Legal Q&A Bot

Boris Bibić E2 26/2019
Računarstvo i automatika
Fakultet tehničkih nauka
Univerzitet u Novom Sadu
borisbibic1996@gmail.com

Apstrakt

Rad autora je nadogradnja sistema za pružanje odgovora na pravna pitanja. Pravni dokumenti su opisani sa ontologijom i sačuvani u RDF formatu, a za dobijanje odgovora potrebno je izvršavanje SPARQL upita nad tom ontologijom. Pošto pisanje SPARQL upita nije jednostavno niti intuitivno, a u cilju jednostavnijeg postavljanja pitanja osmišljen je sistem koji se zasniva na dubokim neuronskim mrežama za transformisanje prirodnog jezika u SPARQL upite. U radu je opisan transformer model korišćen za pretvaranje pitanja u prirodnom jeziku na predefinisane SPARQL upite. Sistem je limitiran na pitanja za koja već postoje napisani SPARQL upiti.

Ključne reči—transformer model; upiti u prirodnom jeziku; SPARQL; duboka neuronska mreža.

1. UVOD

Sa porastom dostupnosti informacija na internetu, stvara se potreba za razvijanjem efikasnih sistema za pretraživanje informacija, kao i sistema za odgovaranje na pitanja (engl. *Question and Answering – Q&A*). Danas se veliki broj istraživača bavi ovim pitanjem i razvoj *Q&A* sistema je u usponu. Međutim, još uvek nije dostignuta željena preciznost ovih sistema nad širokim skupom domena znanja. Jedna od metoda kojom se pokušava prevazići problem je formiranje ontologija i semantičkog veba [1].

SPARQL [2] je standardni jezik za pretraživanje ontologija koji zahteva znanje o entitetima u domenu koji se ispituje, kao i razumevanje sintakse i semantike jezika. Zbog toga ga koriste isključivo stručnjaci za semantički veb. U cilju širenja pristupačnosti široj publici, poslednjih decenija istraženi su načini za automatsko prevođenje pitanja sa prirodnog jezika (engl. *Natural Language - NL*) na *SPARQL* upite. Najčešći vid mašinskog prevođenja *NL* jeste uz pomoć neuronskih mreža.

U ovom radu je kreiran sistem zasnovan na transformer neuronskoj mreži koji prevodi pravna pitanja postavljena u *NL* na *SPARQL* upite. Istrenirane su dve verzije modela – sa i bez *finetunovanja*. Za meru evaluacije korišćeni su top-5 i top-1 mera. Najbolja top-5 tačnost iznosi 82,93% i dobijena je *finetunovanjem*, dok je najbolja top-1 tačnost dobijena treniranjem modela sa pravnim datasetom i iznosi 62,8%. Oba modela su trenirana sa *batch*-evima od 64 primera. Za ove modele je korišćena tokenizacija na nivou reči, dok je najbolji model sa tokenizacijom na nivou karaktera imao top-5 tačnost od 33,54% i top-1 tačnost od 6,1%.

U poglavlju dva su predstavljena slična rešenja koja su se bavila ispitivanjem mašinskog prevođenja prirodnog jezika u *SPARQL* ili *SQL* jezik. Poglavlje tri se bavi teorijom enkoder-dekoder mreža, mehanizma pažnje, *multi-head attention* mehanizma i transformer modela. Četvrto poglavlje predstavlja implementaciju sistema. U petom poglavlju su predstavljeni rezultati evaluacije treniranih modela. Poglavlje šest predstavlja sumarizaciju čitavog rada i diskusiju o mogućnostima unapređenja.

2. PRETHODNA REŠENJA

Dva pristupa su bila razmatrana od strane autora – prirodni jezik u *SQL*, a potom u *SPARQL* i direktno iz prirodnog jezika u *SPARQL*. Prvi pristup koristi posrednički jezik, u ovom slučaju *SQL* da bi premostio razliku. Pošto ovaj pristup zahteva treniranje dve odvojene neuronske mreže, a za drugi pristup je pronađen veliki skup podataka za treniranje neuronske mreže, odabran je drugi pristup. Mnogo radova se bavi problemom prevođenja prirodnog jezika u *SPARQL* upit. Isti su se često primenjivali za kreiranje *Q&A* botova.

Autori rada „*A Multiple Ontologies Based Systemfor Answering Natural Language Questions*“ [3] predstavili su sistem odgovora na pitanja koji kombinuje više baza znanja sa parserom prirodnog jezika. Da bi transformisao pitanja postavljena prirodnim jezikom u *SPARQL* ili neki drugi upitni jezik, autori rada su koristili *Quepy* okvir [4]. Transformacija je rađena u dva koraka, korišćenjem regularnih izraza za transformaciju prirodnog jezika u meta-objekte, a zatim korišćenjem *Quepy* okvira za transformaciju meta-objekata u *SPARQL* upit. Za procenu preciznosti i relevantnosti modela korišćene su mere evaluacije preciznost (engl. *precision*) i odziv (engl. *recall*). Rezultati su dobijeni subjektivnim testiranjem 10 ispitanika.

Autori *Yin, Gromann* i *Rudolph* su u svojoj studiji [5] ispitali 8 različitih modela za mašinsko prevođenje prirodnog jezika u *SPARLQ* upite upotrebom neuronskih mreža (engl. *Neural Machine Translation - NMT*). Korišćeni modeli su:

NSpM (*RNN* sa dva *LSTM* sloja iz rada [6]), *NSpM+Att1* (dodata je globalna pažnja), *NSpM+Att2* (dodata je lokalna pažnja), *GNMT-4* sa 4 sloja [7], *GNMT-8* sa 8 slojeva [7], *LSTM Luong* [8], *ConvS2S* i *Transformer*. Za meru evaluacije korišćeni su *BLEU* rezultat, *F1* tačnost, kao i perpleksnost (engl. *perplexity - ppl*). Rezultati ovog rada pokazuju da je *ConvS2S* model bio konstantno bolji od ostalih ispitanih modela, dok je *GNMT* pokazao znatno slabije rezultate od očekivanih. Rezultati, takođe, pokazuju dominaciju *CNN* bazirane arhitekture sa *BLEU* rezultatom do 98 i tačnošću do 94%.

I u radu [9] su korišćene neuronske mreže za generisanje *SPARQL* upita iz prirodnog jezika. Pristup autora ovog rada se bazirao na dva koraka. Prvi korak služi za generisanje vektorske predstave rečenice u prirodnom obliku, dok se u drugom koraku ovaj vektor dovodi na ulaz neuronske mreže (*LSTM* sa mehanizmom pažnje). Neuronska mreža enkodira prirodni jezik iz ulaznog vektora i dekodira ga u *SPARQL* upit. Za generisanje vektora korišćen je pristup sličan radu [10], dok je za neuronsku mrežu poslužio primer iz [11]. Autori su odradili eksprimente sa i bez mehanizma pažnje i došli su do zaključka da pažnja povećava tačnost modela.

Gur i saradnici su imali zanimljiv pristup u radu [12], u kojem su se za generisanje *SQL* upita, pored neuronskih mreža, oslanjali i na ljudsku inteligenciju. Osmislili su sistem pod nazivom *DialSQL*, gde se osobama postavljaju pitanja o ispravnosti *SQL* generisanog upita, a zatim se *SQL* upit i odgovori koriste za poboljšanje samog upita. *DialSQL* se bazira na *RNN* enkoder-dekoder arhitekturi sa mehanizmom pažnje i mehanizmom pokazivača. Prva *RNN* enkoduje svaku interakciju sa čovekom u vidu dijaloga, gde se izlaz iz mreže koristi za predikciju kategorije greške, npr. greška u selekciji, projekciji ili agregaciji. Druga *RNN* koristi tu kategoriju da predvidi početak i kraj sekvence u kojoj se nalazi greška. Konačna *SQL* sekvenca se dekoduje iz vektora kategorije i vektora pozicije greške u upitu. Ovaj sistem je pokazao povećanje tačnosti generisanja upita sa 61,3% na 69% korišćenjem u proseku 2,4 validaciona pitanja po upitu.

Xu, Liu i *Song* su u radu [13] predložili *SQLNet* model baziran na "skicama". "Skice" su *SQL* upiti u koje treba da se unesu nazivi kolona i tabela, dok su ključne reči već na predefinisanim mestima u upitu. Ovo znači da njihovo rešenje nije *sequence-to-sequence* nego *sequence-to-set*, gde set predstavlja listu naziva kolona i tabela koje će biti umetnute u "skicu". Njihovo rešenje ima tačnost od 68%, ali je ograničeno na upite koji odgovaraju predefinisanoj skici koju su koristili.

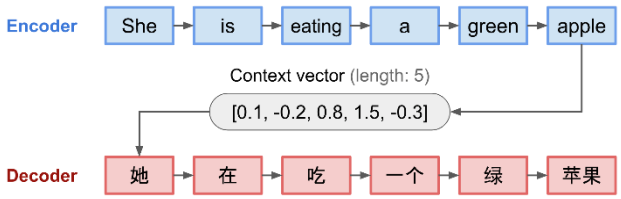
Autori rada [14] su predložili *SyntaxSQLNet*, sintaksno stablo za rešavanje kompleksnih i međudomenskih transformacija prirodnog jezika u *SQL*. *SyntaxSQLNet* se sastoji od više modula sa bidirekcionim *LSTM* slojevima i mehanizmima pažnje. Svaki od modula ima zadatak da prediktuje određene *SQL* tokene, kao npr. *AND/OR, HAVING, DESC/ASC/LIMIT,...* Rezultati su pokazali poboljšanje od 7,3% u odnosu na tada najbolji *SQLNet* [13] model.

3. TEORIJA

Sequence-to-Sequence (*seq2seq*) model je model koji ima za cilj prevođenje jedne sekvence u drugu, što pripada grupi zadataka pod nazivom (neuronsko) mašinsko prevođenje [15]. Glavni predstavnici *seq2seq* modela jesu enkoder-dekoder modeli koji su prvi put objavljeni 2014. godine od strane dve nezavisne grupe istraživača u radovima [16] i [17].

3.1 Enkoder-dekoder

Arhitektura *seq2seq* modela se zasniva na dve rekurentne neuronske mreže koje se zovu enkoder i dekoder. Uloga enkodera je da sekvencu na ulazu (recimo tekst na engleskom jeziku) transformiše u kraću vektorsku reprezentaciju ulaza (engl. *context vector*) koja bi sadržala potencijalno bitne informacije. Ovaj vektor se dalje prosleđuje na ulaz dekodera, koji ima suprotnu ulogu od enkodera. Zadatak dekodera je da iz vektora rekonstruiše početnu sekvencu u željenom formatu (npr. u *SPARQL* jeziku). Cilj treniranja je da se obe mreže obuče tako da enkoder dovoljno dobro kompresuje ulazni vektor u *context* vektor, iz koga bi dekoder mogao da kreira izlazni vektor. Kod *seq2seq* modela enkoder i dekoder mogu biti rekurentne neuronske mreže (engl. *Recurrent Neural Networks - RNN*), koje se često baziraju na *LSTM* ili *GRU* slojevima, ali i konvolucione neuronske mreže (engl. *Convolutional neural network - CNN*). Slika 1 prikazuje ulaz, *context* vektor i izlaz iz enkoder-dekoder modela, gde se rečenica "Ona jede zelenu jabuku" sa engleskog jezika prevodi na kineski jezik.



Slika 1. Prikaz toka prevođenja rečenice sa enkoder-dekoder modelom

3.2 Mehanizam pažnje

Mana *RNN*/*CNN* enkoder-dekoder arhitekture je nemogućnost pamćenja dugačkih vektora (rečenica) uz pomoć vektora fiksne dužine [18, 19]. Zbog ovog problema je nastao mehanizam pažnje (engl. *attention mechanism*) koji je objavljen u radu [15]. Umesto da se prosleđuje poslednji izlazni vektor, ideja *attention* mehanizma je da iskoristi skrivena stanja (izlaze iz skrivenih slojeva) za računanje novih izlaza. Izlaz se računa kao suma svih proizvoda izlaza neurona i *attention* težina (kao dodatnog parametra koji se trenira). *Attention* mehanizam može biti *soft* ili *hard attention* [20], globalni ili lokalni *attention* [20] i *self-attention* [11]. *Soft attention* se računa na osnovu svih skrivenih stanja, dok se kod *hard attention*-a uzima samo deo skrivenih stanja za računanje *attention score*-a, što je manje računarski zahtevno, ali zato nije diferencijabilno. *Global attention*, kao i *soft attention*, koristi sve inpute za računanje *attention*-a, dok lokal *attention* predstavlja kombinaciju *hard* i *soft attention*-a (komputaciono jednostavniji od globalnog i *soft attention* mehanizma, ali je diferencijabilan). Samopažnja (engl. *self-attention*) je jedan od mehanizama koji povezuje različite položaje jedne sekvence da bi izračunao reprezentaciju iste. Ukoliko se posmatra rečenica kao jedna sekvenca, onda se *self-attention* računa za svaku reč na osnovu prethodnih reči u toj rečenici, a ne na osnovu prethodnih stanja mreže. Ovo omogućava paralelizaciju računanja *attention*-a, na koju se oslanja *Multi-head attention*. U tabeli 1 se nalaze neki od *attention* mehanizama sa odgovarajućim funkcijama za računanje *attention score*-a:

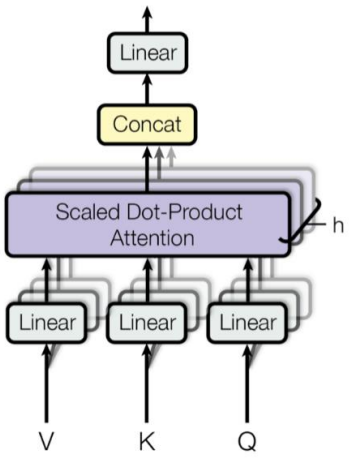
Engleski naziv	*Alignment score* funkcija
Content-base attention [21]	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$
Additive [15] /*Concat* [8] / *Additive attention* [22]	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$
Location-Base [8]	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$
General [8]	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$
Dot-Product [8]	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$
Scaled Dot-Product [22]	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$

Tabela 1. Prikaz nekih od funkcija za računanje *attention score*-a preuzeto sa [23]

U tabeli 1 $\boldsymbol{s}_t$ predstavlja skriveno stanje dekodera za izlaznu reč na poziciji t gde t=1,...,m (dužina izlazne sekvence), $\boldsymbol{h}_i$ je konkatenacija *foward* i *backward* skrivenih stanja bidirekcionog enkodera gde i ide od 1 do dužine ulazne sekvence, $\mathbf{v}_a$ i $\mathbf{W}_a$ su matrice težina *aligement* funkcija koje se treniraju, n je dužina ulazne sekvence.

3.3 Multi-head attention

Transformer model se zasniva na *multi-head scaled dot-product attention* mehanizmu, tj. *multi-head self-attention* (*MHA*) mehanizmu. Pošto *self-attention* mehanizam ne zahteva rekurziju ili konvoluciju poseduje mogućnost paralelizacije, koja je u transformeru iskorišćena upotrebom "glava" (engl. *head*). Svaka "glava" ima zadatak da izračuna pažnju po principu *scaled dot-product attention*-a, a na kraju se rezultati svih "glava" konkateniraju u konačan *attention score* vektor (slika 2).



Slika 2. Prikaz *MHA*

Cilj *MHA* je paralelno prikupljanje informacija iz više uglova, što se pokazalo vrlo uspešno kod transformer modela. Transformer posmatra ulazni vektor kao par ključ-vrednost (*K*,*V*) oba dužine ulazne sekvence. Ključ (engl. *key*) i vrednost (engl. *value*) predstavljaju skrivena stanja enkodera. Kod dekodera, prethodni izlaz je predstavljen vektorom *Q* (engl. *query*) dužine izlazne sekvence, a izlaz iz trenutne "glave" dekodera se računa preko *scaled dot-product* formule:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

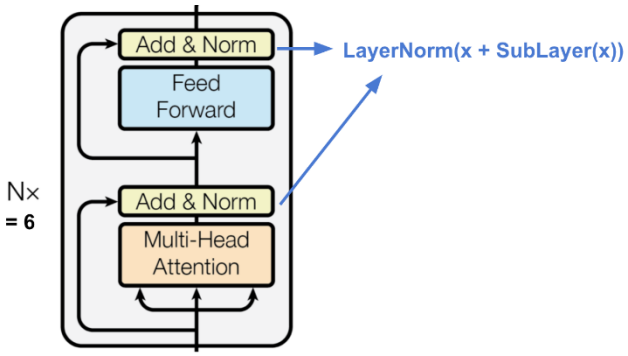
Pošto se transformer model zasniva na *MHA*, onda je potrebno uvesti trenirajuće matrice $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ i $\mathbf{W}_i^V$ i $\mathbf{W}^O$ koje će uticati na *attention* mehanizam tako da svaka glava ima drugačiji pogled na ulaznu sekvencu. Uvođenjem ovih matrica, konačna formula izračunavanja *attention*-a kod *MHA* sa h "glava" izgleda:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O$$

$$\text{gde je head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

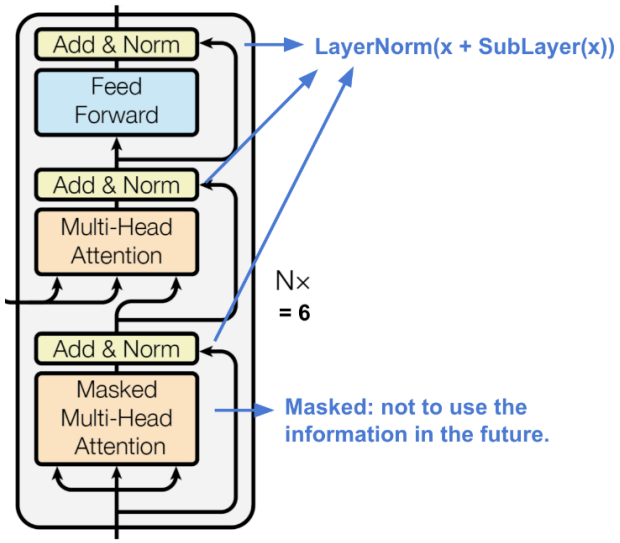
3.4 Transformer model

Transformer model je predložen 2017. godine u radu [22] i uneo je velike promene na polju mašinskog prevođenja. Ovaj model je pokazao da je moguće kreirati *sequence-to-sequence* mrežu samo uz pomoć *self-attention* mehanizma bez rekurentnih slojeva. Transformer se sastoji od enkoder i dekoder mreže. Enkoder je mreža od N slojeva, gde svaki sloj sadrži *MHA* sloj i *fully connected feed-foward* mrežu (*FFNN*), zajedno sa *dropout* slojem i slojem za normalizaciju (slika 3).



Slika 3. Prikaz enkodera sa slojevima

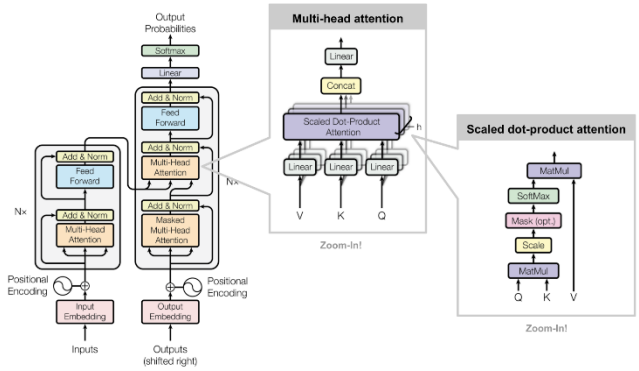
Dekoder sa druge strane sadrži N slojeva, a svaki od slojeva ima 2 subsloja (*MHA* i *FFNN* slojevi), zajedno sa *dropout* slojem i slojem za normalizaciju (slika 4). Prvi *MHA* subsloj je izmenjen tako da se za računanje attention-a željene pozicije mogu gledati samo prethodne podsekvence, sprečavajući gledanje u buduće sekvence.



Slika 4. Prikaz dekodera sa slojevima

Celokupna arhitektura transformera je prikazana na slici 5, gde se primećuje da se ulazna i željena izlazna sekvenca

prvo propuštaju kroz *embedding* sloj (da bi dimenzije bile uniformne), a na poslednji izlaz iz dekodera su dodate *softmax* i linearni sloj (u implementaciji je to *dense* sloj).



Slika 5. Prikaz arhitekture transformer modela

4. IMPLEMENTACIJA

Glavni deo sistema čini transformer model[1] sastavljen od dve neuronske mreže - enkodera i dekodera. Ulaz u sistem je rečenica u prirodnom jeziku, dok je izlaz iz sistema odgovarajući *SPARQL* upit. Ovaj *SPARQL* upit se dalje koristi za dobavljanje odgovora na osnovu kreirane ontologije iz drugog rada[2]. Sistem ima mogućnost da u toku rada vrati i više *SPARQL* upita za koje je sistem odredio da su najverovatniji prevodi ulaznog *NL* pitanja. Transformer model ima zadatak da prevede *NL* pitanje u *SPARQL* upit, ali zbog specifičnosti zadatka očekivano je da će transformer vratiti *SPARQL* upit koji nije uvek semantički ispravan. Da bi se taj problem prevazišao, rešeno je da se upiti dobijeni transformer modelom porede sa postojećim (semantički ispravnim) *SPARQL* upitima u bazi i vrati se najsličniji *SPARQL* upit/upiti iz baze. Izračunavanje sličnosti dva *SPARQL* upita je rađeno sa "*gensim.similarities.Similarity*" klasom koja izračunava kosinusnu sličnost.

Ulazna rečenica se prvo tokenizuje uz pomoć „*tf.keras.preprocessing.text.Tokenizer*“ klase u vektorsku reprezentaciju. Pokušana je tokenizacija na nivou karaktera, kao i tokenizacija na nivou reči, ali je zbog loših rezultata prva odbačena i implementacija je bazirana na "*word-based*" tokenizaciji. Dobijeni vektor se propušta kroz transformer model koji vraća izlaz u vektorskoj reprezentaciji. Dobijeni vektor se pretvara u *SPARQL* upit uz pomoć rečnika iz druge *Tokenizer* klase (bazirane na rečima iz *SPARQL* upita). Dve *Tokenizer* klase predstavljaju rečnike za engleske i *SPARQL* reči, gde se prvo upotrebljava engleski rečnik, a na kraju *SPARQL* rečnik.

Izgled slojeva i broj parametara enkodera i dekodera se nalazi u tabelama 2 i 3. Za aktivacionu funkciju neurona izabrana je *ReLU* zbog njenih pozitivnih karakteristika (brzine obučavanja, lakoće izračunavanja i bez problema sa "*vanishing gradient*"). Verovatnoća odbacivanja neurona kod *dropout* sloja je postavljena na 0,1. Dimenzija *dense embedding*-a je 128, što je ujedno i veličina "glave" *MHA* sloja. Broj paralelnih "glava" u *MHA* sloju je 8. Broj uzastopnih *feed-foward* slojeva i *MHA* slojeva je 4.

Slojevi	Tip sloja	Broj param.
embedding	Embedding	15721344
dropout	Dropout	0
multi_head_attention	MultiHeadAttention	66048
multi_head_attention_1	MultiHeadAttention	66048
multi_head_attention_2	MultiHeadAttention	66048
multi_head_attention_3	MultiHeadAttention	66048
dropout_1	Dropout	0
dropout_2	Dropout	0
dropout_3	Dropout	0
dropout_4	Dropout	0
layer_normalization	LayerNormalization	256
layer_normalization_1	LayerNormalization	256
layer_normalization_2	LayerNormalization	256
layer_normalization_3	LayerNormalization	256
dense_16	Dense	66048
dense_17	Dense	66048
dense_18	Dense	66048
dense_19	Dense	66048
dense_20	Dense	65664
dense_21	Dense	65664
dense_22	Dense	65664
dense_23	Dense	65664
dropout_5	Dropout	0
dropout_6	Dropout	0
dropout_7	Dropout	0
dropout_8	Dropout	0
layer_normalization_4	LayerNormalization	256
layer_normalization_5	LayerNormalization	256
layer_normalization_6	LayerNormalization	256
layer_normalization_7	LayerNormalization	256
Ukupan broj parametara: 16,514,432		

Tabela 2. Prikaz modela enkodera

Slojevi koji čine ovaj model su opisani ispod:

- *Embedding* [24] - Pretvara pozitivne cele brojeve u guste vektore fiksne veličine. *Embedding* sloj pomaže smanjenjem dimenzionalnosti vektora, najčešće vektora sa puno nula (retki vektori). Korišćen je kao prvi (ulazni) sloj kod enkodera i dekodera.
- *Dropout* [25] - ovaj sloj služi za ignorisanje određenog broja neurona prilikom treniranja, tj. prilikom prolaza unapred ili unazad. Potpuno povezani slojevi mogu dovesti do zavisnosti između neurona koja dovodi do overfitovanja. *Dropout* sloj ignoriše nasumično odabrane neurone prilikom svakog prolaska u trening fazi, dok se kod testiranja koriste svi neuroni i međusobne veze.
- *MultiHeadAttention* [26] - računa uticaj svake reči u rečenici (*attention score*). Ovo omogućava modelu da se fokusira na informacije iz različitih delova rečenice (vektorskog prostora).
- *LayerNormalization* [27] - normalizuje ulazne vrednosti za svaki neuron u sloju. Normalizacija poboljšava brzinu treniranja mreže.
- *Dense* [28] - potpuno povezani sloj neuronske mreže koji za cilj ima promenu dimenzionalnosti vektora.

[1] Kôd modela je preuzet sa *https://github.com/ChunML/NLP/blob/master/machine_translation/train_transformer_tf2.py* i prilagođen potrebama projekta

[2] Rad je dostupan na: *https://github.com/Sale1996/Legal-QA-Bot*

Slojevi	Tip sloja	Broj param.
embedding_1	Embedding	15811200
dropout_9	Dropout	0
multi_head_attention_4	MultiHeadAttention	66048
multi_head_attention_5	MultiHeadAttention	66048
multi_head_attention_6	MultiHeadAttention	66048
multi_head_attention_7	MultiHeadAttention	66048
dropout_10	Dropout	0
dropout_11	Dropout	0
dropout_12	Dropout	0
dropout_13	Dropout	0
layer_normalization_8	LayerNormalization	256
layer_normalization_9	LayerNormalization	256
layer_normalization_10	LayerNormalization	256
layer_normalization_11	LayerNormalization	256
multi_head_attention_8	MultiHeadAttention	66048
multi_head_attention_9	MultiHeadAttention	66048
multi_head_attention_10	MultiHeadAttention	66048
multi_head_attention_11	MultiHeadAttention	66048
dropout_14	Dropout	0
dropout_15	Dropout	0
dropout_16	Dropout	0
dropout_17	Dropout	0
layer_normalization_12	LayerNormalization	256
layer_normalization_13	LayerNormalization	256
layer_normalization_14	LayerNormalization	256
layer_normalization_15	LayerNormalization	256
dense_56	Dense	66048
dense_57	Dense	66048
dense_58	Dense	66048
dense_59	Dense	66048
dense_60	Dense	65664
dense_61	Dense	65664
dense_62	Dense	65664
dense_63	Dense	65664
dropout_18	Dropout	0
dropout_19	Dropout	0
dropout_20	Dropout	0
dropout_21	Dropout	0
layer_normalization_16	LayerNormalization	256
layer_normalization_17	LayerNormalization	256
layer_normalization_18	LayerNormalization	256
layer_normalization_19	LayerNormalization	256
dense_64	Dense	15934725
Ukupan broj parametara: 32,804,229		

Tabela 3. Prikaz modela dekodera

4.1 Datasetovi

Testirane su 4 varijacije transformer modela u zavisnosti od veličine paketa i korišćenih datasetova. Prvo je model treniran nad *DBNQA* datasetom [29] koji sadrži 894499 parova engleska pitanja-*SPARQL* upiti i poseduje oko 131000 engleskih reči i 244900 *SPARQL* tokena. Zatim je istrenirani model *finetunovan* sa ručno kreiranim datasetom pravnih pitanja. Ovaj pravni dataset je kreiran od strane autora i sadrži 1625 pitanja na engleskom jeziku, koja se preslikavaju na 38 *SPARQL* pitanja. Osim *finetunovanja*, ovaj dataset je upotrebljen da se istrenira drugi transformer model, bez upotrebe *DBNQA* dataseta. Za potrebe drugog dela sistema kreiran je dataset sa 38 *SPARQL* pitanja sadržanih u ručno kreiranom pravnom datasetu. Rečenice iz ovog dataseta su upoređivane sa rečenicom dobijenom sa transformer modelom. Dataset je podeljen na trening, test i validacioni skup u odnosu 80:10:10.

4.2 Treniranje

Obe verzije transformer modela su trenirane na dva načina - u paketima (engl. *batch*) od po 64 sekvence i pojedinačne sekvence (*batch* = 1). Korišćenjem paketa koji je veći od 1 omogućava se veća otpornost na šum (engl. *outliers*) pri treniranju, tako što se ceo *batch* propušta kroz mrežu pre nego što se ažuriraju težine. Za ažuriranje težina neuronskih mreža odabran je optimizator *Adam* zbog dobrih performansi kod dubokih neuronskih mreža. Brzina učenja (engl. *learning rate - lr*) je sprovedena po principu "*warmup then decay*" sa "*warmup*" periodom od 4000 koraka (slika 6).

Slika 6. Prikaz brzine učenja *finetunovanog* modela

Postepenim "zagrevanjem" daje se vreme *Adam* optimizatoru da podesi parametre gradijenta, a nakon 4000 koraka započinje optimizacija modela ka konvergenciji. Pored brzine učenja, drugi parametri *Adam* optimizatora su: $\beta 1$=0,9, $\beta 2$=0,98 i ε=1e-9. Svaki model je treniran najmanje 28 epoha, a najviše 220 epoha. U toku treniranja korišćen je "*TensorBoard*" [30] za praćenje rezultata treniranja i evaluacije modela.

4.3 Evaluacija modela

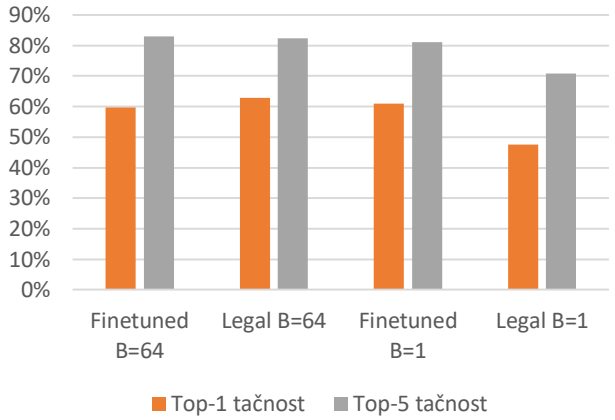
Pošto je potrebno da sistem vrati kao rezultat neku od predefinisanih rečenica koje su najverovatniji prevod ulaznog pitanja, *BLEU* rezultat nije relevantna mera evaluacije. *BLEU* rezultat govori koliki je kvalitet teksta dobijenog mašinskim prevođenjem, što u ovom slučaju nije bitno jer se na kraju uvek vraćaju predefinisani sintaksički ispravni *SPARQL* upiti. Iz tog razloga je korišćena top-n (tačnije top-1 i top-5) mera evaluacije. One pokazuju kolika je tačnost celokupnog sistema, a ne samo transformer modela. Top-n metrika govori koliko često prediktovana klasa upada u gornjih *N* vrednosti *softmax* distribucije, što se koristi kod sistema za rekomendaciju (npr. preporuka oglasa, *Netfliks* filmova, *Amazon* robe,...). U ovom radu sistem preporučuje top 5 odgovora na postavljeno pitanje i time povećava šansu da će osoba koja postavlja pitanje dobiti adekvatan odgovor. Top-1 metrika ili tačnost (engl. *accuracy*) je mera koja pokazuje koliko je model puta dao očekivan prevod, dok top-5 pokazuje koliko je puta bilo koji od 5 ponuđenih odgovora sistema odgovarao traženom prevodu. Iz ovog sledi da top-1 mera ima strožije kriterijume, a samim tim i manji procenat pogađanja (u idealnom slučaju će imati isti rezultat kao i top-5 mera). Validacija modela sa top-n metrikom je rađena nad validacionim skupom nekoliko puta po epohi.

5. Rezultati

Za *finetunovanje* je korišćen početni model treniran sa *DBNQA* datasetom koji je postigao najbolje rezultate. Ovaj model ima validacionu tačnost (engl. *validation accuracy*) od 67,01% i grešku treniranja (engl. *training loss*) 0,0978. Ova tačnost je postignuta na kraju druge epohe.

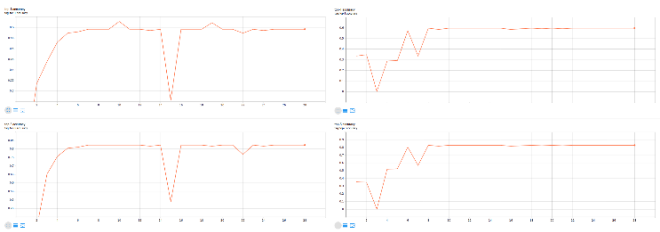
Poređenjem rezultata *finetunovanog* modela i modela koji je treniran direktno nad pravnim datasetom (slika 7) ističu se dve činjenice:

- treniranje sa *batch*-evima pozitivno utiče na rezultate, gde su se bolje pokazala dva modela sa *batch*-evima od 64 sekvence, nego modeli sa *batch*=1;
- treniranje nad većim skupom, a zatim *finetunovanje* nad problemski specifičnim primerima daje malo bolje rezultate od direktnog treniranja modela sa manjim pravnim skupom podataka.



Slika 7. Presek najboljih top-n rezultata modela

Najlošije se pokazao model treniran nad pravnim skupom sa *batch*=1, što je i očekivano. Najbolja top-5 tačnost iznosi 82,93% i dobijena je *finetunovanjem*, dok je najbolja top-1 tačnost dobijena treniranjem modela sa pravnim datasetom i iznosi 62,8% - oba modela su trenirana sa *batch*-evima od 64 primera. Drugi najbolji rezultati su 82,32% za top-5 (pravni skup, *batch*=64) i 60,98% za top-1 tačnost (*DBNQA* + pravni skup, *batch*=1). Prosečna razlika između top-1 i top-5 tačnosti modela iznosi 21,5%. Test greška je u svim primerima pala na 0 posle dužeg treniranja, što ukazuje na overfitovanje usled malog pravnog skupa podataka. Ovo se može rešiti ubacivanjem više primera u skup podataka, za šta nije bilo moguće u maloj ontologiji koja je bila korišćena za razvijanje *SPARQL* upita. Promene top-n tačnosti modela (treniranih sa *batch*-evima od 64 sekvence) tokom treniranja se mogu videti na slici 8.



Slika 8. Top-n rezultati modela tokom treniranja. Graf gore levo je top-1 tačnost *legal* modela, dole levo je top-5 tačnost *legal* modela, gore desno je top-1 tačnost *finetunovanog* modela, dole desno je top-5 tačnost *finetunovanog* modela.

Gorenavedeni rezultati su vezani za tokenizaciju na nivou reči, dok se tokenizacija na nivou karaktera pokazala znatno inferiornom. Najbolji rezultati ovih modela su pokazali top-5 tačnost od 33,54% i top-1 tačnost od 6,1%.

6. ZAKLJUČAK

Sa porastom dostupnosti informacija sve je veći broj radova koji se bave razvojem *Q&A* sistema iako još nije dostignuta željena preciznost nad širokim skupom domena znanja. U radu autora se koristio dataset parova pravnih pitanja u prirodnom jeziku i *SPARQL* upita.

Cilj rada je bio kreiranje sistema zasnovanog na transformer neuronskoj mreži. Sistem na ulazu prima pitanje na engleskom jeziku, a na izlazu vraća 5 najverovatnijih *SPARQL* upita koji bi izvršavanjem nad ontologijom dali odgovor na postavljeno pitanje. Transformer model se sastoji iz enkoder i dekoder neuronskih mreža, koje prevode ulaznu sekvencu u *SPARQL* upit. Istrenirane su dve verzije modela – sa i bez *finetunovanja*. Takođe, isprobano je treniranje sa *batch*-evima od 64 sekvence i bez *batch*-eva. Za potrebe *finetunovanja*, prvo je istreniran transformer model upotrebom *DBNQA* dataseta. *Finetunovanje* i treniranje modela bez *finetunovanja* je rađeno upotrebom ručno napisanog pravnog dataseta. Ovaj dataset sadrži 1625 parova engleskih pitanja - *SPARQL* upita. Za meru evaluacije korišćeni su top-5 i top-1 mera. Najbolja top-5 tačnost iznosi 82,93% i dobijena je *finetunovanjem*, dok je najbolja top-1 tačnost dobijena treniranjem modela sa pravnim datasetom i iznosi 62,8%. Oba modela su trenirana sa *batch*-evima od 64 primera. Za ove modele je korišćena tokenizacija na nivou reči, dok je najbolji model sa tokenizacijom na nivou karaktera imao top-5 tačnost od 33,54% i top-1 tačnost od 6,1%.

Moguće je da bi se korišćenjem *engrama* (recimo *trigrama*) tačnost sistema još više poboljšala. Takođe, povećanjem relativno malog pravnog skupa, tačnost modela bi se povećala. Za buduća istraživanja, bilo bi interesantno istražiti uticaj hiperparametara modela, kao što su broj slojeva, broj *MHA* „glava“, izbor optimizatora i veća segmentacija *batch*-eva.

REFERENCE

[1] Ramkumar, A.S. and Poorna, B., 2014, March. Ontology based semantic search: an introduction and a survey of current approaches. In 2014 International Conference on Intelligent Computing Applications (pp. 372-376). IEEE.Zou, H. and Hastie, T., 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, *67*(2), pp.301-320.

[2] W3C, „SPARQL Query Language for RDF” (https://www.w3.org/TR/rdf-sparql-query/, pristupano 11.4.2020)

[3] El-Ansari, A., Beni-Hssane, A. and Saadi, M., 2017. A multiple ontologies based system for answering natural language questions. In Europe and MENA Cooperation Advances in Information and Communication Technologies (pp. 177-186). Springer, Cham.

[4] Quepy (https://quepy.readthedocs.io/en/latest/, pristupano 14.6.2020)

[5] Yin, X., Gromann, D. and Rudolph, S., 2019. Neural Machine Translating from Natural Language to SPARQL. arXiv preprint arXiv:1906.09302.

[6] Soru, T., Marx, E., Moussallem, D., Publio, G., Valdestilhas, A., Esteves, D. and Neto, C.B., 2017. SPARQL as a Foreign Language. arXiv preprint arXiv:1708.07624.

[7] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. and Klingner, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

[8] Luong, M.T., Pham, H. and Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

[9] Luz, F.F. and Finger, M., 2018. Semantic parsing natural language into sparql: improving target language representation with neural attention. arXiv preprint arXiv:1803.04329.

[10] Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[11] Cheng, J., Dong, L. and Lapata, M., 2016. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733.

[12] Gur, I., Yavuz, S., Su, Y. and Yan, X., 2018, July. Dialsql: Dialogue based structured query generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1339-1349).

[13] Xu, X., Liu, C. and Song, D., 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. arXiv preprint arXiv:1711.04436.

[14] Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z. and Radev, D., 2018. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. arXiv preprint arXiv:1810.05237.

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

[16] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[17] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[18] Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

[19] Koehn, P. and Knowles, R., 2017. Six challenges for neural machine translation. arXiv preprint arXiv:1706.03872.

[20] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, June. Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057).

[21] Graves, A., Wayne, G. and Danihelka, I., 2014. Neural turing machines. arXiv preprint arXiv:1410.5401.

[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[23] Attention? Attention! (https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html, pristupano 14.6.2020)

[24] TensorFlow dokumentacija za Embedding sloj (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding, pristupano 14.6.2020)

[25] TensorFlow dokumentacija za Dropout sloj (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout, pristupano 14.6.2020)

[26] TensorFlow dokumentacija za MultiHeadAttention sloj (https://www.tensorflow.org/addons/api_docs/python/tfa/layers/MultiHeadAttention, pristupano 14.6.2020)

[27] TensorFlow dokumentacija za LayerNormalization sloj (https://www.tensorflow.org/api_docs/python/tf/keras/layers/LayerNormalization, pristupano 14.6.2020)

[28] TensorFlow dokumentacija za Dense sloj (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense, pristupano 14.6.2020)

[29] DBpedia Neural Question Answering (DBNQA) dataset (https://figshare.com/articles/Question-NSpM_SPARQL_dataset_EN_/6118505, pristupano 14.6.2020)

[30] TensorFlow, "TensorBoard: TensorFlow's visualization toolkit" (https://www.tensorflow.org/tensorboard, pristupano 14.6.