegnut tako da se podaci dodatno podijele. Na kraju poglavlja 3.3.2 je bilo spomenuto da su datoteke s tokenziranim podacima veličine oko 8.25 GB pa da bi bile manje od 2 GB bilo ih je potrebno podijeliti na još 5 dijelova. Dakle, početni se skup podataka prvo podijelio na 7 dijelova kako bi se podaci očistili, zatim svaki od njih na još 3 kako bi se tokenizirali, a sada na kraju svaki od ta 3 na još 5 kako bi mogli biti pretvoreni u tenzore za treniranje. Time se od početnog skupa podataka dolazi na 105 podskupova.

Ta dodatna podjela skupa podataka i postupak treniranja su prikazani u sljedećem programskom odlomku:

[6] Poveznica na GitHub izvješće o greški: https://github.com/tensorflow/tensorflow/issues/24459
[7] Poveznica na GitHub izvješće o greški https://github.com/pytorch/xla/issues/1633

```
BATCH_SIZE_PER_REPLICA = 12
EPOCHS = 5

try:
    BATCH_SIZE = 12 * strategy.num_replicas_in_sync
except NameError as e:
    BATCH_SIZE = BATCH_SIZE_PER_REPLICA

data = Dataset.load_from_disk("/putanja/do/direktorija/
    datoteka_s_tokeniziranim_podacima")

for i in range (5):
  data_part = data.shard(num_shards = 5, index=i)
  data_part.set_format(type="python", columns=["input_ids", "attention_mask", "
      labels"])
  data_part = data_part.train_test_split(
      test_size=0.20, shuffle=True, seed=42, load_from_cache_file=True
  )

  train_tensor_inputs = tf.convert_to_tensor(data_part["train"]["input_ids"])
  train_tensor_labels = tf.convert_to_tensor(data_part["train"]["labels"])
  train_tensor_mask = tf.convert_to_tensor(data_part["train"]["attention_mask"])
  train = tf.data.Dataset.from_tensor_slices(
      (
          {"input_ids": train_tensor_inputs, "attention_mask": train_tensor_mask},
          train_tensor_labels,
      )
  )
  del train_tensor_inputs, train_tensor_labels, train_tensor_mask
  gc.collect()

  test_tensor_inputs = tf.convert_to_tensor(data_part["test"]["input_ids"])
  test_tensor_labels = tf.convert_to_tensor(data_part["test"]["labels"])
  test_tensor_mask = tf.convert_to_tensor(data_part["test"]["attention_mask"])
  test = tf.data.Dataset.from_tensor_slices(
      (
          {"input_ids": test_tensor_inputs, "attention_mask": test_tensor_mask},
          test_tensor_labels,
      )
  )

  del test_tensor_inputs, test_tensor_labels, test_tensor_mask
  gc.collect()

  BUFFER_SIZE = len(train)

  train_ds = (
      train.shuffle(BUFFER_SIZE).batch(BATCH_SIZE, drop_remainder=True)
  )
  test_ds = test.batch(BATCH_SIZE, drop_remainder=True)

  steps_per_epoch = int(BUFFER_SIZE // BATCH_SIZE)

```

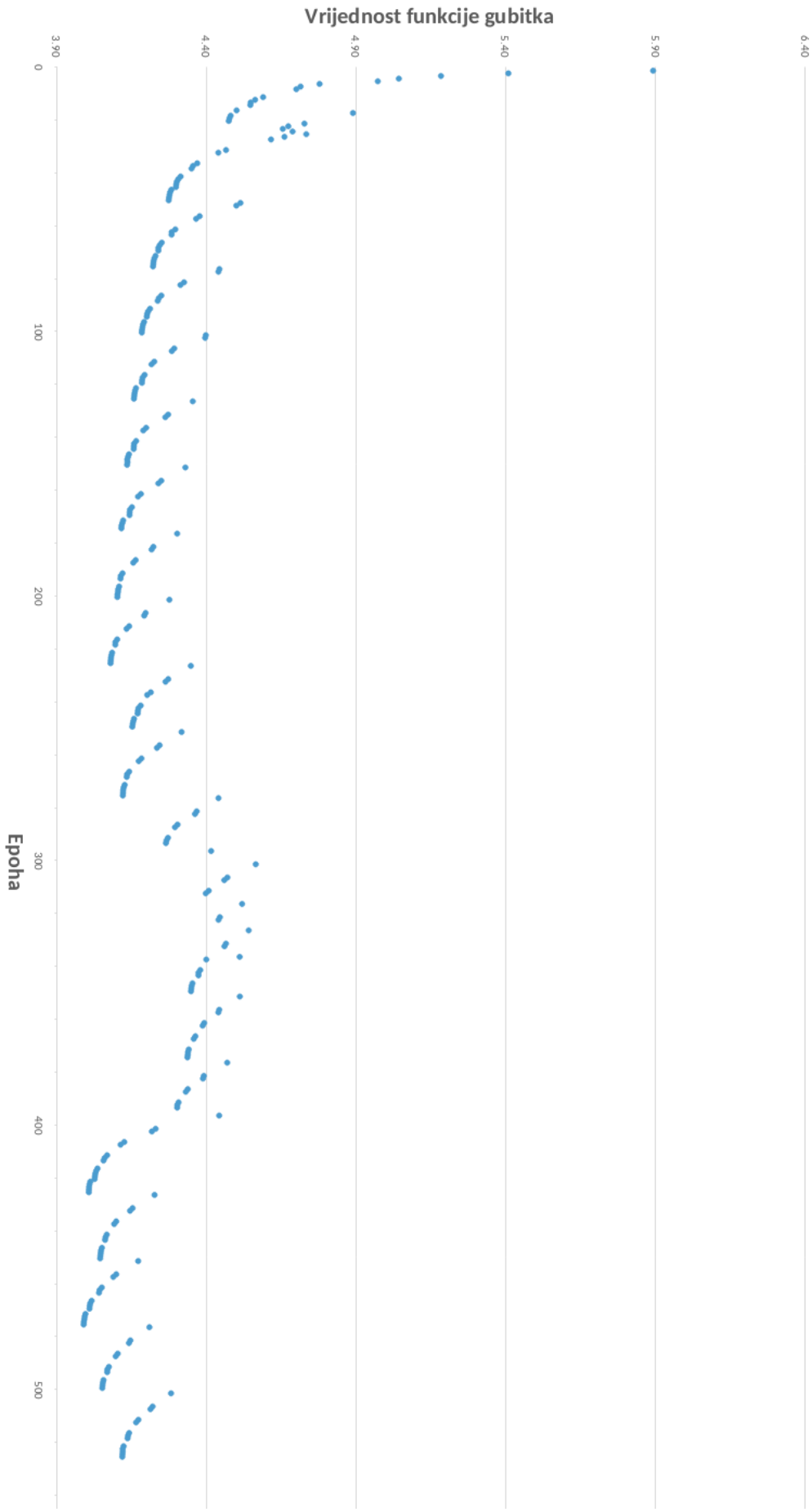
```
52  hist = model.fit(
53      train_ds,
54      validation_data=test_ds,
55      batch_size=BATCH_SIZE,
56      epochs=EPOCHS,
57      callbacks=callbacks,
58      verbose=1,
59  )
```

U ovom se programskom odlomku može primjetiti korištenje *shard* metode koja je dio Huggingface-ove Dataset klase. Ona razlama skup podataka na jednake dijelove i uzima onaj dio koji je specificiran indeksom. Od 12. do 16. linije može se uočiti definiranje formata skupa podataka te njegova podjela na skupove za treniranje i testiranje. 20% podataka je predviđeno za testiranje što je uobičajena praksa. Dalje se od linije 18 do linije 41 podaci pretvaraju u tenzore od kojih se onda kreira Tensorflow Dataset objekt. Sve se pomoćne varijable, nakon što više nisu potrebne, brišu kako je prikazano na linijama 27 i 40, a da bi se oslobodila radna memorija potrebno je "sakupiti smeće" što je prikazano u linijama 28 i 41.

U linijama 45 do 48 raniji se zapisi iz skupova podataka grupiraju u skupove u kojima će zajedno biti obrađeni tokom treniranja. Veličina tih skupova je određena količinom dostupnih računalnih resursa. Tako se empirijski pokazalo da jedna TPU jezgra može obraditi maksimalno 12 zapisa istovremeno, no ako je spajanje na Google TPU prošlo uspješno tada se taj broj množi s količinom dostupnih jezgri što je u ovom slučaju 8. Tako se dolazi do broja od 96 zapisa istovremeno. Provjera dostupnosti TPU jezgri i definiranje veličine skupova je prikazana u linijama 4 do 7 te u liniji 1.

Linije 52 do 59 prikazuju jednostavan način pozivanja *fit* metode Tensorflow objekta kojom započinje treniranje. Metodi su kao argumenti proslijeđeni skup podataka za treniranje, skup za testiranje, ranije definirane veličina grupe zapisa te *Callback* funkcije.

Proces se treniranja, dakle, izvodio za 105 podskupova podataka te 5 epoha po podskupu. Trajanje jedne epohe je, u pravilu, trajalo između 1700 i 1720 sekundi. Tako se za vrijeme trajanja jedne sjednice treniranje moglo odraditi na 5 podskupova, što u prosjeku iznosi nešto kraće od 12 sati. Iz toga se jednostavno može zaključiti da je cijeli proces treniranja ukupno trajao oko 10 i pol dana. Na slici 14 su prikazane vrijednosti funkcije gubitka nakon svake spremljene epohe. To znači da ako nije ostvareno poboljšanje dvije epohe zaredom, stanje modela se vraća na prvo prethodno stanje s ostvarenim poboljšanjem, a stanje bez poboljšanja nije prikazano na slici 14. Gotovo svako povećanje vrijednosti funkcije gubitka na slici dolazi zbog početka treniranja na novom podskupu podataka. Na slici se također može jasno uočiti 21 identični uzorak, uz malobrojne iznimke, kretanja vrijednosti funkcije gubitka što je rezultat opadajuće vrijednosti brzine učenja koja bi početkom svake sjednice imala istu vrijednost. Detaljne vrijednosti po podskupovima i epohama su priložene u prilogu. 1.



Slika 14: Vrijednosti funkcije gubitka prilikom treniranja (Vlastita izrada)

4. Evaluacija

Kao predtrenirani model, ovaj se model može primijeniti za niz različitih zadataka iz područja obrade prirodnog jezika te je javno dostupan na Huggingface repozitoriju pod nazivom "GPcroaT" [1] bilo kome tko ga želi prilagoditi vlastitim ciljanim zadacima.

4.1. Jezično modeliranje

Najdirektniji način evaluacije predtreniranog modela je evaluacija onog zadatka na kojemu je model bio treniran uz korištenje drugih skupova podataka. Time se, kako Radford, Wu, Child i dr. [46] navode, jasno može razumijeti uspješnost prenošenja domene, odnosno parametara, bez prethodnog prilagođavanja na novom skupu podataka. Kako model funkcionira na razini bajtova, pretprocesuiranje nije nužno pa tako ne postoje prepreke koje bi spriječile direktnu primjenu na novim podacima.

Ocjenjivanje uspješnosti u generiranju teksta nije jednostavan zadatak pa je tako jedna od mogućih metrika ljudska procjena generiranih tekstova. Tu ispitanici mogu ocjenjivati subjektivni dojam o koherentnosti teksta ili njegovoj prihvatljivosti. Gornju granicu bi tako predstavljao tekst napisan od strane ljudi koji bi bio evaluiran kao kontrolni skup. Time bi se otklonili neki nedostaci takve metrike no najveći problem je i dalje skalabilnost i praktičnost takvog pristupa. Iz toga se razloga umjesto ljuske procjene može koristiti metrika zbunjenosti (engl. *Perplexity*), unakrsne entropije (engl. *Cross-entropy*) ili broj bitova po znaku (engl. *bits-per-character*, BPC).

Radford, Wu, Child i dr. [46] kao metriku ocjenjivanja GPT-2 modela koriste zbunjenost koja mjeri razinu nesigurnosti jezičnog modela prilikom generiranja novog tokena te računa prosjek takvih vrijednosti kako bi se dobio rezultat za dulje tekstove. Tako intuitivno slijedi da manja vrijednost zbunjenosti znači višu sigurnost pri predviđanju sljedećeg tokena u nizu, odnosno bolja je mogućnost generalizacije. Slijedom toga usporedba različitih modela ima smisla samo pod uvjetom da koriste isti tokenizator. Stoga, takva usporedba za ovaj model neće biti moguća jer je izrađen posebni tokenizator, ali će biti moguće provjeriti zbunjenost na podacima s kojima se model nije prije susreo te bez prethodnog finog prilagođavanja, na takozvanom *zero-shot* pristupu.

Novi skupovi podataka koji će se koristiti se odnose na *OSCAR* (engl. *Open Super-large Crawled ALMAnaCH coRpus, otvorena super-velika indeksirana almanah zbirka*) [66] i *Wikipedia* [67] skupove. OSCAR je, kako ime govori, izrazito velika višejezična zbirka dobivena klasifikacijom i filtriranjnem *CommonCrawl* skupa koristeći goclassy [68] arhitekturu. Wikipedia skup podataka je, kako pak njegovo ime govori, sačinjen od članaka na Wikipediji.

OSCAR skup se sastoji od 321,484 zapisa koji uglavnom odgovaraju jednoj rečenici ili kraćem odlomku. Duplikati su u tom skupu uklonjeni kao i zapisi kraći od 100 znakova gdje svaki znak mora biti u UTF-8 formatu. Kako zapisi dolaze iz veoma različitih izvora s različitim

[1] Poveznica na projekt: `https://huggingface.co/domsebalj/GPcroaT`

kontekstima te se radi o kraćim zapisima, ne očekuju su dobri rezultati modela. Wikipedia skup se pak sastoji od 198,702 zapisa, ali svaki zapis odgovara jednom članku gdje su nepotrebni odlomci, kao odlomak s referencama, uklonjeni. Tako je svaki zapis uglavnom dulji od zapisa iz OSCAR skupa te je kontekstualno povezaniji. Kako bi se izbjeglo predugo zadržavanje na zapisima s istom temom te utjecaj duljine zapisa na uspješnost jezičnog modeliranja, iz svakog se od njih koristi samo prvih 511 tokena, odnosno jednak broj kao kod treniranja modela. Tih prvih 511 tokena se u svakom zapisu uglavnom odnosi na sažetak članka tako da model i dalje prolazi kroz većinu ključnih riječi neke specifične teme, a koje će se tada pojavljivati prvi put, pa tako možemo očekivati poprilično loše mogućnosti generalizacije.

Tablica 6: Zbunjenost modela na novim skupovima podataka bez prilagođavanja (Osim trening vrijednosti koja je dodana radi usporedbe)

	Trening	OSCAR	Wikipedia
GPcroaT	61.6147	30490.59	31287.95

(Vlastita izrada)

Ove je rezultate najlakše interpretirati na primjeru skupa podataka za treniranje. Prilikom odabira novog tokena u nizu, model u prosjeku odabire, odnosno zbunjen je, između 61.6147 različitih tokena. Dalje se može uočiti velika razlika između rezultata prilikom treniranja i rezultata na novim skupovima podataka. Jedan dio tih rezultata opravdava razlika strukture podataka za treniranje i novih skupova podataka, no također se može zaključiti da je model trebao biti dodatno treniran.

Premda 11 i pol dana zvuči puno, na slici 14 se može uočiti da, iako se približavaju, većina skupova nije dosegla najnižu moguću vrijednost funkcije gubitka prije uvođenja novog podskupa podataka. Jedan od uzroka tome leži i u načinu treniranja, odnosno korištenju 105 odvojenih podskupova podataka. Tako se može pretpostaviti da bi se ponovnim treniranjem na istom skupu podataka, uz izmiješane zapise kako bi se razlikovali od prvog prolaza, mogli ostvariti još bolji rezultati. No ako se uzme u obzir da treniranje ovakvih modela može koštati nekoliko desetaka tisuća dolara, ovaj model, izrađen korištenjem izrazito jeftinih resursa [2], ostvaruje dovoljno dobre rezultate.

[2] Plaćena je mjesečna pretplata u iznosu 10 dolara radi višeg prioriteta pri dodjeli resursa no model se mogao trenirati i bez nje uz malo veći broj poteškoća

4.2. Prevođenje

Sljedeća evaluacija, koju su koristili i Radford, Wu, Child i dr. [46], je strojno prevođenje. Oni su uvjetovali model na primjerima rečenica u obliku $Tekst\ na\ engleskom\ =\ Tekst\ na\ francuskom$ te kasnije izbacili desnu stranu tog izraza, odnosno tekst na francuskom, pa je model generirao prijevod. U ovom je radu korištena ista takva metoda no umjesto engleskog i francuskog jezika korišteni su hrvatski i engleski jezik.

Za ovaj zadatak je korišten ccmatrix [69] [70] skup podataka, odnosno 10% njegovih zapisa za hrvatski i engleski jezik. Zapisi dolaze u obliku parova rečenica gdje je jedna rečenica na hrvatskom, a druga njen prijevod na engleskom. Tokom tokenizacije, ti su parovi rečenica postavljani u oblik $Tekst\ na\ hrvatskom\ =\ Tekst\ na\ engleskom$ pa se zatim mreža prilagođavala na oko 1,5 milijuna takvih zapisa. Slijedi primjer nekoliko takvih zapisa za treniranje:

- Idealna težina odgovara tjelesnoj težini postignutoj na kraju skladnog razvoja. = The ideal weight corresponds to the body weight reached at the end of a harmonious development.
- Ali jao! „Jadni bogati“ će proći kroz strašna iskustva. = But alas! "the poor rich" will pass through terrible experiences.
- Tama se vidjelo iz Mainea u New Jersey, a ono što je tada gotovo polovica Amerike bilo u strahu i kaosu. = The darkness was witnessed from Maine to New Jersey, plunging what was then almost half of America into fear and chaos.

U trećem se od ovih primjera može primijetiti nekolicina gramatičkih grešaka, a drugom dijelu rečenice jednostavno nestaje koherentnost. To pokazuje kako ni svi zapisi u podacima za treniranje nisu savršeni prijevodi te će to isto imati utjecaja na preformanse modela, odnosno postoji šansa da model zbog krivih prijevoda u podacima za treniranje nauči krivo prevoditi.

Nakon prilagodbe, modelu je predstavljen zapis u obliku $Tekst\ na\ hrvatskom\ =$ te se metodom pohlepnog dekodiranja (engl. *greedy decoding)* generirao nastavak zapisa, odnosno prijevod teksta na engleski jezik. Metoda pohlepnog dekodiranja je najdirektniji pristup generiranju tokena u kojem se uvijek uzima onaj token sa najvećom pripadajućom vjerojatnošću. U slučajevima generiranja većeg teksta na jednom jeziku ovakva metoda bi ostvarivala poprilično jednolike, ponavljajuće i dosadne tekstove, no za zadatak prevođenja njeno korištenje ima najviše smisla. Tako se za prethodno nabrojane primjere dobivaju sljedeći izlazi:

- Idealna težina odgovara tjelesnoj težini postignutoj na kraju skladnog razvoja. = The ideal weight corresponds to the body weight achieved at the end of a harmonious development.
- Ali jao „Jadni bogati“ će proći kroz strašna iskustva. = But the poor rich will go through terrible experiences.
- Tama se vidjelo iz Mainea u New Jersey, a ono što je tada gotovo polovica Amerike bilo u strahu i kaosu. = The darkness was seen from Maine in New Jersey, and what was almost half of America was in fear and chaos.

Ti su prijevodi prilično razumljivi, no kod drugog možemo primijetiti da model nije uspješno naučio koristiti pravopisne znakove, odnosno navodnike, a može se pretpostaviti da je razlog tome mali broj zapisa u kojima se takvi znakovi pojavljuju. U trećem se primjeru može uočiti da greške koje se pojavljuju za tekst na hrvatskom jeziku ne utječu toliko koliko bi se isprva pomislilo, ali korištenje riječi "u" umjesto "do" u "... iz Mainea u New Jersey,..." u rečenici na hrvatskom jeziku poprilično doprinosi promjeni informacije u rečenici.

Za određivanje uspješnosti prijevoda korištena je BLEU [71] (engl. *BiLingual Evaluation Understudy*, Dvojezično evaluacijsko vrednovanje) metoda koja dodjeljuje ocjenu od 0 do 1. Ova metoda funkcionira na način da računa broj odgovarajućih parova n-grama u generiranom i referentnom prijevodu, gdje 1-gram ili unigram uspoređuje svaki token posebno, a 2-gram ili bigram riječne parove. Usporedbe se uvijek odrađuju neovisno o položaju riječi u rečenici. Što je veći broj poklapanja n-grama u generiranom i referentnom prijevodu, to je prijevod bolji. BLEU ocjene se isto kao i ocjena zbunjenosti ne bi trebale uspoređivati s drugim modelima, a pogotovo ne drugim jezicima jer ocjene mogu varirati čak i na istom skupu podataka. Google stoga koristi sljedeću skalu za interpretaciju BLEU rezultata:

Tablica 7: Google-ova skala za interpretaciju BLEU rezultata

BLEU bodovi	Interpretacija
< 10	Skoro beskorisni prijevodi
10 - 19	Teško je shvatiti suštinu
20 - 29	Suština je jasna, ali uz značajne gramatičke greške
30 - 39	Razumljivi, dobri prijevodi
40 - 49	Visokokvalitetni prijevodi
50 - 60	Visokokvalitetni, adekvatni i tečni prijevodi
> 60	Kvaliteta često nadmašuje ljudske prijevode

(Vlastita izrada prema: Google developers, 2022 [72])

Bodovanje BLEU metodom se u Pythonu odrađuje korištenjem NLTK biblioteke spomenute u poglavlju 3.1 koja dolazi sa nizom metoda vezanih uz BLEU bodovanje. Bodovi za ranije navedene primjere zapisa su prikazani na tablici 8. U tim primjerima možemo vidjeti da BLEU bodovanje više preferira nedostatak neke riječi od korištenja krive. Tako drugi zapis ostvaruje bolje bodove od prvog, iako je prijevod u prvom primjeru gotovo identičan originalu, izuzev jedne riječi. To nam govori da BLEU metoda nije savršeni indikator uspješnosti prevođenja već samo provjerava poklapanje n-gramova riječi u originalnom i generiranom prijevodu.

Kako bi se ublažili negativni efekti povremenih iznimaka, kao što je drugi primjer u tablici 8, za evaluaciju modela treba uzeti što je moguće veći skup podataka. Za evaluiranje ovog modela je od skupa podataka uzet uzorak od 1000 zapisa s kojima se mreža nije susrela u finom prilagođavanju. Tu se radi o najvećem broju zapisa koji je bilo moguće uzeti s obzirom na ograničenje korištenih alata i računalnih resursa. Zatim je korištenjem *sentence_bleu* funkcije izračunata BLEU ocjena za svaku od njih. Od tih se 1000 ocjena izračunala prosječna ocjena.

Tablica 8: BLEU bodovi za primjere rečenica

Original	Generirani prijevod	BLEU bodovi
The ideal weight corresponds to the body weight reached at the end of a harmonious development.	The ideal weight corresponds to the body weight achieved at the end of a harmonious development.	36.82
But alas! "the poor rich" will pass through terrible experiences.	But the poor rich will go through terrible experiences.	48.36
The darkness was witnessed from Maine to New Jersey, plunging what was then almost half of America into fear and chaos.	The darkness was seen from Maine in New Jersey, and what was almost half of America was in fear and chaos.	30.68

(Vlastita izrada)

Treba ponovo napomenuti da ta ocjena ne predstavlja postotak potpuno točno prevedenih rečenica, već prosječnu ocjenu ranije spomenutih poklapanja n-gramova u svim rečenicama. Ta ocjena za GPcroaT model iznosi 0.4082, odnosno 40.82 po Google skali, što ju postavlja u razred visokokvalitetnih prijevoda.

4.3. Analiza sentimenta

Analiza sentimenta je zadatak klasifikacije u kojemu je cilj modela klasificirati tekstne zapise u neki broj ranije određenih klasa gdje svaka od njih predstavlja jedan sentiment, odnosno osjećaj. Tako se ovdje radi o evaluaciji koja se, za razliku od prethodne dvije, može koristiti za usporedbu GPcroaT-a s ostalim modelima. Ptiček [73] je napravila usporedbu 4 predtreniranih modela temeljenih na BERT arhitekturi koji su, uz GPTcroaT, pobrojani na tablici 9. Koristeći iste podatke te iste parametre, te je modele stoga moguće kvalitetno usporediti s GPcroaT-om kako bi se bolje predočile njegove preformanse.

Tablica 9: Uspoređeni modeli na zadatku analize sentimenta.

Model	Slojeva	d_{model}	Parametara
mBERT	12	768	179 Mil.
DistilBERT	6	768	134 Mil.
XML-RoBERTa	12	768	270 Mil.
CroSloEngular	12	768	110 Mil.
GPcroaT	12	768	124 Mil.

(Vlastita izrada prema: Ptiček, 2021)

U zadatku iz [73] korišten je skup Twitter objava na hrvatskom jeziku kojeg su prikupili Mozetič, Grčar i Smailović [74]. Svaki se zapis u skupu sastoji od URL-a jedne objave na Twitteru te im je jedan ljudski anontator prirodao oznaku sentimenta: pozitivan, negativan ili

neutralan. Skup se originalno sastojao od 97921 zapisa no zbog korištenja URL-a neki zapisi više nisu bili dostupni pa je broj pao na 47276. Neki su zapisi bili označeni dva puta različitim sentimentima pa se nakon njihovih uklanjanja broj zapisa dodatno spustio na 35880 od kojih 18146 predstavlja pozitivni, 8492 negativni, a 9242 neutralni sentiment. Ptiček [73] je dalje iz zapisa uklonila sve URL-ove, korisnička imena i sve posebne znakove te je pretvorila sva slova u mala. Takvi pretprocesuirani podaci su bili ustupljeni za testiranje i usporedbu GPcroaT modela. Svi su modeli trenirani korištenjem 80%, a testirani na stratificiranom uzorku od 20% podataka.

Kako je ranije navedeno, svaki je model treniran korištenjem jedne kombinacije hiperparametara prikazanih u prva dva stupca na tablici 10. Ti parametri dolaze kao preporuka Devlin, Chang, Lee i dr. iz [35], a jedino je veličina skupa za istovremenu obradu promijenjena iz 16 u 32 zapisa te se u toj veličini koristi sa svim kombinacijama stope učenja i broja epoha. Uz te parametre Ptiček [73] je koristila AdamW optimizator iz Transformers biblioteke. Taj je optimizator dostupan samo uz PyTorch pa je umjesto njega u ovom radu za testiranje GPcroaT-a korišten AdamWeightDecay koji bi trebao biti njegov pandan za korištenje uz Tensorflow. Osim navedenoga svi su parametri i uvjeti testiranja jednaki, kao i činjenica da su oba eksperimenta izvršena u Google Collaboratory okružju na GPU uređaju.

Za usporedbu su korišteni ponderirani F1 rezultati, a prikazani su u tablici 10 gdje je najbolji rezultat svakog modela podebljan. CroSloEngular ostvaruje najbolje rezultate u svim kombinacijama što je očekivano uzme li se u obzir da je za treniranje koristio skoro 5 puta više riječi i dvostruko dulje vrijeme na TPU uređaju od GPcroaT modela, a kako je objašnjeno u poglavlju 2.5.2. Također, prednost CroSloEngulara, kako Ptiček [73] navodi, dolazi i iz činjenica da se skup podataka na kojemu su izvršena testiranja sastoji od zapisa na kolokvijalnom jeziku s dosta ubačenih riječi na engleskom, a CroSloEngular je učen na oba jezika koristeći i standardni i kolokvijalni jezik. U usporedbi sa mBERT-om i DistilBERT-om, GPcroaT ostvaruje bolje rezultate ponajprije zbog toga što je treniran za samo jedan jezik.

Također, zanimljivo je za primijetiti da GPcroaT najbolje rezultate uvijek postiže u zadnjoj, odnosno 4. epohi, što daje za pretpostaviti da bi modelu koristilo dodatno treniranje, ali povećavanjem broja epoha na ovom zadatku se događa pretjerano prilagođavanje (engl. *overfitting*). Slijedeći pretpostavku iz zadnjeg odlomka poglavlja 4.1, ponovo se može pretpostaviti da bi rezultati bili bolji uz dodatno predtreniranje modela kako bi se postigla bolja mogućnost generalizacije.

Tablica 10: F1 rezultati za uspoređene modele po kombinacijama hiperparametara

Stopa učenja	Broj epoha	mBERT	DistilBERT	XML-RoBERTa	CroSloEngular	GPcroaT
2e-5	2	64.58%	63.74%	69.41%	71.48%	57.69%
2e-5	3	66.00%	65.15%	70.31%	71.82%	61.39%
2e-5	4	66.08%	65.13%	**70.65%**	71.60%	63.55%
3e-5	2	63.80%	62.71%	69.27%	70.95%	61.09%
3e-5	3	65.70%	64.03%	70.38%	**72.01%**	64.69%
3e-5	4	**66.38%**	65.15%	70.18%	71.30%	66.52%
5e-5	2	65.47%	65.04%	68.08%	71.18%	66.05%
5e-5	3	64.64%	**66.03%**	68.76%	71.75%	66.40%
5e-5	4	65.79%	65.47%	66.18%	70.70%	**67.46%**

(Vlastita izrada prema: Ptiček, 2021.)

5. Budući rad

Kako je navedeno u poglavljima 4.1 i 4.3 smatra se da model nije dostigao maksimalne performanse koje nudi njegova arhitektura. Temeljem toga se pretpostavlja da bi nastavak predtreniranja nad istim skupom podataka omogućilo bolje rezultate. Slijedom toga mogu se pretpostaviti bolji rezultati i promjenom tehnike, odnosno načina, predtreniranja. Predtreniranje na većem broju epoha s kraćim sekvencama čemu slijedi manji broj epoha sa duljim sekvencama po uzoru na CroSloEngular ili neka druga kombinacija hiperparametara bi mogla dati dobre rezultate uz smanjeno korištenje računalnih resursa.

Osim načina predtreniranja i pretpostavaka vezanih uz model, daljnji rad bi bio koristan u području prikupljanja podataka. Kako je navedeno u poglavlju 3.2, podaci s kojima je bio predtreniran ovaj model su rezultat istraživanja iz 2014. godine. Korištenje tako starog skupa podataka za predtreniranje ima svoje mane jer sam model za generiranje teksta prenosi pristranosti s kojima se susreo u podacima za predtreniranje. Kako se društvo mijenja, mijenja se i sadržaj na internetu, stoga su suvremeni podaci nužni kako bi model držao korak s vremenom.

Nastavljeno na podatke, daljnji rad bi bio nužan u području prikupljanja i moderiranja podataka za specifične zadatke. GPcroaT, model predstavljen u ovom radu, nije mogao biti evaluiran na svim zadacima kao originalni GPT-2 model baš zbog nedostatka javno dostupnih skupova podataka za te specifične zadatke. Zadaci kao sažimanje teksta (engl. *Summarization*), modeliranje teme (engl. *Topic modeling*), parafraziranje (engl. *Paraphrasing*), razumijevanje pročitanog (engl. *Text comprehension*), zdravorazumno zaključivanje (engl. *Commonsense reasoning*) i razgovorno odgovaranje na pitanja (engl. *Conversational question answering*) su sve zadaci u kojima bi modeli temeljeni na Transformer arhitekturi mogli ostvarivati vrlo dobre rezultate, ali skupovi podataka za takve zadatke, ako postoje, nisu lako dostupni.

Slijedom svega navedenoga, zaključuje se da bi budući rad trebao ići u smjeru testiranja različitih tehnika predtreniranja koje mogu raditi u skladu s ograničenjima jeftinih ili slobodno dostupnih računalnih resursa, kreiranja lako dostupnih skupova podataka za predtreniranje te za specifične zadatke za koje lako dostupni skupovi već ne postoje.

6. Zaključak

U ovom je radu predstavljen povijesni razvoj predtreniranih modela i način rada suvremenih arhitektura. Izrađen je vlastiti predtrenirani model pod nazivom GPcroaT temeljen na GPT-2 arhitekturi te je evaluiran na nekolicini zadataka. Time se može odgovoriti na pitanje postavljeno u uvodu ovoga rada, a to je da iako je za predtreniranje jezičnog modela potrebna velika količina računalnih resursa, moguće je korištenjem besplatnih ili jako jeftinih alata kreirati model za slabo zastupljen jezik. U pogledu performansi, velike tvrtke s većom količinom resursa su u dalekoj prednosti, ali inovativnim pristupima predtreniranju i suočavanju s ograničenjima besplatnih resursa moguće je dobiti zadovoljavajuće rezultate. To ne mijenja činjenicu da te tvrtke ostvaruju daljnji napredak s još većim modelima, treniranim na još više podataka, no inovativnim pristupom problemu moći će se riješiti neki problemi koje te tvrtke rješavaju povećanom uporabom računalnih resursa.

To dovodi do drugog zaključka, a to je da za jezik sa manje govornika poput hrvatskog treba uložiti napore kako si se kreirali skupovi podataka za specifične zadatke poput sažimanja teksta ili ostalih spomenutih u poglavlju 5. Do samih nestrukturiranih podatka nije pretjerano teško doći, no skupovi za čije je kreiranje potreban ljudski rad predstavljaju problem. Tako bi se predtrenirani modeli mogli lako i brzo prilagođavati za nove zadatke te biti primijenjeni u raznim granama gospodarstva. Osim toga, lako dostupni skupovi podataka za specifične zadatke omogućuju njihovo korištenje za sve nove modele te na taj način mogu postati česte metrike za uspoređivanje performansi predtreniranih modela.

Popis literature

[1] Y. Bengio, R. Ducharme i P. Vincent, „A Neural Probabilistic Language Model," *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich i V. Tresp, ur., sv. 13, MIT Press, 2000. adresa: `https://proceedings.neurips.cc/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf`.

[2] A. Krizhevsky, I. Sutskever i G. E. Hinton, „ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou i K. Weinberger, ur., sv. 25, Curran Associates, Inc., 2012.

[3] Y. Kim, „Convolutional neural networks for sentence classification. arXiv 2014," *arXiv preprint arXiv:1408.5882*, 2019.

[4] N. Kalchbrenner, E. Grefenstette i P. Blunsom, „A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[5] I. Sutskever, O. Vinyals i Q. V. Le, „Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence i K. Weinberger, ur., sv. 27, Curran Associates, Inc., 2014.

[6] J. Donahue, L. Anne Hendricks, S. Guadarrama i dr., „Long-term recurrent convolutional networks for visual recognition and description," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015., str. 2625–2634.

[7] P. Liu, X. Qiu i X. Huang, „Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[8] R. Caruana, „Multitask learning: A knowledge-based source of inductive bias1," *Proceedings of the Tenth International Conference on Machine Learning*, Citeseer, 1993., str. 41–48.

[9] R. Collobert i J. Weston, „A unified architecture for natural language processing: Deep neural networks with multitask learning," *Proceedings of the 25th international conference on Machine learning*, 2008., str. 160–167.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu i P. Kuksa, „Natural language processing (almost) from scratch," *Journal of machine learning research*, sv. 12, br. ARTICLE, str. 2493–2537, 2011.

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

[12] S. Thrun i L. Pratt, „Learning to learn: Introduction and overview," *Learning to learn*, Springer, 1998., str. 3–17.

[13] X. Han, Z. Zhang, N. Ding i dr., „Pre-trained models: Past, present and future," *AI Open*, sv. 2, str. 225–250, 2021., ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.08.002. adresa: https://www.sciencedirect.com/science/article/pii/S2666651021000231.

[14] R. Johnson i T. Zhang, „A high-performance semi-supervised learning method for text chunking," *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 2005., str. 1–9.

[15] A. Argyriou, T. Evgeniou i M. Pontil, „Multi-task feature learning," *Advances in neural information processing systems*, sv. 19, 2006.

[16] W. Dai, G.-R. Xue, Q. Yang i Y. Yu, „Co-clustering based classification for out-of-domain documents," *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007., str. 210–219.

[17] R. Raina, A. Battle, H. Lee, B. Packer i A. Y. Ng, „Self-taught learning: transfer learning from unlabeled data," *Proceedings of the 24th international conference on Machine learning*, 2007., str. 759–766.

[18] N. D. Lawrence i J. C. Platt, „Learning to learn with the informative vector machine," *Proceedings of the twenty-first international conference on Machine learning*, 2004., str. 65.

[19] T. Evgeniou i M. Pontil, „Regularized multi–task learning," *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004., str. 109–117.

[20] E. V. Bonilla, K. Chai i C. Williams, „Multi-task Gaussian process prediction," *Advances in neural information processing systems*, sv. 20, 2007.

[21] J. Gao, W. Fan, J. Jiang i J. Han, „Knowledge transfer via multiple model local structure mapping," *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008., str. 283–291.

[22] J. Pennington, R. Socher i C. Manning, „GloVe: Global Vectors for Word Representation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, listopad 2014., str. 1532–1543. DOI: 10.3115/v1/D14-1162. adresa: https://aclanthology.org/D14-1162.

[23] K. He, X. Zhang, S. Ren i J. Sun, „Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016., str. 770–778.

[24] K. Simonyan i A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014. DOI: 10.48550/ARXIV.1409.1556. adresa: https://arxiv.org/abs/1409.1556.

[25] B. McCann, J. Bradbury, C. Xiong i R. Socher, „Learned in translation: Contextualized word vectors," *Advances in neural information processing systems*, sv. 30, 2017.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado i J. Dean, „Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, sv. 26, 2013.

[27] M. E. Peters, M. Neumann, M. Iyyer i dr., *Deep contextualized word representations*, 2018. DOI: 10.48550/ARXIV.1802.05365. adresa: https://arxiv.org/abs/1802.05365.

[28] J. Turian, L. Ratinov i Y. Bengio, „Word representations: a simple and general method for semi-supervised learning," *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010., str. 384–394.

[29] M. Peters, M. Neumann, M. Iyyer i dr., „Deep contextualized word representations. arXiv 2018," *arXiv preprint arXiv:1802.05365*, sv. 12, 1802.

[30] A. Vaswani, N. Shazeer, N. Parmar i dr., „Attention is all you need," *Advances in neural information processing systems*, sv. 30, 2017.

[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov i Q. V. Le, „Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, sv. 32, 2019.

[32] Y. Liu, M. Ott, N. Goyal i dr., „Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[33] M. Lewis, Y. Liu, N. Goyal i dr., „Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[34] C. Raffel, N. Shazeer, A. Roberts i dr., „Exploring the limits of transfer learning with a unified text-to-text transformer.," *J. Mach. Learn. Res.*, sv. 21, br. 140, str. 1–67, 2020.

[35] J. Devlin, M.-W. Chang, K. Lee i K. Toutanova, „Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[36] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever i dr., „Improving language understanding by generative pre-training," 2018.

[37] K. Cho, B. Van Merriënboer, C. Gulcehre i dr., „Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[38] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu i J. Gao, „Deep learning–based text classification: a comprehensive review," *ACM Computing Surveys (CSUR)*, sv. 54, br. 3, str. 1–40, 2021.

[39] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink i J. Schmidhuber, „LSTM: A search space odyssey," *IEEE transactions on neural networks and learning systems*, sv. 28, br. 10, str. 2222–2232, 2016.

[40] K. Cho, B. Van Merriënboer, C. Gulcehre i dr., „Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[41] M.-T. Luong, H. Pham i C. D. Manning, „Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[42] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola i E. Hovy, „Hierarchical attention networks for document classification," *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016., str. 1480–1489.

[43] C. d. Santos, M. Tan, B. Xiang i B. Zhou, „Attentive pooling networks," *arXiv preprint arXiv:1602.03609*, 2016.

[44] J. L. Ba, J. R. Kiros i G. E. Hinton, „Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[45] P. J. Liu, M. Saleh, E. Pot i dr., „Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[46] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever i dr., „Language models are unsupervised multitask learners," *OpenAI blog*, sv. 1, br. 8, str. 9, 2019.

[47] T. B. Brown, B. Mann, N. Ryder i dr., „Language Models are Few-Shot Learners," 2020. arXiv: `2005.14165 [cs.CL]`.

[48] W. L. Taylor, „"Cloze procedure": A new tool for measuring readability," *Journalism quarterly*, sv. 30, br. 4, str. 415–433, 1953.

[49] A. Williams, N. Nangia i S. R. Bowman, „A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[50] P. Rajpurkar, J. Zhang, K. Lopyrev i P. Liang, „Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[51] N. Ljubešić i D. Lauc, „BERTić - The Transformer Language Model for Bosnian, Croatian, Montenegrin and Serbian," *Proceedings of the 8th BSNLP Workshop on Balto-Slavic Natural Language Processing*, 2021., str. 37–42.

[52] M. Ulčar i M. Robnik-Šikonja, „FinEst BERT and CroSloEngual BERT," *Text, Speech, and Dialogue*, P. Sojka, I. Kopeček, K. Pala i A. Horák, ur., Cham: Springer International Publishing, 2020., str. 104–111, ISBN: 978-3-030-58323-1.

[53] N. Ljubešić i K. Dobrovoljc, „What does Neural Bring? Analysing improvements in morphosyntactic annotation and lemmatisation of Slovenian, Croatian and Serbian," *Proceedings of the 7th workshop on balto-slavic natural language processing*, 2019., str. 29–34.

[54] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma i R. Soricut, „Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[55] L. Dong, N. Yang, W. Wang i dr., „Unified language model pre-training for natural language understanding and generation," *Advances in Neural Information Processing Systems*, sv. 32, 2019.

[56] K. Song, X. Tan, T. Qin, J. Lu i T.-Y. Liu, „Mass: Masked sequence to sequence pre-training for language generation," *arXiv preprint arXiv:1905.02450*, 2019.

[57] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer i O. Levy, „Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, sv. 8, str. 64–77, 2020.

[58] K. Clark, M.-T. Luong, Q. V. Le i C. D. Manning, „Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[59] W. Fedus, B. Zoph i N. Shazeer, *Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*, 2021.

[60] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper i B. Catanzaro, „Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[61] S. Rajbhandari, J. Rasley, O. Ruwase i Y. He, „Zero: Memory optimizations toward training trillion parameter models," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2020., str. 1–16.

[62] J. Ren, S. Rajbhandari, R. Y. Aminabadi i dr., „{ZeRO-Offload}: Democratizing {Billion-Scale} Model Training," *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021., str. 551–564.

[63] N. Ljubešić i F. Klubička, *Croatian web corpus hrWaC 2.1*, Slovenian language resource repository CLARIN.SI, 2016. adresa: `http://hdl.handle.net/11356/1064`.

[64] P. Halácsy, A. Kornai i C. Oravecz, „HunPos-an open source trigram tagger," 2007.

[65] R. Sennrich, B. Haddow i A. Birch, „Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[66] P. J. Ortiz Su'arez, L. Romary i B. Sagot, „A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, srpanj 2020., str. 1703–1714. adresa: `https://www.aclweb.org/anthology/2020.acl-main.156`.

[67] Wikimedia Foundation. „Wikimedia Downloads." (20.08.2022.), adresa: `https://dumps.wikimedia.org`.

[68] P. J. Ortiz Suárez, B. Sagot i L. Romary, „Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures," en, P. Bański, A. Barbaresi, H. Biber i dr., ur., serija Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, Mannheim: Leibniz-Institut für Deutsche Sprache, 2019., str. 9–16. DOI: `10.14618/ids-pub-9021`. adresa: `http://nbn-resolving.de/urn:nbn:de:bsz:mh39-90215`.

[69] H. Schwenk, G. Wenzek, S. Edunov, E. Grave i A. Joulin, *CCMatrix: Mining Billions of High-Quality Parallel Sentences on the WEB*, 2019. DOI: `10.48550/ARXIV.1911.04944`. adresa: `https://arxiv.org/abs/1911.04944`.

[70] A. Fan, S. Bhosale, H. Schwenk i dr., *Beyond English-Centric Multilingual Machine Translation*, 2020. DOI: `10.48550/ARXIV.2010.11125`. adresa: `https://arxiv.org/abs/2010.11125`.

[71] K. Papineni, S. Roukos, T. Ward i W.-j. Zhu, „BLEU: a Method for Automatic Evaluation of Machine Translation," 2002., str. 311–318.

[72] Google developers. „AutoML: Evaluating models documentation." (2022.), adresa: https://cloud.google.com/translate/automl/docs/evaluate#automl_translate_get_model_evaluation-python (pogledano 25. 8. 2022.).

[73] M. Ptiček, „How good BERT based models are in sentiment analysis of Croatian tweets: comparison of four multilingual BERTs," *Central European Conference on Information and Intelligent Systems*, Faculty of Organization i Informatics Varazdin, 2021., str. 175–182.

[74] I. Mozetič, M. Grčar i J. Smailović, „Multilingual Twitter Sentiment Classification: The Role of Human Annotators," *PLOS ONE*, sv. 11, br. 5, str. 1–26, svibanj 2016. DOI: 10.1371/journal.pone.0155036. adresa: https://doi.org/10.1371/journal.pone.0155036.

Popis slika

1. Podjela načina učenja predtreniranih modela (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021) 3
2. Usporedba RNN i LSTM arhitektura (Vlastita izrada prema: Greff, Srivastava, Koutník i dr., 2016) 5
3. Enkoder - dekoder arhitektura (Izvor: Cho, Van Merriënboer, Gulcehre i dr., 2014) 6
4. Enkoder - dekoder primjer (Vlastita izrada prema: Cho, Van Merriënboer, Gulcehre i dr., 2014) 7
5. Konteksutalna ovisnost prilikom korištenja mehanizma pozornosti (Izvor: Bahdanau, Cho i Bengio, 2014) 8
6. Seq2Seq primjer s mehanizmom pozornosti (Vlastita izrada prema: Cho, Van Merriënboer, Gulcehre i dr., 2014) 9
7. Arhitektura modela Transformera (Vlastita izrada prema: Vaswani, Shazeer, Parmar i dr., 2017) 11
8. Prikaz mehanizma pozornosti skaliranog skalarnog produkta i višeglave pozornosti (Vlastita izrada prema: Vaswani, Shazeer, Parmar i dr., 2017) 12
9. Prikaz mehanizma samopozornosti (izvor: Han, Zhang, Ding i dr., 2021) 15
10. Usporedba arhitekture GPT-a i BERT-a (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021) 16
11. Razlika samopozornosti GPT i BERT arhitekture (izvor: Han, Zhang, Ding i dr., 2021) 17
12. BERT predtreniranje i fino prilagođavanje (izvor: Devlin, Chang, Lee i dr., 2018) . 20
13. Podjela modela i njihovih prethodnika (Vlastita izrada prema: Han, Zhang, Ding i dr., 2021) 23
14. Vrijednosti funkcije gubitka prilikom treniranja (Vlastita izrada) 35

Popis tablica

1. Hiperparametri arhitekture za 4 različite veličine GPT-2 modela. 18
2. Hiperparametri arhitekture za 8 različitih veličina GPT-3 modela. 19
3. Prosječni microF1 rezultati za morfosintaktičku anotaciju 21
4. Prosječni microF1 rezultati za zadatak prepoznavanja imenovanih entiteta 22
5. Zapis jedne web stranice u skupu podataka 26
6. Zbunjenost modela na novim skupovima podataka bez prilagođavanja (Osim trening vrijednosti koja je dodana radi usporedbe) 37
7. Google-ova skala za interpretaciju BLEU rezultata 39
8. BLEU bodovi za primjere rečenica 40
9. Uspoređeni modeli na zadatku analize sentimenta. 40
10. F1 rezultati za uspoređene modele po kombinacijama hiperparametara 42

1. Prilog 1: Tablični prikaz vrijednosti funkcije gubitka po skupovima podataka i epohama

U prikazanoj tablici prvi se stupac odnosi na podskup podataka, a ostalih pet na vrijednost funkcije gubitka po epohama. Nazivi skupa podataka su definirani podjelama spomenutima u radu. Tako se prvi broj df(1-7)x_x odnosi na podjelu podataka prilikom pretprocesuiranja, drugi broj dfx(1-3)_x se odnosi na podjelu prilikom tokenizacije, a treći broj dfxx_(1-5) na podjelu zbog *Protocol Buffera* tijekom treniranja. Vrijednosti *Nan* označavaju da u toj epohi nije ostvareno poboljšanje u odnosu na prethodnu, odnosno da se to stanje nije spremilo nego je treniranje sa sljedećim skupom podataka nastavljeno od stanja prve prethodne epohe koja je ostvarila poboljšanje.

Skup/Epoha	1	2	3	4	5
df11_1	5.8946	5.4109	5.1857	5.0449	4.9746
df11_2	4.7800	4.7166	4.7023	Nan	Nan
df11_3	4.5914	4.5647	4.5500	4.5488	Nan
df11_4	4.5028	4.8916	4.4828	4.4789	4.4766
df11_5	4.7294	4.6756	4.6572	4.6898	4.7356
df12_1	4.6626	4.6178	Nan	Nan	Nan
df12_2	4.4674	4.4418	Nan	Nan	Nan
df12_3	4.3712	4.3572	4.3522	Nan	Nan
df12_4	4.3157	4.3078	4.3029	4.3010	4.3001
df12_5	4.2848	4.2804	4.2785	4.2768	4.2758
df13_1	4.5156	4.5020	Nan	Nan	Nan
df13_2	4.3793	4.3675	Nan	Nan	Nan
df13_3	4.2980	4.2859	4.2855	Nan	Nan
df13_4	4.2527	4.2457	4.2414	4.2414	Nan
df13_5	4.2319	4.2277	4.2254	4.2245	4.2235
df21_1	4.4453	4.4421	Nan	Nan	Nan
df21_2	4.3271	4.3152	Nan	Nan	Nan
df21_3	4.2513	4.2428	4.2392	Nan	Nan
df21_4	4.2135	4.2070	4.2038	4.2026	Nan
df21_5	4.1934	4.1897	4.1882	4.1867	4.1860
df22_1	4.3998	4.3977	Nan	Nan	Nan
df22_2	4.2944	4.2862	Nan	Nan	Nan
df22_3	4.2276	4.2185	Nan	Nan	Nan
df22_4	4.1954	4.1886	4.1870	4.1865	Nan
df22_5	4.1665	4.1635	4.1617	4.1606	4.1597

Skup/Epoha	1	2	3	4	5
df23_1	4.3563	Nan	Nan	Nan	Nan
df23_2	4.2740	4.2646	Nan	Nan	Nan
df23_3	4.2006	4.1910	Nan	Nan	Nan
df23_4	4.1671	4.1601	4.1593	4.1589	Nan
df23_5	4.1431	4.1399	4.1374	4.1380	4.1368
df31_1	4.3316	Nan	Nan	Nan	Nan
df31_2	4.2511	4.2414	Nan	Nan	Nan
df31_3	4.1830	4.1734	Nan	Nan	Nan
df31_4	4.1530	4.1462	4.1453	4.1450	Nan
df31_5	4.1237	4.1210	4.1182	4.1178	Nan
df32_1	4.3042	Nan	Nan	Nan	Nan
df32_2	4.2247	4.2187	Nan	Nan	Nan
df32_3	4.1655	4.1575	Nan	Nan	Nan
df32_4	4.1216	4.1156	4.1148	Nan	Nan
df32_5	4.1102	4.1072	4.1063	4.1048	4.1044
df33_1	4.2782	Nan	Nan	Nan	Nan
df33_2	4.1987	4.1941	Nan	Nan	Nan
df33_3	4.1439	4.1350	Nan	Nan	Nan
df33_4	4.1038	4.0978	4.0975	Nan	Nan
df33_5	4.0869	4.0844	4.0829	4.0822	4.0815
df41_1	4.3498	Nan	Nan	Nan	Nan
df41_2	4.2743	4.2652	Nan	Nan	Nan
df41_3	4.2160	4.2041	Nan	Nan	Nan
df41_4	4.1823	4.1747	4.1730	4.1722	Nan
df41_5	4.1600	4.1573	4.1555	4.1542	Nan
df42_1	4.3189	Nan	Nan	Nan	Nan
df42_2	4.2455	4.2366	Nan	Nan	Nan
df42_3	4.1849	4.1758	Nan	Nan	Nan
df42_4	4.1439	4.1369	4.1357	Nan	Nan
df42_5	4.1288	4.1252	4.1237	4.1233	4.1226
df43_1	4.4422	Nan	Nan	Nan	Nan
df43_2	4.3695	4.3637	Nan	Nan	Nan
df43_3	4.3054	4.2968	Nan	Nan	Nan
df43_4	4.2734	4.2685	4.2666	Nan	Nan
df43_5	4.4177	Nan	Nan	Nan	Nan
df51_1	4.5668	Nan	Nan	Nan	Nan
df51_2	4.4720	4.4617	Nan	Nan	Nan
df51_3	4.4097	4.3997	Nan	Nan	Nan
df51_4	4.5211	Nan	Nan	Nan	Nan
df51_5	4.4468	4.4423	Nan	Nan	Nan

Skup/Epoha	1	2	3	4	5
df52_1	4.5432	Nan	Nan	Nan	Nan
df52_2	4.4672	4.4622	Nan	Nan	Nan
df52_3	4.5130	4.4015	Nan	Nan	Nan
df52_4	4.3813	4.3755	4.3754	Nan	Nan
df52_5	4.3553	4.3525	4.3514	4.3505	Nan
df53_1	4.5136	Nan	Nan	Nan	Nan
df53_2	4.4447	4.4419	Nan	Nan	Nan
df53_3	4.3945	4.3893	Nan	Nan	Nan
df53_4	4.3651	4.3595	Nan	Nan	Nan
df53_5	4.3429	4.3404	4.3397	4.3385	Nan
df61_1	4.4714	Nan	Nan	Nan	Nan
df61_2	4.3933	4.3904	Nan	Nan	Nan
df61_3	4.3397	4.3330	Nan	Nan	Nan
df61_4	4.3088	4.3046	4.3043	Nan	Nan
df61_5	4.4446	Nan	Nan	Nan	Nan
df62_1	4.2322	4.2196	Nan	Nan	Nan
df62_2	4.1274	4.1152	Nan	Nan	Nan
df62_3	4.0701	4.0619	4.0581	Nan	Nan
df62_4	4.0379	4.0330	4.0306	4.0293	4.0283
df62_5	4.0139	4.0112	4.0099	4.0094	4.0091
df63_1	4.2286	Nan	Nan	Nan	Nan
df63_2	4.1552	4.1467	Nan	Nan	Nan
df63_3	4.1006	4.0937	Nan	Nan	Nan
df63_4	4.0689	4.0647	4.0633	Nan	Nan
df63_5	4.0527	4.0498	4.0487	4.0476	4.0468
df71_1	4.1739	Nan	Nan	Nan	Nan
df71_2	4.1007	4.0908	Nan	Nan	Nan
df71_3	4.0523	4.0448	4.0430	Nan	Nan
df71_4	4.0184	4.0136	4.0119	4.0112	Nan
df71_5	3.9978	3.9952	3.9939	3.9922	3.9918
df72_1	4.2114	Nan	Nan	Nan	Nan
df72_2	4.1479	4.1431	Nan	Nan	Nan
df72_3	4.1056	4.0987	Nan	Nan	Nan
df72_4	4.0761	4.0709	4.0704	Nan	Nan
df72_5	4.0580	4.0559	4.0548	4.0540	Nan
df73_1	4.2837	Nan	Nan	Nan	Nan
df73_2	4.2221	4.2149	Nan	Nan	Nan
df73_3	4.1747	4.1669	Nan	Nan	Nan
df73_4	4.1436	4.1399	4.1385	Nan	Nan
df73_5	4.1252	4.1222	4.1222	4.1215	4.1209