LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

ALTERNATĪVAS NEIRONU MAŠĪNTULKOŠANAS ARHITEKTŪRAS

MAĢISTRA DARBS

Darba autors: **Eduards Mukāns**

Stud. apl. Nr. em18044

Darba vadītājs:

profesors Dr. Sc. Comp. Guntis Bārzdiņš

RĪGA 2020

ANOTĀCIJA

Pētījuma mērķis: Izpētīt alternatīvas mašīntulkošanas arhitektūras, pielietojot jaunas pieejas, lai izveidotu angļu-latviešu tulkotāju un apvienot attēlu ar teikumu matricu vienā modelī.

Nozīmīgākie rezultāti: Darba ietvaros tika uztrēnēti dažādi mašīntulkotāji. Viens balstās uz «transformer» arhitektūru. Šis modeļis uztrenēts pilnīgi no jauna. Otrs mašīntulkotājs balstās uz pretrenētiem modeliem.

Darbā tika salīdzinātas dažādas pieejas un tika izvēlēts labākais pretrenēts modelis XLM-R, uz kura bāzes izveidots tulkotājs. Pirmajā tulkotājā iegūtie tulkojuma rezultāti ir pietiekami precīzi un gramatiski pareizi. Taču šis tulkotājs pielāgots politiski vai juridiski virzītam tekstam, jo uztrenēts izmantojot pārsvarā Eiroparlamenta sesijas datus. Otrajam tulkotājam pietrūka resursu apreķināšanai, lai uztrenētu modeļi ar visiem datiem. Trešais modeļis ieejā pieņem gan XLM-R matricu, gan InceptionV3 matricu. Iegūtie rezultāti nedod viennozīmīgu atbildi par to, kāds modeļis uzbūvē labāku valodas modeļi. Tomēr modeļa rezultāti apstiprināja hipotēzi, ka nav svarīgi no kura avota tiek iegūta matrica, bet būtiski tikai tas, cik precīzi matrica reprezentē savstārpējos attālumus ar citiem teikumiem.

Atslēgas vārdi: BERT, XLM-R, InceptionV3, Transformer, mašīntulkošana.

ABSTRACT

Title: Alternative neural machine translation architectures

Research purpose: To study alternative machine translation architectures, using new approaches to create an English-Latvian translator and combine the image with a sentence embeddings in a single model.

The most significant results: Within the work, various machine translators were trained. One is based on the "transformer" architecture. This model has been trained from scratch. The second machine translator is based on pre-trained models.

Different approaches were compared in the work and the most significant results were obtained using XLM-R model, on the basis of that model the translator was created. The translation results obtained by the first translator are sufficiently accurate and grammatically correct. However, the translator has been adapted to a politically or legally motivated text, as it has been trained mostly using European Parliament sessions data. The second translator lacked the computing resources to train model with all the data. The third model accepts both an XLM-R embedding and an InceptionV3 embedding at the input. The obtained results do not provide an unambiguous answer as to which model performs better in language modeling. However, the results of the model confirmed the hypothesis that it does not matter from which source the embedding is obtained, but what matters, is how accurately the embedding represents the distances between other sentences.

Keywords: BERT, XLM-R, InceptionV3, Transformer, machine translation.

AUTOREFERĀTS

Darbā tika pielietots «Transformer» modeļis, lai uzbūvētu angļu-latviešu valodas tulkotāju. Kā arī tika izpetītas jaunākas pieejas un arhitektūras neironu mašīntulkošānā. Balstoties uz šiem pētījumiem tika izveidots cits tulkotājs, kas balstās uz pretrenētiem modeliem. Izpētot citus rakstus un pētījumus nav atrasti citi tulkotāji, kas balstās uz BERT teikuma vektoriem vai XLM-R vārdu vektoriem. Lai būtu iespējams uztrenēt modeli, tika apkopoti dati no Eiroparlamenta un Eiropas Investīcijas bankas sesijām, MS-COCO anotācijām un citiem resursiem. Tika izveidota programma, kas notīra un atfiltrē teksta datus.

SATURA RĀDĪTĀJS

IEVADS

Parādoties internetam cilvēki sāka vairāk starptautiski komunicēt, kā arī uzņēmēji sāka vairāk orientēties uz globālo tirgu un savas valsts robežās mazāk paļauties uz lokālām iespējām. Cilvēkiem ir nepieciešams vairāk komunicēt dažādās valodās globalizācijas procesa dēļ. Cilvēka smadzenes ir ierobežotas un nevar iemācīties valodu tik ātri kā gribētos, tāpēc tulkotāji kļūst arvien populārāki un pieprasītāki. Profesionālie tulkotāji ir dārgi un nav vienmēr pieejami. Līdz ar to nepieciešams izmantot alternatīvas. Meklēt informāciju vārdnīcās nav ērti un tas aizņem vairāk laika. Tāpēc mūsdienās pārsvarā tiek izmantoti elektroniskie tulkotāji. Tā ir lēta un viegli pieejama alternatīva.

Pašā sākumā elektroniskie tulkotāji bija ļoti primitīvi. Varēja vienkārši atrast konkrēto vārdu vai izplatītu teikumu. Vajadzēja pašam domāt par gramatiku, kā pielietot vārdus un sastādīt teikumus. Valodu noteikumi atšķiras, tāpēc, lai sastādītu pareizu vai saprotamu teikumu bija nepieciešams zināt vismaz valodas pamatus. Piemēram, vācu valodā teikuma priekšmets atrodas teikuma beigās. Modernie elektroniskie tulkotāji dod iespēju pašam ievadīt teikumu, kuru ir nepieciešams pārtulkot savā dzimtā valodā un tulkotājs izvadīs rezultātu nepieciešamā valodā.

Tulkotājus, kas spēj pietiekami kvalitatīvi pārtulkot latviešu valodas tekstu ir divi - Google Translate un Tildes tulkotājs. Google Translate ir universālais tulkotājs, kas nespecializējās uz konkrētu valodu, izņemot angļu valodu, jo tā ir starptautiska valoda. Bet šis tulkotājs ir inovatīvs. Google Brain Team pirmie izgudro un ieviesa jaunas pieejas. Tilde ir tulkotājs, kas specializējas uz latviešu valodu. Par labāku un efektīvāku tulkotāju var diskutēt.

Darba mērķis ir izpētīt alternatīvas mašīntulkošanas arhitektūras, pielietojot jaunas pieejas, lai izveidotu angļu-latviešu tulkotāju un apvienot attēlu ar teikumu matricu vienā modelī.

Darbs ir sadalīts uz divām daļām - teorētisko un praktisko. Teoretiskajā daļā ir īsi aprakstītas mūsdienu pieejas, arhitektūras un metodes. Praktiskajā daļā ir veikti ekperimenti, kuru mērķis ir pielietot jaunas arhitektūras, salīdzināt ar esošiem risinājumiem un mēģināt attīstīt šīs pieejas. Visi ekperimenti ir veikti izmantojot sekojošas tehnoloģijas:

- Google Colab;
- Tensorflow;
- Python.

Praktiskajā daļā ir izvirzīti sekojoši uzdevumi:

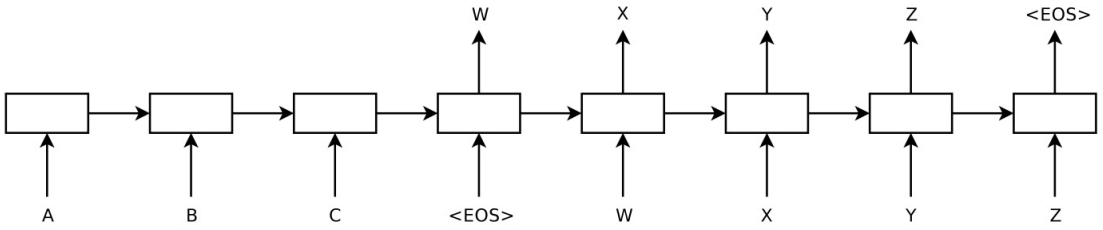
1. Notīrīt tekstus, lai būtu iespējams pielietot tos modeļu trenēšanai;
2. Pielietot «transformer» arhitektūru angļu-latviešu tekstiem un uztrenēt modeļi, kas varēs pārtulkot angļu valodas tekstus uz latviešu valodu;
3. Pielietot pretrenētus modeļus tulkošanas uzdevumam;
4. Mēģināt apvienot pretrenēta modeļa matricu ar InceptionV3 matricu un iegūt pārtulkotu anotāciju.

1. MAŠĪNTULKOŠANAS ARHITEKTŪRAS UN METODES

1.1. Sequence to Sequence

Sequence to sequence (seq2seq, secība-secība) modelis ir dziļas mācīšanās modelis, kas ir guvis daudz panākumu tādos uzdevumos kā mašīntulkošana, teksta apkopošana un attēla parakstīšana. Google Translate sāka izmantot šādu modeli ražošanā 2016. gada beigās.

Seq2Seq modelis ir modelis, kas ņem vārdu vai burtu secību un izvada citu secību. Mašīntulkošanas gadījumā ieeja ir vārdu virkne vienā valodā un izejā ir tulkota vārdu virkne.



1.1. attēls. Sequence to Sequence modelis (Sutskever et al, 2014).

1.1. attēlā ir paradīts secības tulkojuma piemērs. Modelis saņem secību «ABC» un izvada secību «WXYZ». Pielietojot šo modeli nav obligāti kā secības garums sakritīs ar izejas garumu, kas arī ir parādīts 1.1. attēla piemērā. Modeļa galvenie elementi ir enkodētāji un dekodētāji, kas attēlā ir apzīmēti ar baltiem taisnstūriem. Enkodēšana ir datu konvertēšanas process no vienas formas uz citu. Dekodēšana ir enkodēšanas pretējais process. Mašīntulkošanas gadījumā, enkodētājs pārveido teikumu uz matemātisko vektoru un dekodētājs var saprast šo vektoru un mēģināt atrast atbilsošu teikumu citā valodā. Pēc būtības enkodētājs un dekodētājs ir rekurentie neironu tīkli (recurrent neural networks, RNN).

Modelis darbojas sekojoši:

1. Teikums sakotnējā valodā tiek sadalīts uz vārdiem;
2. Katrs vārds atsevišķi tiek ielikts enkodētājā, kur vārds ir pārveidots uz vektoru;
3. Kad cikls sasniedz speciālo simbolu <EOS> (End-Of-Sentence, teikuma nobeigums) enkodētājs aprēķina pilna teikuma vektoru, kas tiek saukts par teikuma konteksta vektoru;
4. Dekodētājs saņem konteksta vektoru un aprēķina varbūtību katram vārdam balstoties uz iepriekšejo kontekstu;
5. Process turpinas kamēr <EOS> simbolam nebūs lielākā varbūtība.

Šajā gadījumā konteksta vektors ir vienkāršs saraksts ar decimāliem skaitļiem. Šī vektora piemērs var būt sekojošs: (0.32, 0.81, -0.64, 0.08). Iepriekš minētā piemēra ietvaros, konteksta vektora lielums ir 4, bet realā dzīvē RNN tīklos tiek izmantoti vektori ar lielumam 256, 512, 1024.

Pēc uzbūves RNN katrā posmā pieņem divas ieejas komponentes: vārds no ievadīta teikuma un slēpto stāvokli. Tomēr vārdu vajag attēlot ar vektoru. Lai vārdu pārveidotu vektoru formā, tiek izmantoti "word embedding" algoritmi. Šie algoritmi pārvērš vārdus vektoru formā, kas spēj uztvert daudz vārdu nozīmes vai semantisko informāciju. Piemēram, šādiem vārdiem ir raksturīgas sekojošas matemātiskas darbības: karalis - vīrietis + sieviete = karaliene.

1.2. Semi-supervised Sequence Learning

«Semi-supervised Sequence Learning» ir jauna pieeja, kas tiek izmantota modeļu trenēšanas laikā. Šī pieeja ir izstradāta Google un ir aprakstīta darbā Andrew Dai, Quoc V.Le, «Semi-supervised Sequence Learning». Darbā ir piedavātas 2 pieejas kā uzlabot modeļu trenēšanu nenomarķētiem datiem. Pirma pieeja mēģina prognozēt kas būs tālāk vārdu secībā. Otra pieeja izmanto autoenkoderi, kas enkodē teikumu vektorā un mēģina atkodēt to pašu vektoru uz sākotnējo teikumu. Abas pieejas nepieprasa nomarķētus datus, kā arī cilvēku resursus, lai sagatavotu datus. Metodes var būt izmantotas pirmstrenēšanas fāzē, lai modeļu svari būtu labāk novietoti sākotnējā pozīcijā un pēc tam precizēti trenēšanas laikā.

1.3. Attention

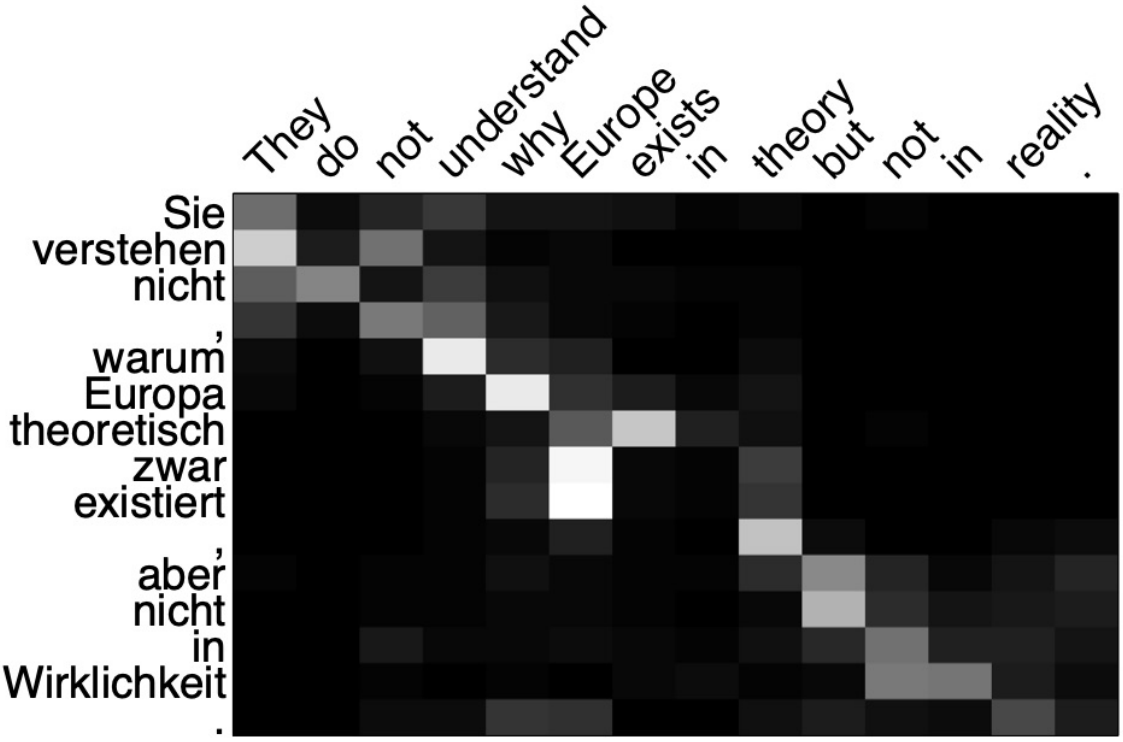
Neskatoties uz to, ka sequence-to-sequence modelis bija liels soļis mašīntulkošanā un ievērojami uzlaboja tulkošanas rezultātus, kā arī uz kādu laiku kļuva par «state-of-the-art» modeļiem dažas nozares (Chiu et al, 2018). Šajā modeļī bija liels trūkums - teikuma garums. Problēmas cēlonis bija paša konteksta vektorā, jo vajadzēja aptvert visu teikumu kontekstu vektorā ar fiksētu garumu. Līdz ar to bija piedavāts risinājums darbos Bahdanau et al., «Neural Machine Translation by Jointly Learning to Align and Translate» un Luong et al., «Effective Approaches to Attention-based Neural Machine Translation». Šie darbi ieviesa un pilnveidoja metodi un nosauca to par «Attention» (latv. val. «uzmanība»), kas uzlaboja mašīntulkošanas kvalitāti. Piedāvātājs risinājums ļauj modeļim pēc vajadzības koncentrēties uz attiecīgajām teikuma daļām un vārdiem.

Uzmanības modelis atšķiras no klasiskā sequence-to-sequence modeļa divos veidos. Pirmkārt, enkodētājs nosūta dekodētājam daudz vairāk datu. Enkodētājam vajag nosūtīt visus slēptus stāvokļus dekoderam, lai pārtulkotu teikumu. Sequence-to-sequence modeļī bija nepieciešams nosūtīt tikai pēdējo slēpto stāvokļi. Otrkārt, uzmanības dekodētājs veic papildu darbību, pirms izveidot pārtulkotu rezultātu. Lai pievērstu uzmanību tikai šim teikuma daļām, kas attiecas uz šo dekodēšanas laika posmu, dekodētājs rīkojas šādi:

1. Apskata saņemtu enkodēto slēpto stāvokļu kopu. Katrs enkodētais slēptais stāvoklis ir visvairāk saistīts ar konkrētu vārdu ieejas teikumā;
2. Piešķir katram slēptajam stāvoklim punktu skaitu;
3. Sareizina katru slēpto stāvokli un to punktu skaitu (pielietojot softmax transformāciju). Ar sekojošu darbību pastiprinās slēptie stāvokļi ar augstu punktu skaitu un pavājina slēptus stāvokļus ar zemu punktu skaitu.

Šīs darbības rezultāts ir vektors, uz kuru ietekmēja visi slēpti stāvokļi dažādās proporcijās. Šādas darbības piemērs:

1. h1, h2, h3 - Slēptie stāvokļi no enkodētāja;
2. 13, 9, 9 - Katram stāvokļim piešķirts punktu skaits;
3. 0.96, 0.02, 0.02 - softmax(x) funkcijas rezultāts katram piešķirtam punktu skaitam;
4. 0.96 * h1 + 0.02 * h2 + 0.02 * h3 - beiguma konteksta vektors



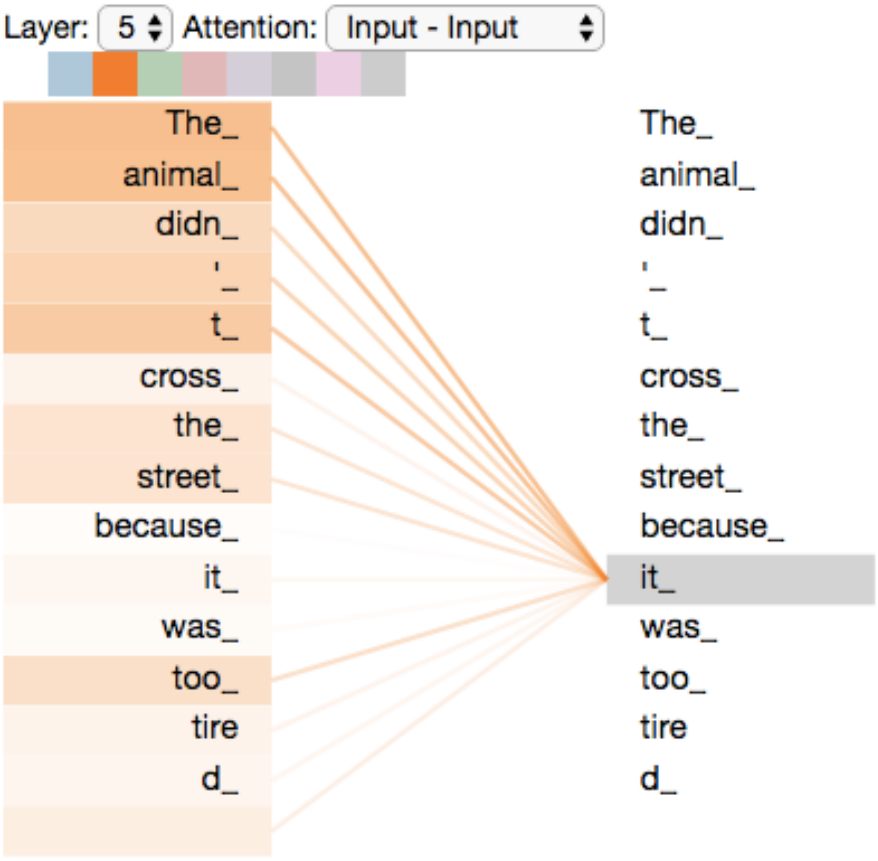
1.2. attēls. Uzmanības svars teikuma vārdiem (Luong et al, 2015)

Augstāk minētā piemērā ir redzams kā darbojas «Attention» mehānisms. Uz rezultējošu vektoru ietekmēja visi slēptie stāvokļi, kas atnāca no enkodētājā un teikumā tulkojumā piedalās visi teikuma vārdi. Šis punktu skaitīšanas uzdevums tiek veikts katrā dekodētāja solī.

Slēpto kontekstu reizināto ar atbiltošu softmax punktu skaitu bieži attēlo matricas veidā. Piemēram, 1.2. attēlā ir parādīts teikuma tulkojums no angļu uz vācu valodu. Ar spilgtākiem taisnstūriem ir apzīmēti vārdi, kas vairāk ietekmēja tulkotu vārdu. Kā redzams, matrica pārsvarā ir diagonālā. Tas tiek izskaidrota ar to, ka lielākais svars ir pievērsts tiešajam vārdu tulkojumam un tikai pēc tam tas tiek precizēts ar pielietojumu kontekstu.

1.4. Self-Attention

Termins «self-attention» tika aprakstīts darbā Vaswani et al., «Attention Is All You Need». Self-attention terminu ir iespējams izskaidrot ar iepriekš ieviestu terminu «attention». Attention nozīme to cik liels svars ir iepriekš minētiem vārdiem, bet self-attention parāda attiecības starp vārdiem vienā un tajā pašā teikumā.



1.3. attēls. Self-attention vārdam «it» (Alammar, 2018)

Aplūkot terminu «self-attention» labāk uz konkretā piemēra. Attēlā 1.3. ir apskatīts teikums «The animal didn't cross the street because it was too tired» un vārda «it» self-attention vektora attiecības. Ar tumšo krāsu ir parādīti vārdi, kas vairāk attiecas uz vārdu. Ja cilvēks apskatīs šo teikumu un viņam uzdos jautājumu uz kādu vārdu attiecas vārds «it», cilvēkam ir viegli atbildēt uz šo jautājumu, ja ir atbilstošas zināšanas angļu valodā. Vārds «it» attiecas uz vārdiem «the animal». Šī informācija viegli padodas cilvēkiem, bet algoritmam izskaidrot to nav tik viegli. Kad modelis apstrādā katru vārdu, self-attention metode ļauj apskatīt vārdus arī uz citām pozīcijam teikumā, lai iegūtu labāku atbilsošu vektoru šim vārdam.

Pirmais solis self-attention skaitļošanā ir izveidot 3 vektorus katram vārdam tulkojamā teikumā. Katram vārdam vajag izveidot «Query» vektoru, «Key» vektoru un «Value» vektoru. Šie vektori tiek izveidoti, reizinot vārdu vektoru ar trim matricām, kuras ir nepieciešams apmācīt trenēšanas procesā. Šie vektori ir mazāki, nekā slēpta konteksta (hidden context) vektori, kas ir aprakstīti agrāk. Šo vektoru lielums ir 64 cipari. Vektorus var apreķināt pēc sekojošām formulām:

$$q_i = W^q \cdot x_i \tag{2.1}$$

$$k_i = W^k \cdot x_i \tag{2.2}$$

$$v_i = W^v \cdot x_i \tag{2.3}$$

kur, q_i, k_i, v_i - Query, Key un Value vektori,

W^q, W^k, W^v - Query, Key un Value trenējamas matricas,

x_i - i-ta vārda vektors

Pēc vektoru apreķināšanas, otrais solis self-attention aprēķināšanā ir «score» aprēķināšana. Šajā soļī ir nepieciešams novērtēt katru teikuma vārdu pret šo vārdu. Score nosaka, cik daudz uzmanības jāpievērš citām teikuma daļām, kad tiek iekodēts tekošs vārds. Score tiek aprēķināts, ņemot skalāro Query vektora reizinājumu ar attiecīgā vārda Key vektoru.

Trešais un ceturtais solis ir score dalīšana ar $\sqrt{d_k}$, kur d ir Key vektora dimensijas skaits. Tā kā darbā tika izvelēts 64, tad $\sqrt{d_k} = 8$. Tas noved pie tā, ka rezultējošie gradienti ir stabilāki. Šeit varētu būt arī citas iespējamās vērtības, taču šī vertība tiek izmantota pēc noklusējuma. Pēc tam score rezultātu vajag pārveidot, izmantojot softmax operāciju. Softmax normalizē rādītājus, lai tie visi būtu pozitīvi un būti mazāki nekā 1.

Šis softmax vērtējums nosaka, cik daudz katrs vārds tiks izteikts šajā pozīcijā. Ir skaidrs, ka vārdam, kas atrodas šajā pozīcijā būs visaugstākais softmax vērtējums (gandrīz diagonāla matrica, sk. 1.3. att.), taču dažreiz ir noderīgi apmeklēt arī citu vārdu, kas attiecas uz pašreizējo vārdu.

Piektais solis ir reizināt katru Value vektoru ar softmax score lielumu. Šādai darbībai ir sekojošs merķis - mazāk ietekmēt tās vārdu vērtības, uz kurām vēlamies koncentrēties un pazemināt vārdu vērtības, uz kurām nevajag koncentrēties, un kuras ietekme uz tekošo vārdu ir niecīga. Lai to panāktu, vajag reizināt vārdu vektorus ar nelieliem cipariem, piemēram, 0.001.

Sestais solis ir summēt novērtētus Value vektorus. Rezultāts nosaka self-attention slāņu vertību tekošam vārdam.

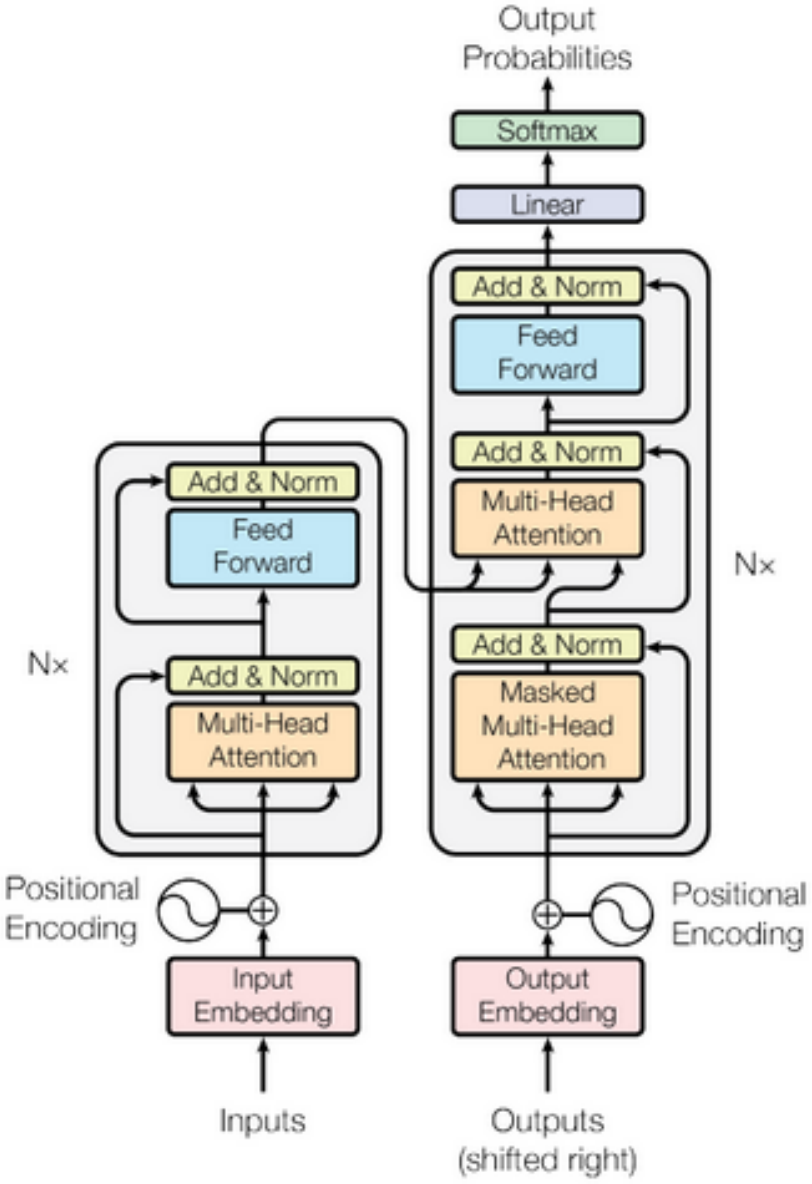
1.5. ELMo

Standarta pieejā, tādā kā GloVe, lai dabūtu vārdu vektoru, vienam vārdam atbilst viens vektors. Tā ir problēma, kad vienam un tam pašam vārdam var būt dažādas nozīmes. Un tā nozīme ir atkarīga no teikuma konteksta. Tādi vārdi tiek saukti par homonīmiem. Lai atpazītu homonīmus vienu no otra ir nepieciešams izmantot teikuma kontekstu. Tas arī tiek darīts Peters et al., «Deep contextualized word representations» darbā un tika piedavāts «Embedding from Language Model» (ELMo) modelis. Šis modelis apskata visu teikumu un pēc tam izrēķina vektoru konkretam vārdam. Modelis izmanto LSTM tīklu, kas ir trenēts uz specifiskiem uzdevumiem, lai varētu izveidod šos vektorus. Tīkls ir uztrenēts uz liela datu apjoma, līdz ar to ir iespējams to izmantot kā pastāvīgo komponentu citos risinājumos. ELMo tiek trenēts izmantojot «unsupervised learning», tas nozīmē, ka pastāv iespēja izmantot nenomarķētus datus. Trenēšanas laikā vajadzēja noteikt nākamo vārdu, šādā veidā tika izveidots valodas modelis. Jāpiemin, ka modeļī tiek izmantots «bi-directional LSTM», tas nozīme, ka modelim nav svarīgi kādā virziena lasīt, modelis spēj uztvert uzreiz visu teikumu.

1.6. Transformer

Attēlā 1.4. tika ilustrēts «Transformer» arhitektūras piemērs. Sākumā tas tiek fokusēts uz transformera enkodētāja un dekodētāja daļām, kas ir ilustrēti ar pēlēkiem taisnstūriem 1.4. attēlā. Pa kreisi ir enkodētājs un pa labi ir dekodētājs. Transformer arhitektūra tika piedāvāta iepriekš minētajā darbā Vaswani et al., «Attention Is All You Need». Enkodētāja un dekodētāja bloku faktiski ir daudz, nevis viens. Attēla 1.4. tas tika apzīmēts ar Nx simbolu.

Tie ir identiski enkodētāji un dekodētāji, kas apkopoti viens virs otra. Enkodētāju un dekodētāju skaits ir arhitektūras hiperparametrs. Iepriekš minētajā rakstā tika izmantoti 6 enkodētāji un dekodētāji.



1.4. attēls. Transformer arhitektūra (Vaswani et al, 2017)

Enkodētāju un dekodētāju bloks strādā sekojoši:

1. Vārdu vektors no ieejas teikuma tiek padots uz enkodētāju;
2. Vektori tiek pārveidoti un tiek nodoti uz nākošo enkodētāju;
3. Rezultāts no pēdēja enkodētāja tiek nosūtīts uz visiem dekodētājiem.

Enkodētājā un dekodētājā galvenie komponenti ir «self-attention» komponenti, kas 1.4. attēlā ir apzīmēti ar «Multi-Head Attention». Atšķirībā no iepriekš aprākstītām «self-attention» komponentēm, šis ir tādas, ka «Multi-Head Attention» izmanto vairākus «self-

attention» komponentes un beigās apvieno tās uz vienu rezultējošo self-attention komponenti. Vairāku self-attention komponentu skaits darbā ir pamatots sekojoši: «Multi-Head attention ļauj modeļim kopīgi apkopot informāciju no dažādām reprezentācijas apakšgrupām dažādās pozīcijās.» (Vaswani et al, 2017). To saprot tā, ka trenēšanās laikā viena self-attention slāņa izmantošana ir pakļauta datu pārtrenēšanai (data overfitting).

Transformer arhitektūra ir liels uzlabojums salīdzinājumā ar seq2seq modeļiem. Bet tiem arī ir savi ierobežojumi:

- Uzmanību ir iespējams pievērst tikai tekstam ar fiksētu garumu. Pirms ievadīt tekstu sistēmā, tas tiek sadalīts uz noteiktu segmentu skaitu.
- No pirmā punkta iziet otrais ierobežojums - šī teksta šķelšanās izraisa konteksta fragmentāciju. Piemēram, gadījumā, kad teikums tiek sadalīts pa vidu, tad tiek zaudēts ievērojams konteksta daudzums.

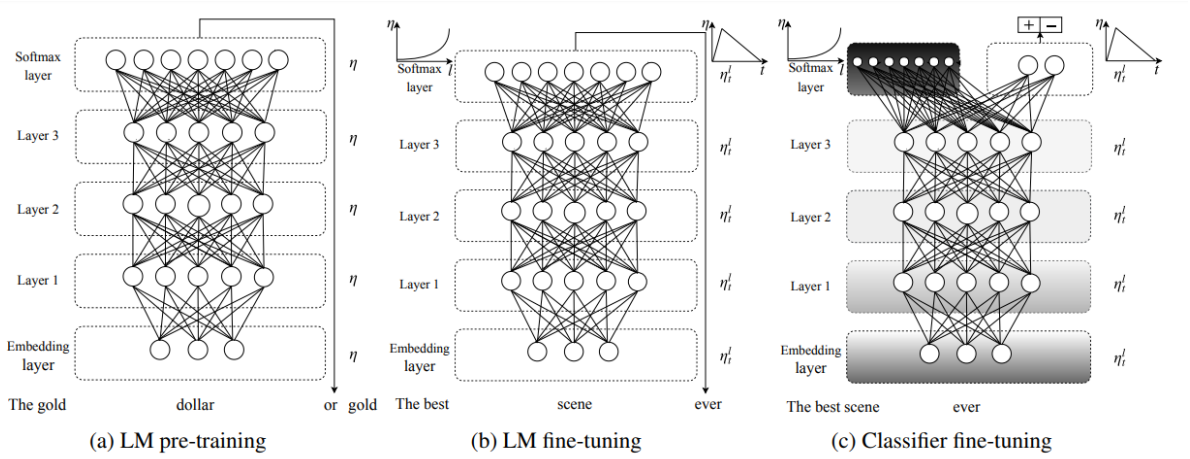
Lai apietu šos ierobežojumus darbā Dai et al., «Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context» tika piedāvāta jauna pieeja - izmantot palielinātu Transformer - XL. Šajā arhitektūrā iepriekšējos segmentos iegūtie slēptie stāvokļi tiek atkārtoti izmantoti kā informācijas avoti pašreizējam segmentam. Tas ļauj modelēt ilgtermiņa atkarību, jo informācija var plūst no viena segmenta uz nākamo.

Ir arī citi risinājumi, Al-Rfou et al., «Character-Level Language Modeling with Deeper Self-Attention» ierosināja ideju kā piemērot Transformer modeli valodas modelēšanai. Kā norādīts rakstā, visu korpusu var sadalīt fiksētos lieluma segmentos, kurus viegli pārvaldīt. Pēc tam uz tiem segmentiem patstāvīgi trenē Transformer modeli, ignorējot visu kontekstuālo informāciju no iepriekšējiem segmentiem. Šīs arhitektūras pieeja nav raksturīga gradientu izsīkšanas problēma (vanishing gradients). Bet konteksta fragmentācija stipri ierobežo tā ilgtermiņa atkarības mācīšanu. Novērtēšanas posmā (evaluation step) segments tiek nobīdīts uz vienu pozīciju. Jaunais segmentu ir jāapstrādā pilnībā no nulles. Šī novērtēšanas metode diemžēl ir diezgan dārga skaitļošanai.

1.7. UMLFiT

Datorredzē eksistē induktīvie algoritmi, lai pārnestu trenēšanas rezultātus uz citiem modeliem vai izmantotu vienus un tos pašus svarus vairākiem modeliem. Bet valodu apstrādē ļoti ilgi tādu algoritmu nebija. ULM-FiT ieviesa tādas metodes, ka efektīvi izmantot to, ka modeli apgūt pretrenēšanas laikā. Šis metodes tika aprakstītas Jeremy Howard un Sebastian

Ruder darbā «Universal Language Model Fine-tuning for Text Classification». Metodes dalās uz 3 posmiem - pretrenēšana, precizēšana (fine-tuning) un klasifikatora precizēšana.



1.5. attēls. UMLFiT posmi (Howard et. al, 2018)

Pretrenēšana ir visdārgāka operācija pēc resursiem un laika. Tā pieprasa daudz dažādu datu. Piemēram, darbā ir pieminēts, ka tiek izmantoti Wikitext-103 dati, kur ir 103 miljoni vārdu. Precizēšanas laikā modelis tiek trenēts, lai labi izpildītu konkrētu uzdevumu un arī izmanto specifiskus datus, bet bāze ir kopīga. Pirmajos posmos tiek sastādīts valodu modelis, bet pēdējā solī trenējas un precizējas pats klasifikators.

Šī metode ir universāla, tāpēc ka atbilst šiem kritērijiem:

- Tas darbojas dažādos uzdevumos, kas atšķiras pēc dokumenta lieluma, numura un marķēšanas veida;
- Tas izmanto vienotu arhitektūru un apmācības procesu;
- Tam nav nepieciešami specializēti izgudrojumi vai pirmsapstrāde;
- Tam nav nepieciešami papildus specializēti dokumenti vai marķeri.

1.8. BERT

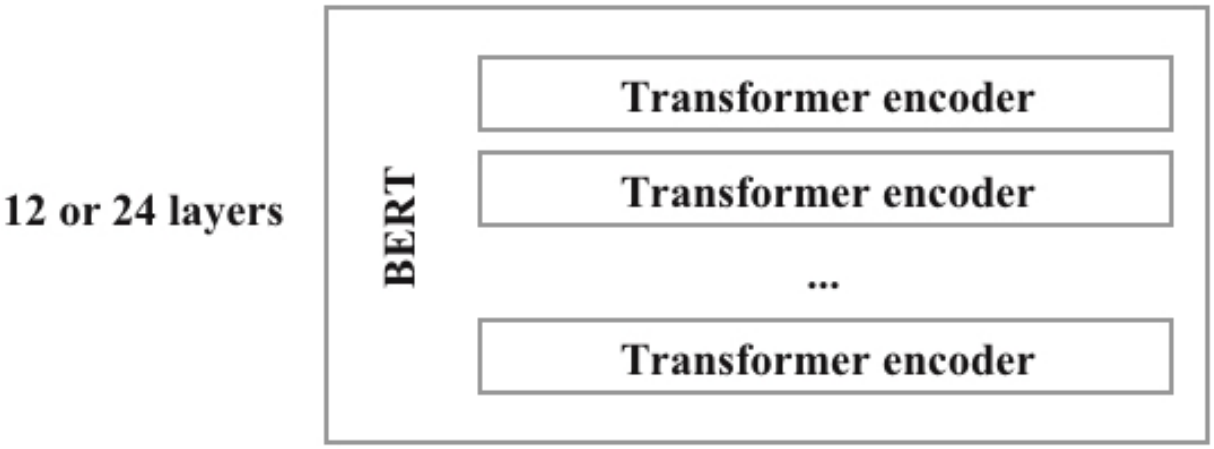
BERT modelis izstradāja Google 2018. gadā. Tās apvienoja sevī - «semi-supervised sequence learning», «attention» un «self-attention», «ELMo», «transformer», «UMLFiT». Modelis ir aprakstīts Devlin et al., «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding» darbā.

Vienkāršākais BERT izmantošanas veids ir tā pielietošana, lai klasificētu kādu teksta daļu vai teikumu. Lai uztrenētu modeli nepieciešams tikai precizēt (fine-tune) klasifikatoru.

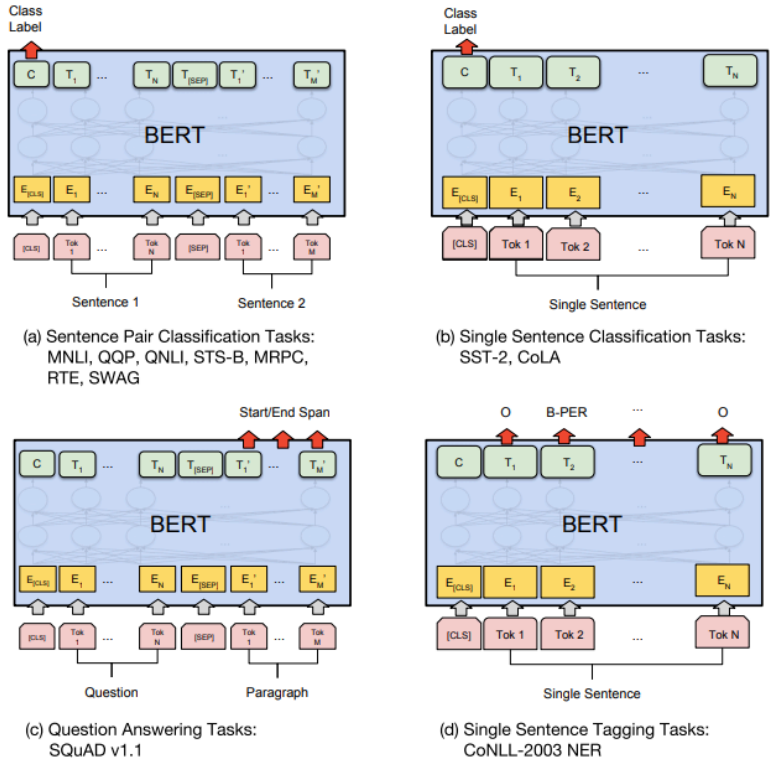
Ideja ir ņemta no «Semi-supervised Sequence Learning» un «ULMFiT». Pastāv iespēja modeli izmantot arī sentimenta analīzē, faktu pārbaudē un valodu modeļu reprezentācijā.

Darbā ir izlaisti divi modeļu varianti:

- BERT Base: Transformer enkoderu slāņu skaits = 12, Parameteru kopumā = 110M;
- BERT Large: Transformer enkoderu slāņu skaits = 24, Parameteru kopumā = 340M.



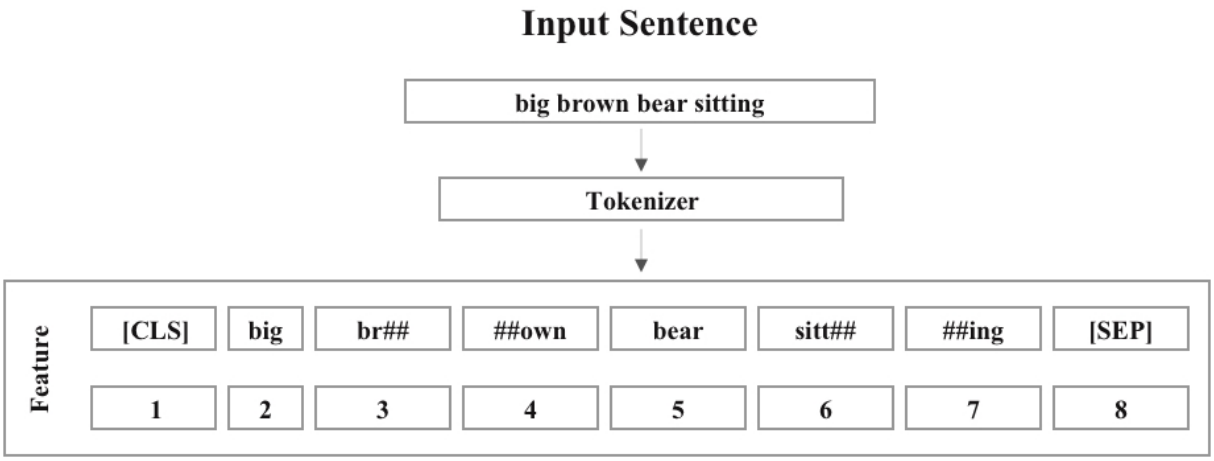
1.6. attēls. BERT struktūra.



1.7. attēls. BERT ieejas struktūra.

Modeļu ieejas dati pieprasa speciālu pirmsapstrādi. Ir nepieciešams sadalīt ieejas teikumu uz vārdu daļām jeb tokeniem un katra teikuma sākumā vajag iekļaut specializētu

tokenu - [CLS]. Šis tokens diezgan bieži tiek izmantots, lai reprezentētu visu teikumu, bet kā redzams zemāk, parasti šāda pieeja nav precīza un nav labāka.



1.8. attēls. BERT izmantošanas piemēri (Devlin et. al, 2019).

Tāpat kā «Transformer» enkoders, BERT ņem vārdu secību un padod to uz enkodera ieeju, tā secība iziet cauri visiem enkodetājiem. Katrs slānis pievieno «self-attention» un padod rezultātus tālāk caur visu tīklu. Līdz šim brīdim, process atkārto «transformer». Atšķiras tikai izmēri. Izejas stāvoklī katrs tokens ir aizvietots ar vektoru, BERT Base gadījumā - vektora lielums ir 768.

Trenēšanas laikā BERT izmanto 2 uzdevumus:

- Vārdu maskēšana un atgādināšana pēc teikuma konteksta;
- Teikumu saistības noteikšana.

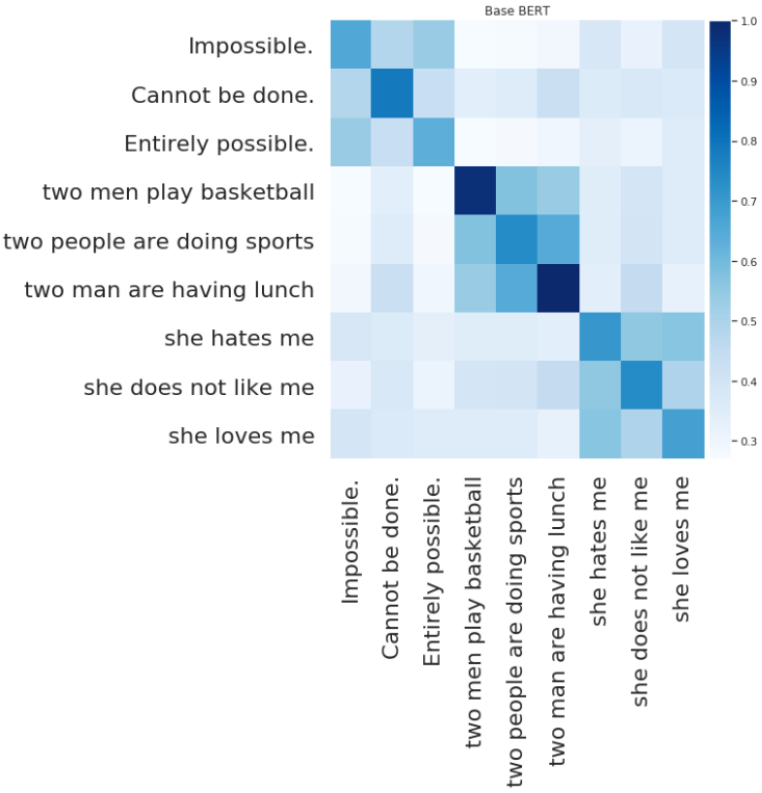
Rakstā autori arī piemin kādos uzdevumos BERT sasniedza labākos rezultātus (1.7.att):

- Teikumu pāra klasifikācija;
- Teikumu klasifikācija;
- Jautājumu atbildēšana;
- Teikumu marķēšana.

Pastāv iespēja izmantot BERT, lai uzbūvētu valodas modeli. Šī ideja tika attīstīta darba praktiskajā daļā.

1.9. SBERT

Kā bija pieminēts iepriekš, BERT'ā ļoti bieži izmanto speciālu tokenu [CLS] kā teikumu reprezentāciju. Rezultāti nav labāki un to arī parada Denisa Antyunhova ekperiments «Improving sentence embeddings with BERT and Representation Learning»



1.9. attēls. BERT vektoru līdzība (Antyunhov, 2020).

Eksperimentā ir izreķināts cik līdzīgi ir teikumi (1.9. att.). Izrādās, ka teikumi ir vairāk saistīti leksiski, nevis semantiski. Tas nozīme to, kā BERT pievērš uzmanību vārdu līdzībai.

Lai iegūtu teikuma vektoru izmanto dažādas pieejas. Visvienkāršāka ir paņemt [CLS] tokena vektoru un pieņemt to kā teikuma vektoru. Parasti tas der vienkaršas klasifikācijas uzdevumos, kad teikumi ir ļoti dažādi gan leksiski, gan sintaktiski. Cita pieeja tika izmantota «bert-as-service» bibliotēkā. Tur ir dažādas pieejas, piemēram, paņemt pēdējos 2 slāņus katram tokena vektoram un izrēķināt vidējo vektoru. Arī ir iespējamas citas stratēģijas - paņemt maksimālo vektoru no visiem tokeniem vai paņemt [SEP] tokena vektoru.

2019. gadā Nils Reimers un Iryna Gurevych darbā «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks» piedāvāja jauno pieeju - SBERT. Darbā tika uztrenēts jauns modelis, kas balstās uz BERT vektoriem. Darbā izmanto SNLI datus. Šie dati ir marķēti vai teikums ir pozitīvs, negatīvs vai neitrāls. Modelis trenēja 3 vektorus paralēli un beigās apvienoja rezultātus vienā vektorā. Pirmais vektors ir vienkārši apvienoti visi tokena vektori, kuri ir reizināti ar citu vektoru, kas tiek precizēts trenēšanas laikā. Otrais ir vektors, kas rāda cik līdzīgi ir teikumi. Tas izrēķina vidējo kvadrātisko kļūdu. Un pēdējais ir vektors, kas minimizē attālumu starp bāzes teikumu un pozitīvu un negatīvu teikumu. Beigās visi 3

vektori tiek apvienoti vienā. Modeļa rezultāti kā arī salīdzinājums ar citiem modeļiem ir pieejami un apkopoti 1.1. tabulā.

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

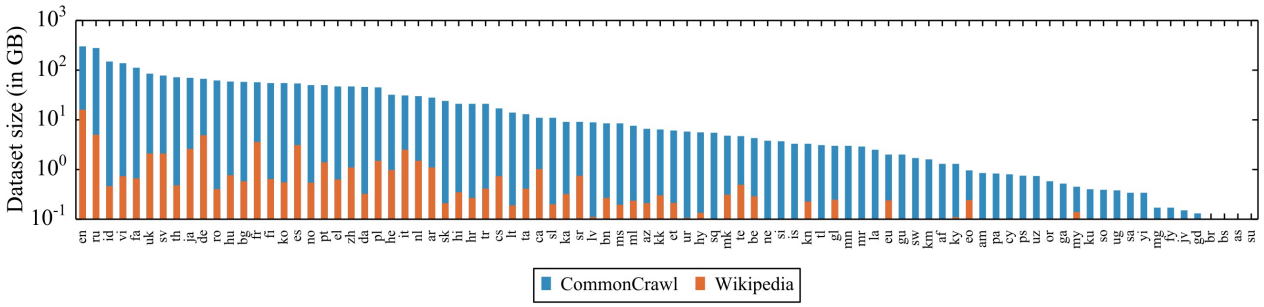
1.1. tabula. SBERT rezultāti dažados teikuma līdzības testos (Reimers, Gurevych, 2019).

1.10. XLM-R

XLM-R modeli izstradāja Facebook AI komanda. Šis modelis, tāpat kā BERT arī balstās uz «Transformer» modeli. XLM-R fokusējās uz daudzvalodu uzdevumiem un ievērojami pārspēj daudzvalodu BERT modeli. Modelis tiek trenēts vairāk nekā uz 2 terabaitu filtrētiem «CommonCrawl» un «Wikipedia» datiem un atbalsta simts valodas, ieskaitot latviešu valodu. Kods, modelis un dati, kas tiek izmantoti trenēšanā ir publiski pieejami šeit:

https://github.com/facebookresearch/XLM

Modeļu trenēšanā tiek izmantots «Transformer» modelis, kas tiek trenēts lielā datu apjomā. Tiek izmantoti monoligvistiski dati. Lai izveidotu valodas modeli, tiek izmantots vārdu vai vārdu daļu maskēšana. Kā arī, tiek izmantots «subwords» tokenizators ar vārdnīcas lielumu 250k. Datu apjoma sadalījums, kas tiek izmantots trenēšanā ir paradīts 1.10. attēlā.



1.10. attēls. Datu apjoms (GB) XLM-R modeļu trenēšanā (Conneau&Khandelwal et al, 2020).

Kā ir redzams, latviešu valodā parsvarā izmanto «CommonCrawl» datus un minmālais datu apjoms, kas ir nepieciešams, lai uztrenētu modeli ir pāris simti megabaiti.

Galvena problēma ar kuru sastopas pētnieki ir kā optimizēt modeli, lai tas strādātu daudz-resursu valodām, gan maz-resursu valodām. Daudz un maz-resursu valodas ir valodas,

kam ir daudz vai maz publiski pieejamu resursu. Piemēram, angļu, krievu un franču valodas ir daudz-valodas resursi. Savukārt, svahili un urdu ir maz-resursu valodas. Lai to panāktu, pētnieki ekperimentēja ar dažādiem parametriem, kā tie ietekmē rezultātus. Piemēram, vārdnīcas lielums, teikumu paraugu ņemšana atkarība no izlases lieluma, utt.

Modeļa rezultāti dažādos NLP uzdevumos ir paradīti tabulā 1.2.

Model	D	#M	#lg	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	Avg
Fine-tune multilingual model on English training set (Cross-lingual Transfer)																			
Lample and Conneau (2019)	Wiki+MT	N	15	85.0	78.7	78.9	77.8	76.6	77.4	75.3	72.5	73.1	76.1	73.2	76.5	69.6	68.4	67.3	75.1
Huang et al. (2019)	Wiki+MT	N	15	85.1	79.0	79.4	77.8	77.2	77.2	76.3	72.8	73.5	76.4	73.6	76.2	69.4	69.7	66.7	75.4
Devlin et al. (2018)	Wiki	N	102	82.1	73.8	74.3	71.1	66.4	68.9	69.0	61.6	64.9	69.5	55.8	69.3	60.0	50.4	58.0	66.3
Lample and Conneau (2019)	Wiki	N	100	83.7	76.2	76.6	73.7	72.4	73.0	72.1	68.1	68.4	72.0	68.2	71.5	64.5	58.0	62.4	71.3
Lample and Conneau (2019)	Wiki	1	100	83.2	76.7	77.7	74.0	72.7	74.1	72.7	68.7	68.6	72.9	68.9	72.5	65.6	58.2	62.4	70.7
XLM-R$_{Base}$	CC	1	100	85.8	79.7	80.7	78.7	77.5	79.6	78.1	74.2	73.8	76.5	74.6	76.7	72.4	66.5	68.3	76.2
XLM-R	CC	1	100	**89.1**	**84.1**	**85.1**	**83.9**	**82.9**	**84.0**	**81.2**	**79.6**	**79.8**	**80.8**	**78.1**	**80.2**	**76.9**	**73.9**	**73.8**	**80.9**
Translate everything to English and use English-only model (TRANSLATE-TEST)																			
BERT-en	Wiki	1	1	88.8	81.4	82.3	80.1	80.3	80.9	76.2	76.0	75.4	72.0	71.9	75.6	70.0	65.8	65.8	76.2
RoBERTa	Wiki+CC	1	1	**91.3**	82.9	84.3	81.2	81.7	83.1	78.3	76.8	76.6	74.2	74.1	77.5	70.9	66.7	66.8	77.8
Fine-tune multilingual model on each training set (TRANSLATE-TRAIN)																			
Lample and Conneau (2019)	Wiki	N	100	82.9	77.6	77.9	77.9	77.1	75.7	75.5	72.6	71.2	75.8	73.1	76.2	70.4	66.5	62.4	74.2
Fine-tune multilingual model on all training sets (TRANSLATE-TRAIN-ALL)																			
Lample and Conneau (2019)†	Wiki+MT	1	15	85.0	80.8	81.3	80.3	79.1	80.9	78.3	75.6	77.6	78.5	76.0	79.5	72.9	72.8	68.5	77.8
Huang et al. (2019)	Wiki+MT	1	15	85.6	81.1	82.3	80.9	79.5	81.4	79.7	76.8	78.2	77.9	77.1	80.5	73.4	73.8	69.6	78.5
Lample and Conneau (2019)	Wiki	1	100	84.5	80.1	81.3	79.3	78.6	79.4	77.5	75.2	75.6	78.3	75.7	78.3	72.1	69.2	67.7	76.9
XLM-R$_{Base}$	CC	1	100	85.4	81.4	82.2	80.3	80.4	81.3	79.7	78.6	77.3	79.7	77.9	80.2	76.1	73.1	73.0	79.1
XLM-R	CC	1	100	**89.1**	**85.1**	**86.6**	**85.7**	**85.3**	**85.9**	**83.5**	**83.2**	**83.1**	**83.7**	**81.5**	**83.7**	**81.6**	**78.0**	**78.1**	**83.6**

1.2. tabula. XLM-R modeļa rezultāti starpvalodas klasifikācijas uzdevumam (Conneau&Khandelwal et al, 2020).

Viens no lielākiem sasniegumiem šajā darbā ir tas, ka ir pierādīts, ka modeļi, kas trenēti daudzvalodas uzdevumiem sasniedz labākus rezultātus, nekā monolingistiskie modeli. Monolingvistiska BERT modeļu sasniegumu rezultātu salīdzinājums ar XLM-R modeli ir parādīts tabulā 1.3. Arī pētījuma rezultāti parāda cik svarīga ir hiperparametru precizēšana un izvēle.

Model	D	#vocab	en	fr	de	ru	zh	sw	ur	Avg
Monolingual baselines										
BERT	Wiki	40k	84.5	78.6	80.0	75.5	77.7	60.1	57.3	73.4
	CC	40k	86.7	81.2	81.2	78.2	79.5	70.8	65.1	77.5
Multilingual models (cross-lingual transfer)										
XLM-7	Wiki	150k	82.3	76.8	74.7	72.5	73.1	60.8	62.3	71.8
	CC	150k	85.7	78.6	79.5	76.4	74.8	71.2	66.9	76.2
Multilingual models (translate-train-all)										
XLM-7	Wiki	150k	84.6	80.1	80.2	75.7	78	68.7	66.7	76.3
	CC	150k	**87.2**	**82.5**	**82.9**	**79.7**	**80.4**	**75.7**	**71.5**	**80.0**

1.3. tabula. Monoligvistisks BERT salīdzinājums ar XLM-R (Conneau&Khandelwal et al, 2020).

2. PRAKTISKA DAĻA

2.1. Datu sagatavošana

Lai uztrenētu modeli, kas spēj tulkot tekstus ir nepieciešams lielu tekstu korpus, kas ir tulkots paralēli divās valodās. Ideālā gadījumā teksts ir sadalīts pēc teikumiem, kur katrs teikums atrodas atsevišķā rindā. Darbā tika izmantoti divi datu avoti - dati no Mašīntulkošanas konferences, Kopengagenā, 2017. gadā (MWT17). http://statmt.org/wmt17/translation-task.html un MS-COCO annotācijas dati http://cocodataset.org. Latviešu - angļu valodas dati ir apkopoti un sagatavoti ar Latvijas Universitātes un Tildes uzņēmuma palīdzību.

Kopumā ir pieejami sekojoši dati:

- Dati no Eiroparlamenta sesijām 2006. gadā - 93 mb;
- «Ardievas ieročiem» stāsts - 440 kb;
- Eiropas Investīcijas banka sesija 2007. gadā - 33 mb;
- Eiroparlamenta komisijas sēde - 242 mb;
- MS-COCO pārtulkoti dati - 32 mb.

No šiem datiem tika izveidots lielākais modelis. Kā ir redzams dati ir dažādi, bet tie ir viendabīgi, politiski un ekonomiski virzīti. Tas arī tika atspoguļots «Transformer» modelī. Tas labi pārtulko ekonomisko un politisko terminoloģiju, bet parastos un ikdienišķos vārdus modelis nespēj kvalitatīvi pārtulkot. MS-COCO dati ir vienīgais korpus, kur ir sastopami vienkāršie, ikdienas vārdi. Šie dati ir veidoti kā anotācijas, kas apraksta, kas ir attēlots uz bildes. «Transformer» modelis trenēts tikai uz šiem datiem, līdz ar to tas saprot vienkāršus teikumus. Bet kopumā šie dati aizņem tikai 8% no kopēja datu korpusa un parasta terminoloģija nebūs dominējoša, ja būs nepieciešams trenēt uz visiem datiem, kā «Transformer» modeli.

Datu tīrīšanas programma ir pašrakstīta un brīvi pieejama Github'ā: https://github.com/emukans/europarlament-clean. Programma ir sadalīta 3 daļās:

1. Sākumā dati tika lejupieladēti no MWT17. Gadījumā, ja fails jau eksistē lokalā direktorijā, tad programma izlaiž failu un turpina strādāt, kamēr visi faili netiks ielādēti. Funkcionalitāte ir izpildīta «maybe_download_and_extract_data» funkcijā. Tā nepieņem argumentus, bet izmanto globālo mainigu - «data_source». Šis mainīgais apkopo visus pieejamus datu avotus formatā - avotu links, faila nosaukums;

2. Nākamais solis iztīrīt datus no liekas informācijas. Šī funkcionalitāte ir izpildīta «maybe_clean_data» funkcijā. Funkcija arī nepieņem nekādus argumentus, bet ņem globālo mainīgo - «data_to_clean». Mainīgais satur sarakstu ar latviešu datu failu nosaukumu un angļu datu failu nosaukumu. Šī funkcija iziet cauri visiem teikumiem un izņem arā dažus simbolus, vārdus vai vispār izlaiž visu teikumu. Teikumu piemēri, kas tika izņemti ir sekojoši - «E-mail: info@eib.org;», «tel.: (+352) 43 79 – 22000;», «fax: (+352) 43 79 – 62000», u.c. Šādi teikumi bieži tiek sastapti politiskajos datos un nedot nekādu valodas specifiska rakstura informāciju. Kā arī tekstos ir ļoti daudz standartu teikumu, ar kuriem apzīme kaut kādu sadaļu vai parādību. Piemēram, šie ir teikumi - «Press Releases:», «Oral question:», «Chapter II», «To the commision:», u.c. Visos iepriekšējos gadījumos vesels teikums tika izdzēsts, bet arī teikumā ietvaros ir simboli, kuriem nav nekādas jēgas teikuma saprašanas vai konteksta ziņā. Sekojoši simboli tika izdzēsti no teikuma, bet pats teikums paliek iztīrītā datu korpusā - (,), -, ”, :, :, u.c. Kā arī tika izlaisti ciparu punkti, piemēram - 1., 2., u.c. Un beigās ir izdzēstas visas liekas atstarpes, punkti, komati un tukšie teikumi. Kad datu tīrīšana ir pabeigta, programma saglabā rezultātu atsevišķā direktorijā «cleaned».
3. Pēdējais solis ir datu verifikācija. Tā notiek manuāli. Terminālā parādās 10 nejauši izvēlēti teikumi no katra datu korpusa un tam atbilsošais tulkojums latviešu valodā.

Vajag arī pieminēt, ka visas operācijas ar datu atfiltrēšanu un tīrīšanu notiek sākumā ar angļu valodu un pēc tam šis operācijas ir dublētas latviešu valodā. Tas ir izdarīts tāpēc, ka angļu valodā nav locījumu, garumzīmju un dati vairāk konsistenti, ir vieglāk atrast paternus un atfiltrēt tos.

Pēc datu tīrīšanas un filtrēšanas tika iegūti 3.5 miljoni teikumu, kas ir pietiekami, lai uztrenētu lielu transformer modeļi.

2.2. Transformer arhitektūras pielietojums

Praktiska darba ietvaros ir izstrādāta mašīntulkošanas programma, kas spēj pārtulkot angļu valodas tekstu uz latviešu valodu. Kā arhitektūra ir izvelēta «Transformer» arhitektūra, jo tā ir relatīvi jauna un to diezgan bieži izmanto tulkošanai citas valodās. Kursa darba ietvaros ir izmantots jau gatavs transformer modelis. Kā pamats tika izmantota bibliotēka, kas minēta darbā Vaswani et al., «Attention Is All You Need» - tensor2tensor. Šī bibliotēka tiek izmantota ne tikai NLP uzdevumiem, bet arī attēlu klasifikācijai un ģenerēšanai, sentimentu analīzei, jautāšanai un atbildēšanai utt. Bibliotēka ir izstrādāta «Google Brain» komandā, lai

paātrinātu mašīnmacīšanas procesu. Bibliotēka ir brīvi pieejama GitHub'ā: https://github.com/tensorflow/tensor2tensor

Modelis ir izstrādāts izmantojot sekojošas tehnoloģijas:

- Python 3.6;
- tensor2tensor 1.15;
- tensorflow 1.15;

2.2.1. Modeļu trenēšana

Kods, kur notika modeļu trenēšana ir pieejams šeit https://colab.research.google.com/drive/1LO8Dp12u61itmUrB4SMs0iJIFzAECv6Q. Modeļu trenēšana notika Colab notebook'ā. Colab tas ir serviss, kas ir izstrādāts Google kompānijā. Serviss piedāvā bezmaksas pieeju pie Google serveriem caur Jupyter notebook'iem. Savukārt, Jupyter notebook tas ir serviss, kas dod iespēju rakstīt un laist kodu tieši pārlūk programmā. Parasti ir pieejamas 2 programmēšanas valodas - Python un R, un arī Markdown, kas tika izmantots dokumentācijas un piezīmju rakstīšanai. Colab'ā ir 3 skaitļošanas paātrināšanas režīmi - CPU (vislēnākais), GPU (Pietiekami ātrs), TPU (Google izgudrojums, kura mērķis ir paātrināt matricu skaitļošanu). Darbā skaitļošanai tika izmantots GPU, jo šādā režīmā ir iespējams izmantot lokālo atmiņu vai Google Drive servisu un nav vajadzīga speciālizēta konfigurācija. Savukārt izmantojot TPU ir nepieciešama attālināta atmiņa un vajag izmantot speciālizētus servisus, tādus kā AWS S3 vai Google Storage, kas ir par maksu. Arī ir nepieciešama specializēta konfigurācija, manā gadījumā bija nepieciešams izmantot specializētu Transformer TPU modeļi un dažus citus parametrus.

Colab ir ierobežots pēc resursiem, un maksimālais darbības laiks 1 notebok'am ir 12 stundas. Tāpēc vajag savlaicīgi padomāt par darbu rezultātu saglabāšanu, lai nebūtu tā, kā viss darbs pazudis tāpēc, ka notebook pārladīsies. Darbā tika izmantots Google Drive, lai pieslēgtu atmiņu un glabātu visus rezultātus un datus tur. Google Drive piedāva 15 Gb bezmaksas atmiņas. Google Drive pieslēgšanai tika izmantots specializētais rīks google.drive.

Viss trenēšanas process tika sadalīts sekojošos posmos:

- Datu ielādēšana no Google Storage uz Google Drive, kur ir vieglāk operēt ar datiem. Savukārt no Google Storage ir vieglāk piekļūt un ielādēt datus;
- Ir definēta angļu - latviešu valodas tulošanas problēma;
- No problēmas tiek uzģenerētas datu trenēšanas kopas un vārdnīcas;
- Tiek uzkonfigurēts Transformer modelis;

- Notiek modeļu trenēšana;
- Modeļi ir iespējams izmantot teikuma tulkošanai.

2.2.2. Problēmas definēšana

«Problēma» ir termins, kas ir specifisks tieši tensor2tensor bibliotēkai. Un tā apzīmē kādu problēmu mēs gribam atrisināt un ko gribam panākt. Darbā es paplašināju tulkošanas problēmas klasi. Definējot problēmu ir nepieciešams norādīt, kur ņemt datus trenēšanai, kā tos sadalīt uz trenēšanas, testa un novērtēšanas datiem. Darbā es izmantoju sekojošo proporciju:

- 90% - trenēšanas dati;
- 5% - testēšanas dati;
- 5% - novertēšans dati.

Kā arī tulkošanas problēmai ir nepieciešams norādīt aptuvenu vārdnīcas lielumu. Vārdnīca tā ir kopa, kas var iekļaut kādu vārdu daļu pēc tokenizācijas, simbolus vai tokenus. Tokenizācija ir process, kurā vārds dalās uz daļām pēc morfoloģiskas nozīmes. Piemēram, vārds «pašnodarbinātājs» var būt sadalīts uz sekojošām daļām - «paš_», «no_», «darbināt_», «_ājs». Var būt arī citi sadalījumi. Tokenizācija notiek nejauši. Tāpēc ir nepieciešams aptuvenais vārdnīcas lielums. Tokenizācijas procesā tensor2tensor bibliotēka mēģina izveidot vardnīcu pēc iespējas tuvāk definētam ciparam. Darbā tika ekperimentēts ar dažādiem vārdnīcas lielumiem. Pašā sākumā tika trenēti nelieli modeļi ar vārdnīcas lielumu 2^13, trenēšanai uz visiem datiem tika meģināts trenēt ar lielumu 2^15, tāpat kā ir aprakstīts darbā Vaswani et al., «Attention Is All You Need», bet Colab'ā pietrūka resursu, tāpēc vārdnīcas lielumu bija nepieciešams samazināt līdz 2^14.

2.2.3. Trenēšanas process

Trenēšanas process bija sadalīts uz trenēšanu, novertēšanu un testēsanu. Viss process notika Colab notebook'ā. Tāpēc kā Colab ir makoņu platforma, tā periodiski atslēdzas, lai taupītu resursus Google serveriem. Tas ir galvenais trūkums modeļu trenēšanā, jo vajag vienmēr kontrolēt procesu un pēc vajadzības to restartēt.

Trenēšana notika sekojoši:

1. Modelis trenējas uz sagatavotiem datiem 2k iterācijas;
2. Pēc tam modelis tika saglabāts uz Google Drive;
3. Nākamais solis ir novērtēšanas posms, kas ilga 100 iterācijas;
4. Cikls turpinājās līdz 30k trenēšanas iterācijām;

5. Pēc katram 30k iterācijām nepieciešams atbrīvot vietu Google Drive;
6. Pēc vietas atbrīvošans process tiek atkārtots līdz 100k-150k iterācijām.

Gadījumā, ja process tiek pārtraukts Colab atslēgšanas dēļ, tad nepieciešams atbrīvot vietu un atkārtot procesu no pēdējās saglabāšanas vietas.

2.2.4. Modeļu pielietojums

Kad trenēšanas process tiek pabeigts ir iespējams novertēt tulkojuma rezultātus. Lai to izdarītu ir nepieciešams no definētas problēmas pielietot enkodētājus un dekodētājus, lai pārveidotu tulkojamo teikumu. Pēc tam ir nepieciešams atjautot modeli no saglabātiem svariem un padod enkodētu teikumu uz transformera ieeju. Rezultātu ir nepieciešams dekodēt un pēc tam novertēt.

Beigās ir iespējams exportēt gatavu modeļi un pielietot citā lietotnē. Modeļu eksports notiek ar t2t-export programmu.

2.2.5. Tulkojuma rezultāti dažādos posmos

Tabulā 2.1. ir apkopoti rezultāti modelei, kas ir trenēta tikai uz MS-COCO anotācijas datiem un bija daudz soļu.

Solis	Angļu teikums	Tulkojums
400000	a man is playing football	vīrietis, kas brauc ar skrituļdēli pa rampas malu.

2.1. tabula. Tulkojuma rezultati, trenējot modeli tikai uz MS-COCO datiem.

Kā redzams tulkojuma rezultāti ir ļoti slikti. No visiem vārdiem pareizi pārtulkots tikai «a man» - «vīrietis». Bet tulkojums ir latviešu valodā, ir pareizi pielietoti locījumi un uzliktas garumzīmes.

Tabulā 2.2. tiek meģināts pielāgot tulkojamo tekstu iegūtajam tulkojumam, bet tomēr rezultāts ir cits un tāpat nepareizs.

Solis	Angļu teikums	Tulkojums
400000	a dragon burned a village	vīrietis, kas sēdēja uz sola un lasīja grāmatu.
400000	a man is sitting on a branch and reading a book	vīrietis, kas stāv tenisa kortā un tur raketi.

2.2. tabula. Tulkojuma rezultati, trenējot modeli tikai uz MS-COCO datiem un pielagojot ieejas teikumu.

Tabulā 2.3. tiek meģināts tulkot tikai vienu vārdu. Kā ir redzams rezultāts ir garāks un tāpat pareizi pārtulkojas tikai daži vārdi.

Solis	Angļu teikums	Tulkojums
400000	a dog	vīrietis, kas brauc ar skrituļdēli pa rampas malu.
400000	a cat	uz galda sēž kaķis, skatīdamies uz kameru.
400000	a bird	uz ielas ir novietots autobuss.

2.3. tabula. Tulkojuma rezultati, trenējot modeli tikai uz MS-COCO datiem un tulkojot tikai vienu vārdu.

Tabulā 2.4. ir apkopoti tulkošanas rezultāti dažādas posmos. Trenēšanas gaitā rezultāti mainījās, bet palika slikti.

Solis	Angļu teikums	Tulkojums
110000	the animal didn't cross the river because it was too tired	uz ielas ir novietota liela, balta lidmašīna.
200000	the animal didn't cross the river because it was too tired	uz koka zara sēž putns.
360000	the animal didn't cross the river because it was too tired	uz ielas ir novietots autobuss.

2.4. tabula. Tulkojuma rezultati dažādas trenēšanas posmos.

Talāk tiek meģināts uztrenēt modeli, kas trenējās uz visiem datiem, bet bija mazāk soļu. Tulkojumi ir bija uz parastiem tekstiem un tie ir apkopoti tabulā 2.5.

Solis	Angļu teikums	Tulkojums
100000	two cars parked on the sidewalk on the street	uz ielas ir novietots sarkans un balts autobuss.
100000	woman stands in front of the mirror to take a picture.	vīrietis, kas stāv blakus sievietei, kas tur tenisa raketi.
100000	A woman	sieviete, kas stāv blakus vīrietim.
100000	A man	vīrietis, kas stāv blakus sievietei, kas tur rokā tenisa raketi.
100000	A cat	uz galda sēž kaķis ar ķepām uz tā.

2.5. tabula. Tulkojuma rezultati uz visiem datiem ar parastiem tekstiem.

Kā redzams rezultāti ir joprojām slikti, bet ievērojami labāki nekā bija iepriekš. Piemēram, teikumā par autobusu, konteksts ir pareizs, bet daži vārdi ir nepareizi.

Dati parsvara ir politiski un juridiski virzīti, tad modelei jaspēj labāk pārtulkot tadus datus. Tabulā 2.6. ir apkopoti dati par politiski un juridiski virzītiem tekstiem.

Solis	Angļu teikums	Tulkojums
100000	commission may decide to refer them to the Court of Justice.	Komiteja var nolemt vērsties pret viņiem Tiesā.
100000	a police	policija
100000	Transport is fundamental to our economy and society.	Transports ir būtisks mūsu ekonomikai un sabiedrībai.

2.6. tabula. Tulkojuma rezultati uz visiem datiem ar parastiem tekstiem.

Šajā gadījumā rezultāti ir diezgan precīzi gan īsam teikumam, gan garam.

2.2.6. Sastaptas problēmas

Trenēšanas process notika pārsvarā uz GPU procesoriem. Lai izmantotu TPU režīmu vajadzēja glābāt visus failus attālināti Google Storage servisā. Savukārt izmantojot GPU vai CPU ir iespējams glabāt datus un rezultātus Colab notebook'ā. Tas ir saistīts ar to, kā TPU tas ir attālināts servers, kas ir domāts tikai matricu skaitļošanai un optimizēts tikai tam. Lai izmantotu Google Storage par to ir nepieciešams maksāt. Šo faktoru ir nepieciešams ievērot, ja grib trenēt uz TPU procesoriem.

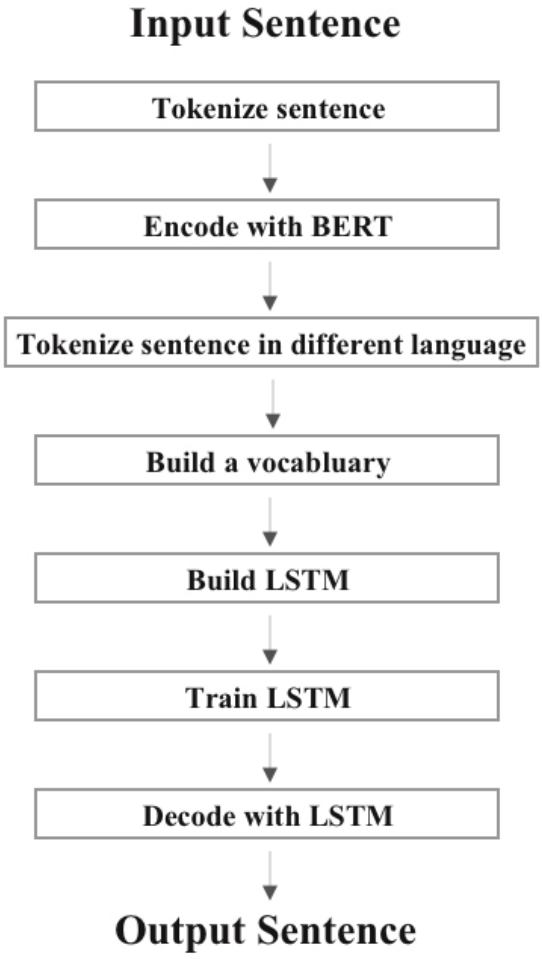
Galvenais trūkums bija Colab sesijas garums un savienojuma sabrukums. Colab nav domāts laiku un resursu ietilpīgiem darbiem. Tas vairāk ir domāts pētnieciskam darbam. Ja ir nepieciešams ilgi trenēt modeli vai tas būs resursu ietilpīgs, tad ir nepieciešams izmantot speciāli būvētus servisus. Piemēram, AWS SageMaker, Google Compute Engine modeļu trenēšanai in AWS S3 vai Google Storage datu glabāšanai.

Paša modeļa trūkums ir datu vienmērīgums. Tie ir politiski virzīti un līdz ar to slikti tulko ikdienišķos tekstus.

2.3. BERT modeļa pielietojums tulkošanas uzdevumam

Nākamais uzdevums ir izveidot tulkotāju uz BERT bāzes. Darbā nesanāca atrast citus rakstus, kur bija minēts kā tika izmantots BERT, lai tulkotu teikumus. Parasti to izmanto, lai klasificētu teikumu, izveidotu teikuma sentimenta analīzi vai uzbūvētu jautājumu atbildēšanas

modeli. Bet bija cerība, ka no BERT būs iespējams uzbūvēt pietiekami precīzu valodas modeli. Līdz ar to, tas ir vairāk pētījums vai eksperiments, kura mērķis ir noskaidrot cik precīzus valodu modeļus ir iespējams iegūt no BERT vektoriem. Tas darbs ir iedvesmots ar «Image captioning with visual attention» darbu https://www.tensorflow.org/tutorials/text/image_captioning. Šeit arī tiek izmantoti MS-COCO dati un ideja ir līdzīga, vienīgais BERT vietā tiek izmantots InceptionV3 modelis. Šis modelis spēj pārveidot attēlu uz vektoru. Modelis pievērš uzmanību objektiem nevis vienkārši parvērš attēlu vektoru formā izmantojot kādu no parastajam konvolūcijas vai saspiešanas metodēm.



2.1. attēls. Trenēšanas process izmantojot BERT.

Līdz ar to, tāpēc, ka idejas balstās uz esošām tehnoloģijām šo uzdevumu ir iespējams sadalīt uz divām daļām: BERT daļa un LSTM daļa. Pirmā daļa ir teikuma vektoru iegūšana un tiks veikti vairāki ekperimenti, lai noskaidrotu veidu kā labāk izveidot teikuma vektoru. Šī daļa vairāk mainīsies. Otrā daļa balstās uz esošo un strādājošo pieeju un mazāk mainīsies.

Otrā daļa tikai pielāgosies BERT vektoram un tekstiem, nevis attēliem, bet kopumā šī daļa paliks tāda paša.

Viss uzdevums ir veikts Colab vidē izmantojot python kā programmēšanas valodu un tensorflow kā mašīnmacīšanas bibliotēku.

Trenēšanas procesu ir iespējams apskatīt attēlā 2.1. Pirms trenēšanas ir nepieciešams sagatavot teikumus trenēšanai, tā kā tas jau bija izpildīts 2.1. nodaļā un tika izmantoti tādi paši dati, kas neatspoguļojas shēmā.

1. Sākumā katru teikumu ir nepieciešams tokenizēt. Rakstā (Devlin et al, 2019) ir norādīts, ka BERT saprot teikumus specifiskā formātā. Katru teikumu ir nepieciešams sadalīt uz vārdu daļām jeb tokeniem. Darbā ir veikti dažādi eksperimenti, līdz ar to mainījas arī tokenizācijas veidi.
2. Otrais solis ir iegūt teikuma vektorus ar BERT. Lai to izdarītu ir iespējams izmantot dažādas stratēģijas. Bet jebkurā gadījumā ir nepieciešams izmantot BERT modeli. Tas tika instalēts no oficiālā repozitorija https://github.com/google-research/bert. Instalēt ir iespējams uzreiz no GitHub izmantojot «git» versijas menedžmentu sistēmu vai arī «pip» python paketu menedžeri. Darbā tika izmantots «pip», jo tas ir ātrākais un vieglākais veids kā pieinstalēt bibliotēkas un paketes iekš Colab.
3. Nākamais solis ir tokenizēt visus teikumus citā valodā. Tā kā es tikai ekperimentēju, tika izmantota tāda pati valoda - angļu. Tas ir izdarīts, lai vienkāršotu novērtēšanas procesu. Un sanāk, ka šajā gadījumā uzdevums ir nevis pārtulkot teikumu citā valodā, bet restaurēt to pašu teikumu izmantojot tikai teikuma vektoru, kas ir enkodēts ar BERT modeli. Tā mēs varam arī novertēt cik daudz informācijas ir iekodētas vienā vektorā un cik precīzi tie vektori atrodas vektoru plaknē. Tokenizācija šajā solī atšķiras no iepriekšējas tokenizācijas. Šī tokenizācija vienkārši pārveido teikuma vārdus uz ciparіem un nesadala vārdus uz daļām.
4. Vārdnīcu būvēšana notiek šādi - ņemam visus vārdus, izvēlamies no tiem «n» populārākus un ievietojam vārdnīcā formātā «cipars» - «vārds». Vārdi kas mazāk ir sastopami tiek atzīmēti kā nezināmie. Tas ir izdarīts, lai pēc iespējas pievērstu vairāk uzmanības vārdiem, kas tiek sastapti biežāk.
5. LSTM slānis sastāv no enkodera un dekodera. Enkoders ir ļoti vienkārš. Tas ir pilnīgi saistīts (fully-connected) slānis un ReLu aktivācijas slānis. Dekoders nav tik vienkāršs. Viņš sastāv no «Embedding», «GRU», «Bahdanau Attention» un «Fully-

connected» slāniem. «Embedding» un «Fully-connected» slāņa ir tokenu reprezentācijas modelī. «GRU» un «Attention» slānis palīdz «pieminēt» iepriekšējo kontekstu.

6. Trenēšanas process notiek sekojoši:
    1. Enkodēt teikumu patiju (batch);
    2. Dekodēt tos pašus teikumus;
    3. Salīdzināt dekodētu rezultātu ar vektoru no BERT modeļa. Salīdzināšanai tika izmantota standarta «Sparse categorical Crossentropy» loss-funkcija;
    4. Vektoru optimizācija izmantojot «Adam» algoritmu
7. Beigās ir iespējams novertēt uztrenētu modeli. Novertēšana notiek līdzīgi kā trenēšana:
    1. Enkodējam teikumu;
    2. Dekodējam teikumu;
    3. Dekoders atgriež varbūtības vektoru. Katram vārdam ir sava varbūtība. Nejauši izvēlamies vārdu, ņemot verā šis varbūtības. Pievienojam vārdu teikumam;
    4. Atkārtojam 2.-3. punktus kamēr speciālais tokens «<end>» nav sasniegts vai ir sasniegts maksimālais teikuma garums.

Tā ir uzdevuma vispārējā struktūra. Darba gaitā tika izmēģināti dažādi paņēmieni un metodes. Rezultāti un kods mainījās, bet galvenas idejas palika tādas pašas.

2.3.1. Datu sagatavošana.

Šajā uzdevumā tika izmantoti tie paši dati, kas tika izmantoti pirmajā eksperimentā - MS-COCO anotācijas teksti. Tie ir jau apstradāti un gatavi trenēšanai. Dati ir saglabāti «Google Storage» servisā saspiesta formatā un pirms trenēšanas ir nepieciešams tos lejupielādēt un sagatavot. Šim nolūkam tika izmantotas «gcloud» un «gsutil» komandas, kas ir peejamas Colab'ā. Autorizācija notiek izmantojot «gcloud» utilitu un tā notiek izmantojot OAuth2 protokolu. Lai varētu autorizēties ir nepieciešams izsaukt komandu «!gcloud auth login». Pēc tam vajag atvērt saiti, autorizēties Google servisā, atļaut Colab'am izmantot «Google Storage» servisu un dabūt piekļuves tokenam (access token). Šo tokenu vajag ielikt formā, kas parādās pēc komandas izsaukšanas un, tad var lejupielādēt datus no «Google Storage». Lai tos lejupielādētu tika izmantota komandu «!gsutil -m cp gs://<faila saite Google storage servisā> <lokalais ceļš uz failu>». Komandas sintakse ir līdzīga standartai UNIX komandai «cp». Kad fails ir lejupielādēts ir nepieciešams to noarhivēt, jo tas ir saspiests ZIP

formatā, lai samazinātu glabāšanas un transportēšanas izmaksas. Šim nolūkam tika izmantota «zipfile» bibliotēka.

Talāk datus vajag sagatavot šim uzdevumam. Tā kā petot BERT Vektora kvalitāti nav nepieciešami daudz dati darbā nav izmantoti visu datu korpusu - 600 000 teikumi, bet tikai 30 000. Tas ir pietiekami, lai uztrēnētu modeli un novērtētu vektora kvalitāti padodot uz modeļa ieeju, kādu teikumu no tiem pašiem anotācijas datiem, bet tikai izmantojot teikumus, kas nepiedalās trenēšanā. Tad pirmais solis ir samazināt datu apjomu. Tas ir izdarīts izmantojot speciālo python sintaksisu - [:30000]. Pēc tam mēs izejam cauri visam palikušam teikumam, apgriežam atstarpes teikumā sākumā un beigās un pievienojam speciālus tokenus «<start>» un «<end>». Šie tokeni tiks izmantoti turpmāk dekodēšanas laikā, lai noskaidrotu teikuma sākumu un beigas.

Šis operācijas atkārtojas katrā eksperimentā un turpmāk tiek pieņemts, ka visi aprakstītie soli jau ir izieti un dati ir gatavi.

2.3.2. «CLS» tokena izmantošana.

Šī uzdevuma galvenais mērķis ir iegūt pēc iespējas kvalitātīvāko teikuma vektoru no BERT modeļa. Ir dažādas pieejas kā to var panākt. Viena no tām ir izmantot speciālo CLS tokenu, kas ir iegūts pēc tokenizācijas. Šī ir standarta pieeja un to visbiežāk izmanto vienkāršas klasifikācijas uzdevumos, lai iegūtu teikuma reprezentāciju. Trenēšanas arhitektūra ir parādīta 2.2. attēlā.

Turpmāk, citās teikuma vektora iegūšanas pieejās, tiks parādītas tikai augšējas daļas, jo tās vairāk mainīsies. Apakšā - «Decoder» daļa paliks tāda pati. Šī ekperimenta pirmskodu var atrast atvērot šo saiti:

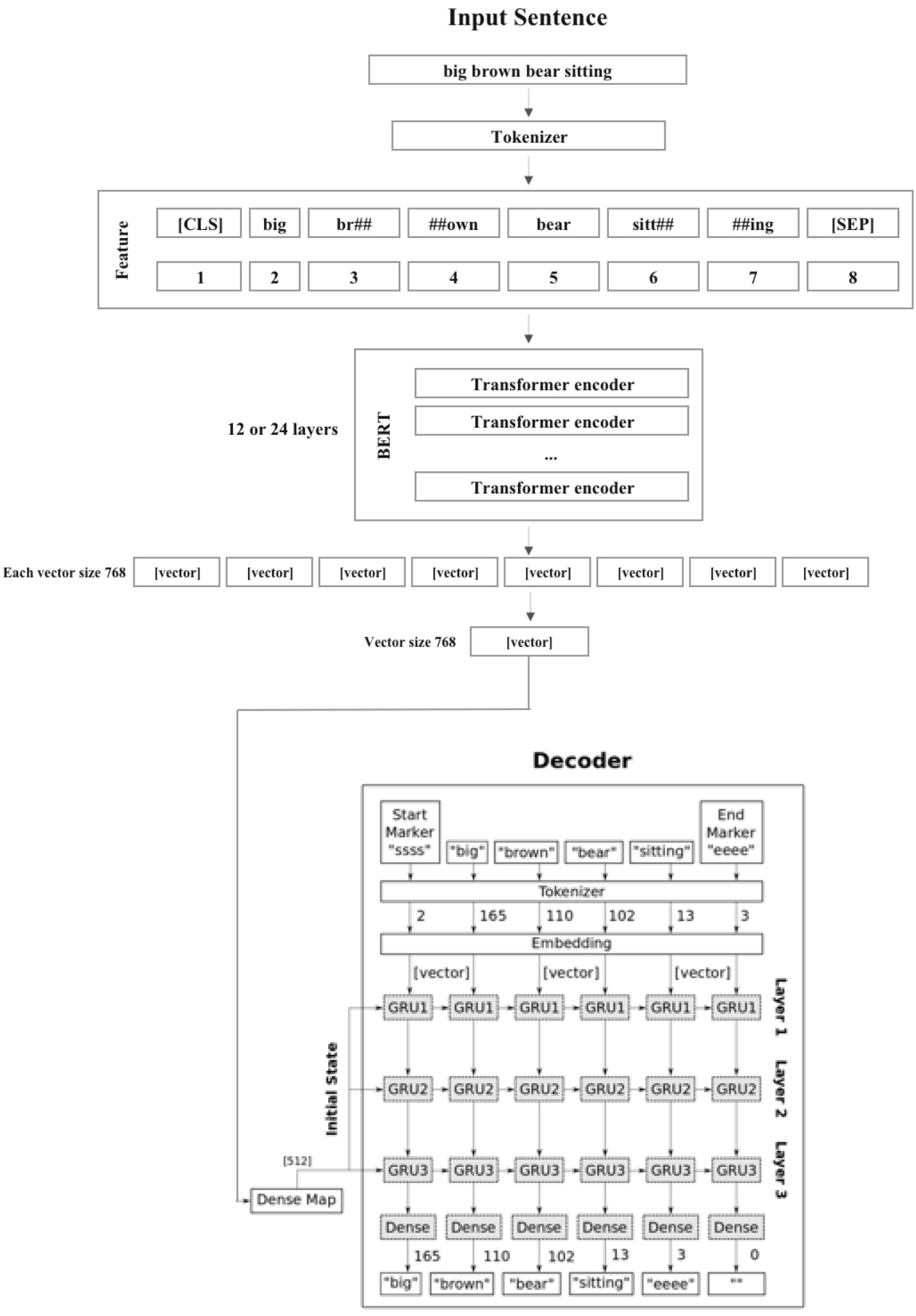
https://colab.research.google.com/drive/1gBDm4tqXeypf6nPKX1E7rnnf7GRKTwPL

Lai izveidotu modeli, kas balstās uz BERT pamatu ir nepieciešamas sekojošas lietas:

1. Pats BERT modelis;
2. «bert-tensorflow» pakete.

Google piedava sekojošas BERT modeļa variācijas:

1. BERT-Large, Uncased (Whole Word Masking);
2. BERT-Large, Cased (Whole Word Masking);
3. BERT-Base, Uncased;
4. BERT-Large, Uncased;
5. BERT-Base, Cased;



2.2. attēls. Trenēšanas arhitektūra izmantojot BERT.

6. BERT-Large, Cased;
7. BERT-Base, Multilingual Cased;
8. BERT-Base, Multilingual Uncased;
9. BERT-Base, Chinese.

Visi modeli atšķiras pēc slāņu un parametru skaita, kā arī pēc tā uz kādiem datiem modelis tiek trenēts. «Uncased» nozīmē to, ka trenēšanā izmantoti visi vārdi ir ar maziem burtiem. Tas ir izdarīts, lai vienkāršotu modeli un palielinātu precizitāti. Piemēram, teikumam «Bērns staigā pagalmā» un «bērns staigā pagalmā» ir viena un tāda pati nozīme. Atšķiras tikai gramatiskie likumi. Līdz ar to sākumā rekomendēja izmantot «BERT-Base, Multilingual Uncased» modeli trenējot savu modeli uz tekstiem citās valodās. Bet nozīme var atķirties dažas valodas, kur ir svārīgi ar kuru burtu sākas vārds. Piemēram, vācu valodā visi lietotājvārdi sākas ar lielo burtu. Tāpēc vēlāk Google piedāvāja jaunu modeli - «BERT-Base, Multilingual Cased», kas ir rekomendējama un trenējot modeli uz latviešu valodas tekstiem, darbā tas arī tika izmantots. Runājot par modeļa izmēru ir divas opcijas - «BERT-Base» un «BERT-Large». Bāzes modelim ir 12 slāņi un kopumā 110 000 parametri, lielam modelim ir 24 slāņi un 340 000 parametri. Tā kā trenēša notiek Colab vidē man nepietiks resursu, lai izmantotu lielo modeli. Trenēšanā izmantojot angļu-latviešu datus tika izmantots «BERT-Base, Multilingual Cased» modelis un angļu-angļu datiem - «BERT-Base, Uncased».

Pirmā pieejā pieņemot CLS tokenu par teikuma reprezentāciju balstās uz oficiāla pirmskoda no «optimization», «tokenization», «modeling», «extract_features» paketem.

1. Sākumā teikumus nepieciešams tokenizēt. Mēs izvēlamies «FullTokenizer». Tas nozīmē, kā teikuma vārdi būs sadalīti uz vārdu daļām, kā arī dati tiks papildus notīrīti. Vārdu sadalīšana notiek nejauši, tas nozīmē, ka vārds var būt sagriezts jebkurā vietā;
2. Tālāk teikumi ir pārveidoti par «piemēru» objektiem. Tas ir nepieciešams, lai dabūtu teikumus formātā, kas spēj saprast BERT;
3. «bert-tensorflow» paķetē ir funkcija «convert_examples_to_feature», kas pieņem «piemērus» un atgriež «features», kurus saprot BERT. «Feature» iekļauj sevī tokenizēto teikumu, tokenu identifikatorus un teikuma masku (kādi ieejas baiti ir aizņemti un kādi nē);
4. Modeļu būvēšanā notiek izmantojot «model_fn_builder» funkciju no «bert-tensorflow» paķetes. Kā arguments, tika padots izvēlēta BERT modeļa konfigurācija, kontrolpunkts, ja trenēšana notiek ne no jauna, kādus slāņus ir nepieciešams ņemt verā, es paņēmu pēdējos 4 slāņus;

5. Trenēšanas funkcija arī ir standarta un paņemta no tās pašas paķetes.

6. Colab vidē ir iespējams trenēt modeli uz TPU. Tās ir specializētas rēķināšanas vienības, kas ir izstradātas Google un to galvenais mērķis ir paātrināt rēķināšanu mašīnmacīšanas trenēšanas laikā.

Kad trenēšana ir pabeigta, «word_embeddings» mainīgā ir saglabāti tokenu vektori. Vektora lielums ir 768. Pirmajā pieejā ņem tikai «[CLS]» tokenu vektoru un pieņem to, kā visa teikuma vektoru. Tajā pašā vietā ir iespējams paņemt vidējo vertību no visiem teikuma vektoriem. Tā ir cita standarta pieeja. Rezultāti gan «[CLS]» tokenam, gan vidējam vektoram ir līdzīgi, tāpēc turpmāk tas netiks apskatīts atsevišķi.

Otra daļa balstās uz «Image captioning» rakstu.

1. Sākumā notiek atkārtota teikuma tokenizācija. Šī tokenizācija atšķiras no tokenizācijas, kas bija pirmajā posmā. Šeit visi vārdi aizvietoti ar kādu numuru, kas identifice šo vārdu. Vārdi netiek dalīti uz daļām.

2. Tālāk tiek izveidota valodas vardnīca, kur katram vārdam ir piesaistīts numurs un iespējams iegūt vārdu pēc numura un numuru pēc vārda;

3. Nākamais solis ir sadalīt datus uz trenēšanas un testa datiem. Darbā tika izmantota 80-20 proporcija. No testa datiem tiek ņemti teikumi novērtēšanas posmā un tie šādi nepiedalās trenēšanā;

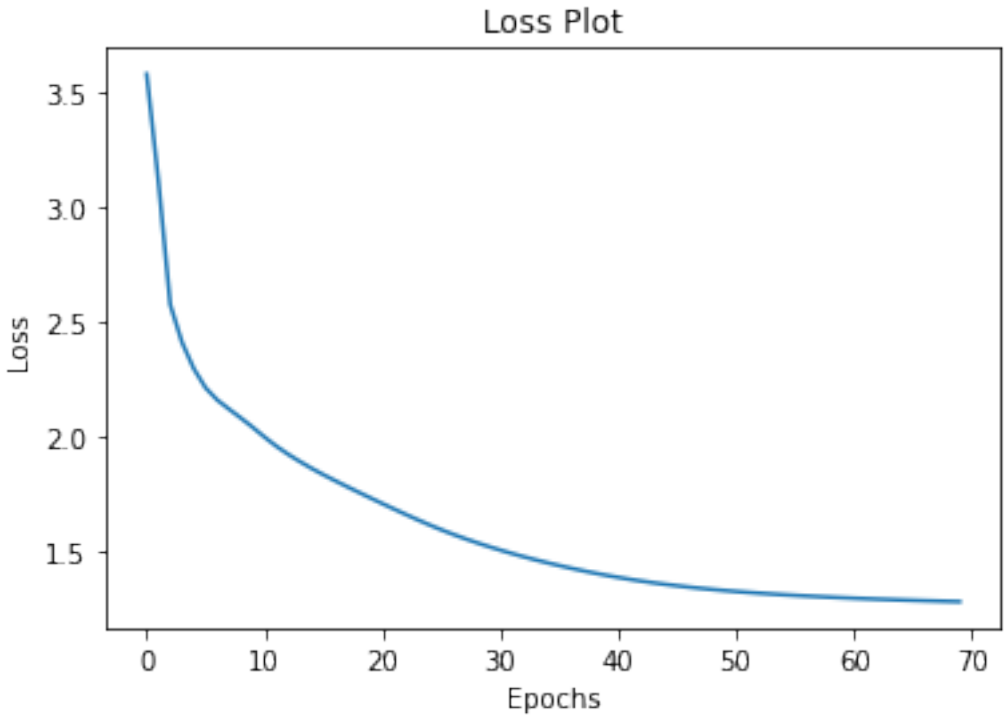
4. Lai trenēšana notiktu efektīvāk un ātrāk, datus nepieciešams sadalīt uz porcijām (batches). Šim uzdevumam tika izvēlēti 64 teikumi vienā trenēšanas solī. Pēc dažiem testiem, šis lielums bija pietiekami efektīvs, pietiek RAM atmiņas un process ir bijis pietiekami ātrs.

5. Šajā uzdevumā tika izmants LSTM modelis, lai atkodētu BERT vektorus vienkāršajā teikumā. Līdz ar to, tika definēts enkoders un dekoders, kas bija aprakstīts 2.3. nodaļā;

6. Tālāk notiek trenēšana.Trenēšanas laikā modeļu kontrolpunkti saglabājās Google Drive. Tas ir izdarīts, lai būtu iespējams turpināt trenēt modeli gadījumā, ja savienojums ar Colab sabruks, kas ir ļoti raksturīgs šai videi;

7. Trenēšanas laikā loss-funkcijas vertības saglabājās atsevišķā mainīgajā. Un to grafiku ir iespējams apskatīt 2.3. attēlā;

8. Beigās notiek modeļu novērtēšana. Novērtēšana cik labi vai slikti ir atkodēts BERT vektors notiek manuāli un vērtējums ir diezgan subjektīvs.



2.3. attēls. Loss-funkcijas vertības izmantojot [CLS] tokenu.

Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.7.

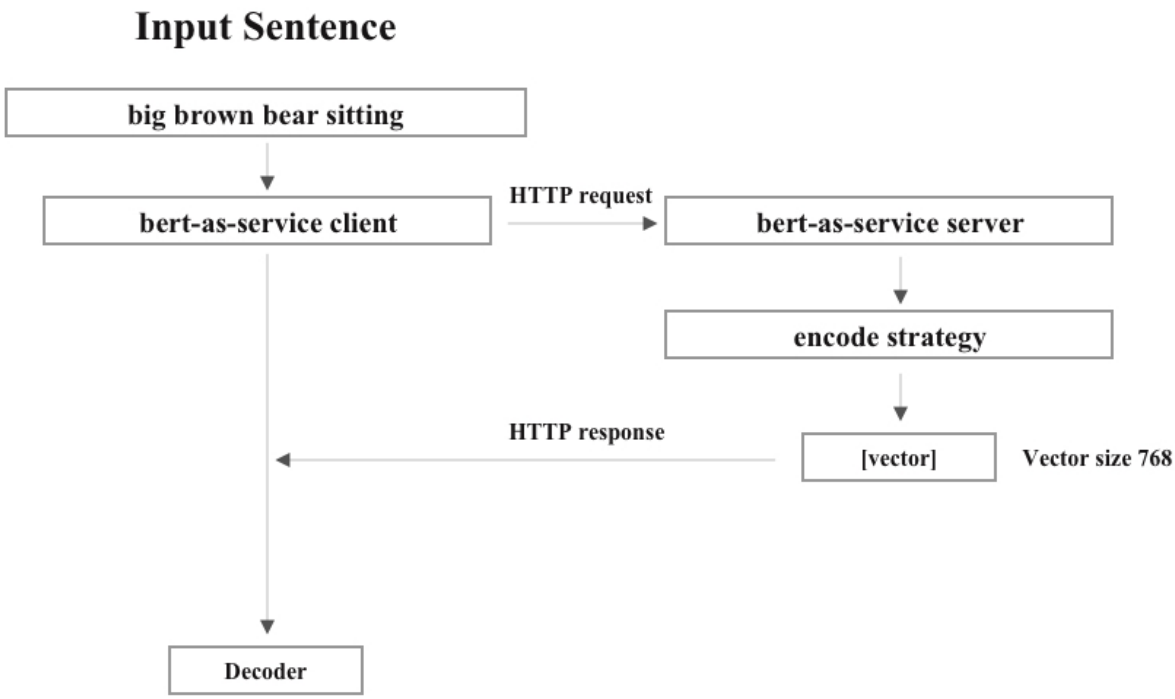
Reals teikums	Atkodēts teikums
a group of people preparing food in a kitchen	a car parked on the side car
a kitchen table with a juicer and vegetable on it	large plane in the back of any children wearing helmets standing next to a living room covered in the back
half of a white cake with coconuts on top	a oneworld passenger plane that cooking in a scene with bulbous lights

2.7. tabula. Tulkojuma rezultati trenējot modeli izmantojot [CLS] tokena vektoru.

Kā redzams, atkodēšana ir pilnīgi nepareiza, bet vismaz vārdi ir saistīti un nav pilnīgi nejauši. Tas nozīmē, ka šāds valodu modelis eksistē, bet nav precīzs.

2.3.3. «bert-as-service» izmantošana tokenu iegūšanai.

Kopumā šis ekperiments atkārto iepriekšējo līdz ar to tiks pieminti tikai fakti, kas atšķiras šajā metodē. Eksperimenta struktūra ir parādīta attēlā 2.4.



2.4. attēls. BERT Vektora iegūšana izmantojot «bert-as-service».

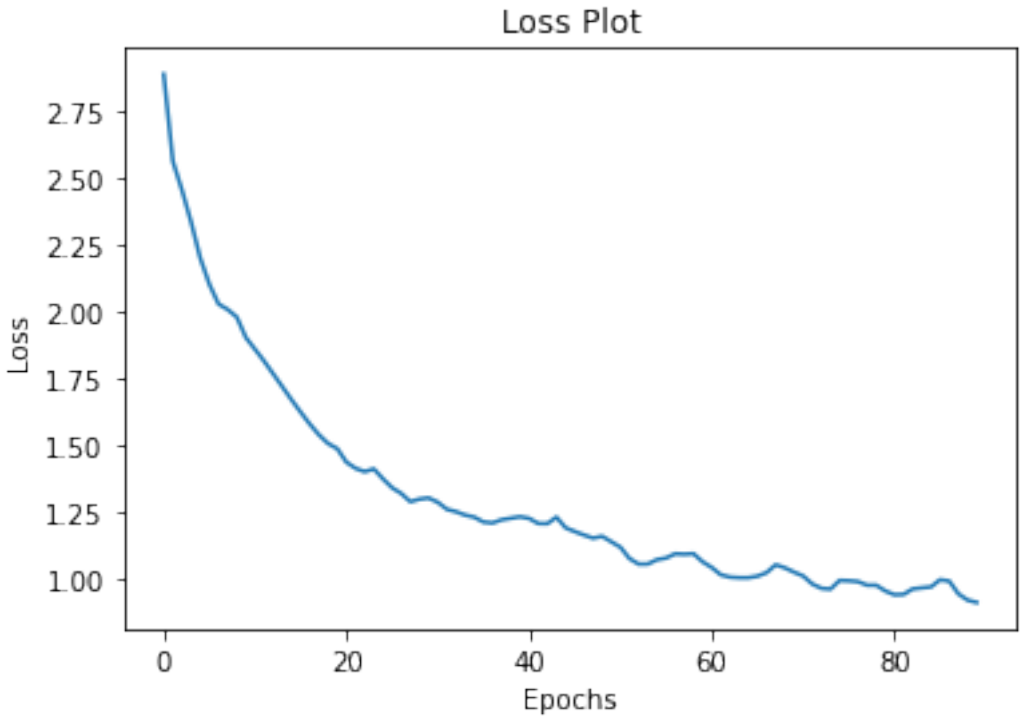
«Bert-as-service» tas ir atsevišķs serviss, kas specializējās uz BERT un to vektoru iegūšanu. Tas strādā pēc vienkāršas klients-servers arhitektūras. Sākumā ir nepieciešams palaist serveri un padod informāciju, kādu BERT modeli vēlas izmantot. Pirms tā, modeli ir nepieciešams lejupielādēt atsevišķa direktorijā. Ir iespējams izmantot oficiālus avotus, gan precizētus (fine-tuned) modeļus. Servera palaišana notiek ar komandu «bert-serving-start -num_worker=1 -model_dir=<BERT modeļa direktorija>». Arī ir iespējams izvēlēties enkodēšanas stratēģiju. Pēc noklusējumā, šis servers ņem pēdējos slāņus un apvieno visus vektorus vienā ņemot vidējo no visiem vektoriem. Tālāk, lai iegūtu vektorus ir nepieciešams izveidod klientu. Tā mērķis ir padod visus teikumus, kas tiks izmantoti trenēšanas un testēšanas posmos uz serveri, saglabāt vektorus lokāli. Klienta pirmskods ir sekojošs:

```
from bert_serving.client import BertClient
bc = BertClient()

with open('sentences.txt', 'r') as source:
    sentences = source.readlines()
    embeddings = bc.encode(sentences)
    with open('embeddings.txt', 'w') as destination:
        destination.writelines(embeddings)
```

Velāk saglabātie vektori lejupieladējas uz Google Drive no kurienes tie tiek lasīti Colab vidē. Turpmāk trenēšana atkārtojas tāpat kā pirmajā eksperimentā. Loss-funkcijas vertības ir apkopotas 2.5. attēlā. Visa eksperimenta kods ir pieejams šajā saitē:

https://colab.research.google.com/drive/1URtg9VKjRN3WZC4-ulPDnEoeU1bIDIPr



2.5. attēls. Loss-funkcijas vertības izmantojot «bert-as-service» tokenu iegūšanai.

Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.8.

Reals teikums	Atkodēts teikums
a dirt bike rider doing a stunt jump in the air	motorcyclist riding motorcycles and children wearing a herd of three computer and woman is a group of a man backing up against a man
a large passenger jet taking off from an airport	commercial airplane parked illegally alone on top of people sitting next to be group of people walking snow next to an airplane themed bathroom

Reals teikums	Atkodēts teikums
a man on a bike pulling a dog in a cart	people sitting horses standing intersection taken inside an older man hang off top of toilet station alone sitting illegally alone laying on motorcycles and
a picture of a cat door is on the bottom corner of a blue door	group of wood surface
a woman is taking a picture of herself in a bathroom mirror	three drivers and white bathroom containing white airplane sitting alone standing near multi story buildings

2.8. tabula. Tulkojuma rezultati trenējot modeli izmantojot «bert-as-service» teikuma vektoru.

Kā redzams, rezulāti joprojām nav korekti, bet vismaz daži vārdi ir pareizi atkodēti, tas ir labāk nekā iepriekšējā ekperimentā. Tas nozīmē, ka ņemt pēdējos 2 slāņus un ņemt to vidējo vērtību ir efektīvāk un valodu modelis ir precīzāks, bet kopumā tas paliek slikts un tulkojuma rezultāti ir daudz sliktāki, neka vienkāršais «sequence-to-sequence» algoritms.

2.3.4. Visa teikuma tokenizācija BERT vektoru iegūšanai.

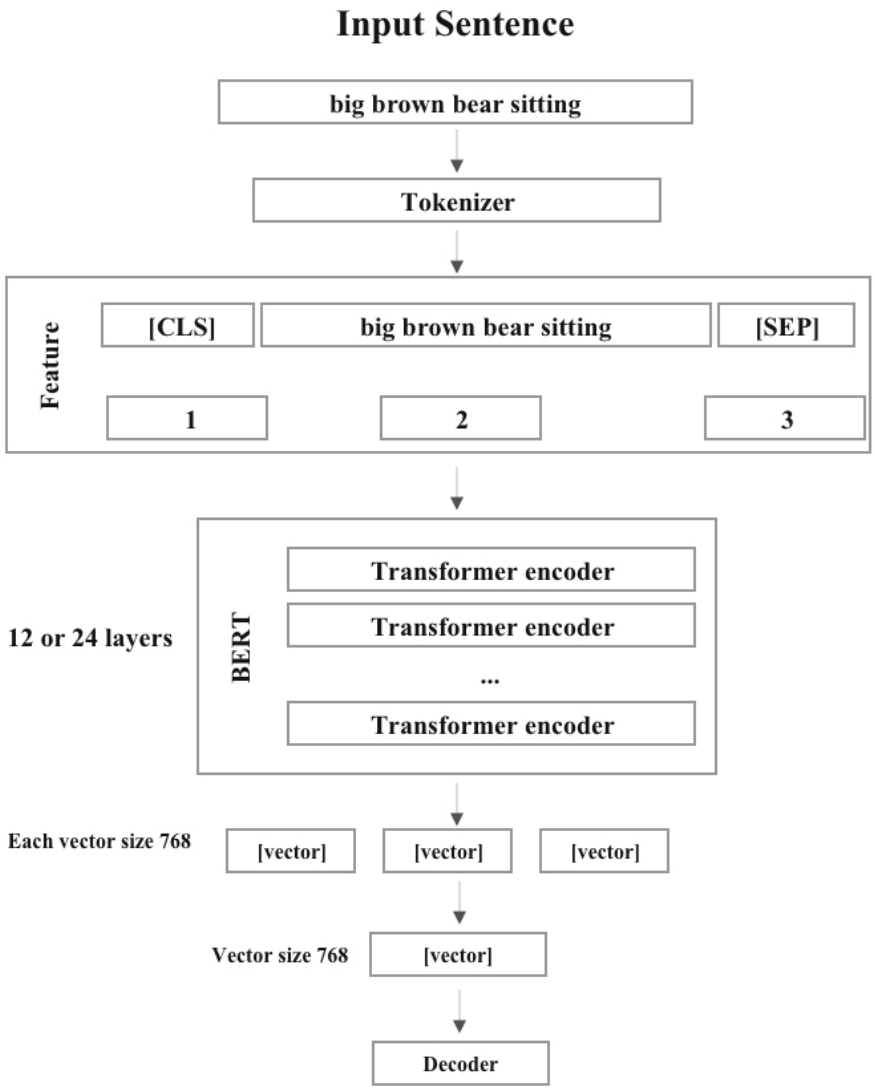
Nākoša pieeja ir mēģinājums enkodēt visu teikumu, nevis atsevišķus tokenus. Šis pieejas struktūra ir parādīta 2.6 attēlā. Šī pieeja ir līdzīga pieejai ar «CLS» tokeniem, bet tikai tiek ņemts cits vektors.

Šī ekperimenta kods ir pieejams šajā saitē:

https://colab.research.google.com/drive/169JAuInGRIuZlpREqlL5Q6VQ2B-UVY2F

Lai tokenizētu visu teikumu ir nepieciešams speciālais tokenizators. Darbā tās tika nosaukts «FullSentenceTokenizer». Šis tokenizators strādā sekojoši:

1. Teikums tiek notīrīts no speciāliem simboliem, liekām atstarpēm un tiek konvertēts uz «UTF-8» enkodingu;
2. Teikumam ir piesaistīts unikālais identifikators;
3. Teikums tiek pievienots vārdnīcai, kas saglabājās iekšējā mainīgā;
4. Vārdnīcā ir iespējams meklēt teikumus pēc identifikatora un identifikatoru pēc teikuma.

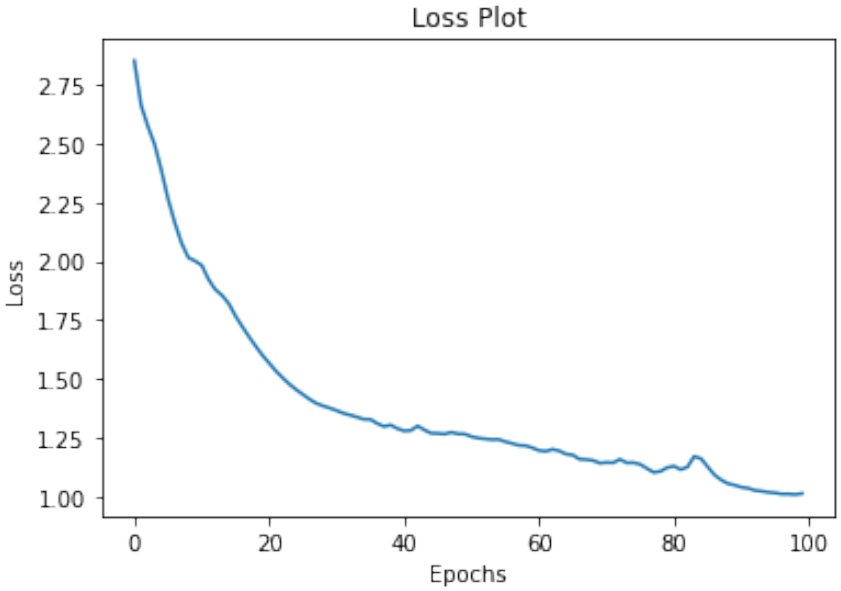


2.6. attēls. BERT Vektora iegūšana izmantojot visu teikuma tokenizāciju.

Tālāk trenēšana notiek tāpat kā iepriekšējos ekperimentos. Loss-funkcijas vērtības ir apkopotas 2.7. attēlā. Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.9.

Reals teikums	Atkodēts teikums
a car show features a yellow hot wheels car	1 military propeller pilot
two woman loading bikes onto a public bus	a scratching pad
a cat standing on the toilet bowl seat	a television show
a pinewood and green modern themed kitchen area	well as it

2.9. tabula. Tulkojuma rezultati trenējot modeli izmantojot pilno teikumu vektoru.



2.7. attēls. Loss-funkcijas vērtības izmantojot visu teikuma tokenizāciju.

Kā redzams rezultāti ir vēl sliktāki, nekā izmantojot «CLS» tokenu. Un teikumi kļuva īsāki. Varbūt, ka šādā vektorā saglabājās mazāk informācijas par sākotnējo teikumu.

2.3.5. SBERT izmantošana teikumu vektoru iegūšanā.

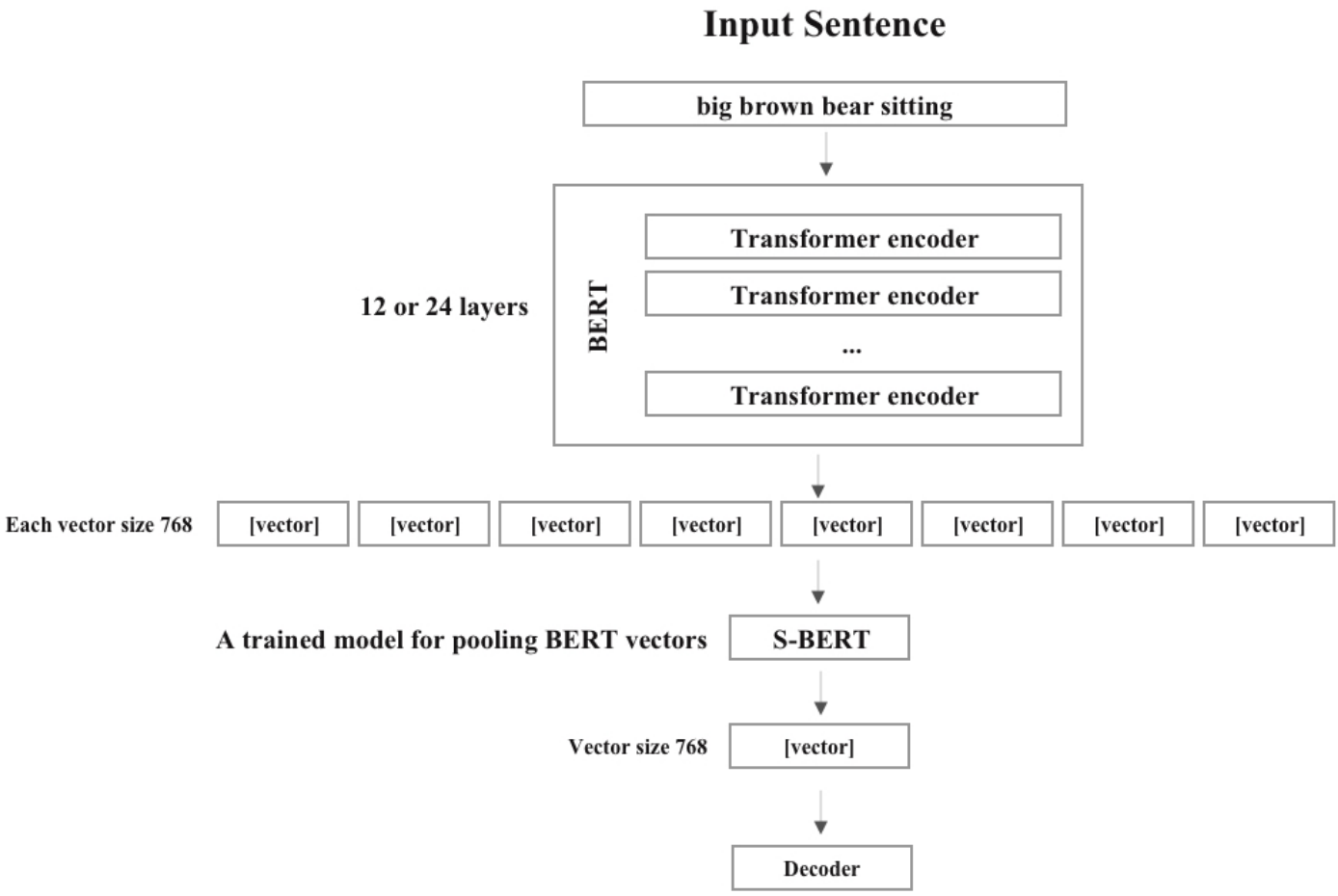
Nākošā pieeja tika izmantots SBERT. Šis modelis tiek specializēts uz teikumu vektoriem un ar to palīdzību ir iespējams iegūt kvalitatīvākus teikuma vektorus. Modelis balstās uz BERT, bet tas tiek precizēts izmantojot trīs stratēģijas. Katra strateģija ģenerē savu vektoru un pēc tam tās tiek apvienotas vienā. Šis pieejas struktūra ir parādīta 2.8. attēlā. Kā redzams, pieeja ir līdzīga BERT CLS tokena iegūšanai. Atšķiras tikai pēdējais solis, kur tiek iegūts jauns vektors.

Šis pieejas pirmskods ir pieejams šeit:

https://colab.research.google.com/drive/1y8B19482mJgT9zr6rdxgKWC-XP_fFl5X

Tā kā šī pieeja ir viena no labākām teikuma vektora iegūšanai, tika meģināts uztrenēt modeli vektoru dekodēšanai gan uz ierobežota datu lieluma, gan uz visiem datiem. Tāpēc, ka trenēšana uz visiem datiem ir resursu ietilpīga, savienojums and Colab servicu bieži sabruka un vajadzēja sākt no jauna vai no pēdēja kontrolpunkta. Tāpēc nesanāca iegūt loss-funkcijas grafiku. Loss-funkcijas grafiks nelielam datu apjomam ir redzams 2.9. attēlā. Jāpiemin, ka lai uztrenētu modeli uz visiem datiem, bija nepieciešams glabāt jau iekodētus teikuma vektorus atsevišķi uz Google Drive, lai nevajadzētu enkodēt tos no jauna katru reizi. Kopumā MS-COCO ir gandrīz 600 000 teikumi. Lai tos veiksmīgi iekodētu bija nepieciešams sadalīt visu datu korpusu uz porcijām - 100 000 teikumi katrā porcijā. Vienas porcijas enkodēšana

aizņema aptuveni 3-4 stundas. Pēc tam vajadzēja trenēt modeli. Colab resursi ir ierobežoti līdz 32GB RAM. Tas ir diezgan daudz, bet tomēr tas pietrūka, lai trenētu vairāk, nekā vienu epohu vienas sessijas garumā. Vienas epohas trenēšana arī aizņēma 3-4 stundas.



2.8. attēls. Teikuma vektora iegūšana izmantojot SBERT.

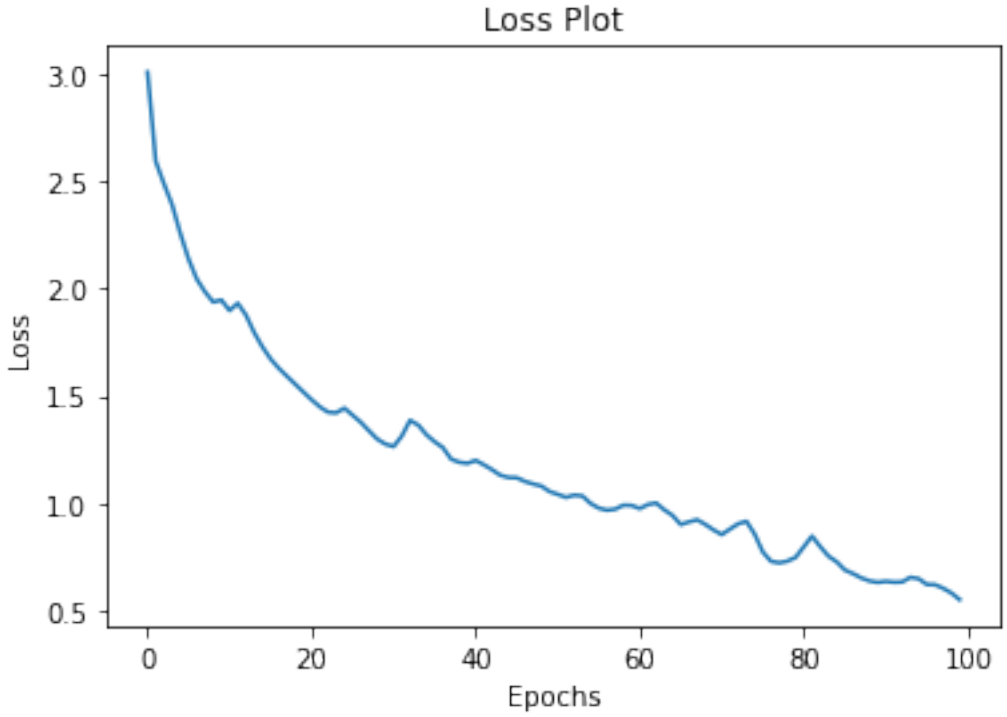
Loss-funkcijas vertības mazam datu korpusam ir apkopotas 2.9. attēlā.

Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.10.

Reals teikums	Atkodēts teikums
three motorcyclists and their bikes stand on the pavement.	a motorcycle parked in a mountain for a <unk> in a <unk> in a woman in a light in a <unk> in a bicycle.
two cars parked on the sidewalk on the street.	the sky sitting next to an empty mirror

Reals teikums	Atkodēts teikums
several motorcycles riding down the road in formation.	people walking in shelves in a sink with marble mirror over a small bathroom with bulbous lights peers through a toilet next to steal
light shining onto the pews of a church through an open door.	stone wall motorcycle sitting next to take an office river <unk> sitting next to <unk> dishes
a man and woman leaning against a motorcycle.	people on tip of some yellow <unk> cat is carrying a man sits on a bright room with marble countertops and her bathroom sink

2.10. tabula. Tulkojuma rezultati trenējot modeli izmantojot S-BERT teikumu vektoru.



2.9. attēls. Loss-funkcijas vertības izmantojot S-BERT teikumu vektorus.

Kā redzams, rezultāti ir līdzīgi «bert-as-service» rezultātiem, bet tomēr pareizi pārtultoti ir tikai daži vārdi. Atkodētam teikumam nav jēgas un bieži nav skaidrs konteksts.

2.3.6. Visu tokenu izmantošana tulkošanas uzdevumam. BERT modelis.

Teikuma vektora izmantošana tulkošanas uzdevumam nesasniedza augstus rezultātus. Es pieņēmu, ka teikuma vektors nesatur pietiekami daudz informācijas, lai būtu iespējams veiksmīgi tulkot. Līdz ar to nākošajā eksperimentā ir meģināts piestrādāt modeli, lai tas strādātu ar vektoru matricām, nevis vektoriem.

Ideja ir sekojoša:

1. Tokenizēt teikumu;
2. Katram tokenam aprēķināt BERT vektoru;
3. Lai visam matricām būtu vienāds lielums, papildināt matricu līdz fiksētam izmēram ar tukšiem vektoriem.

Izpētot teikumus, izrādās kā lielākai teikumu daļai tokenu skaits teikumā nepārsniedz 32. Līdz ar to, tika pieņemts matricas izmērs 32x768, kur 768 ir BERT vektora izmērs. Lai pārbaudītu cik labi šī metode strādā, tas ir pilnīgi pietiekami, jo tika ņemta tikai neliela daļa no visiem teikumiem. Trenējot modeli, kas brīvi tulkos jebkuru teikumu, ir nepieciešams pēc iespējas vairāk datu, kas nāk no dažādiem avotiem. Kā arī ir šādos gadījumos ieteicams izmantot citu servisu Colab vietā, jo trenēt Colab vidē ir laiku ietilpīgi, process bieži pārtraucas un beidzas servisa resursi. Eksperimenta ietvaros tika ņemti tikai 1000 teikumi, kur 800 ir trenēšanas kopa un 200 ir validācijas kopa.

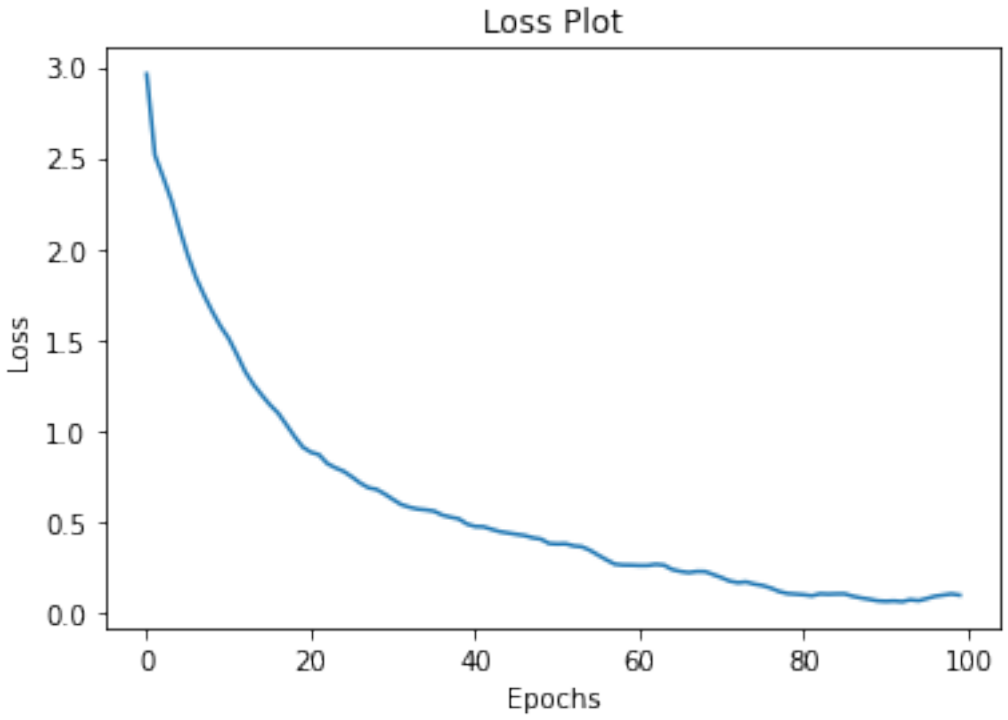
Trenēšanas struktūra ir tāda pati, kā 2.2. attēlā mainās tikai vektors. Tā vietā ir matrica ar visiem teikuma tokeniem. Loss-funkcijas vērtības ir apkopotas 2.10. attēlā. Pirmskods modeļu trenēšanai ir pieejams šeit:

https://colab.research.google.com/drive/1Wboo_Jm3aPRQmNAW_B-ImbaM7MO3lByT

Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.11.

Reals teikums	Atkodēts teikums
a kitchen with a stove sink and <unk> area.	a sink dishwasher and dishes
a woman is taking a picture of herself in a bathroom mirror	a woman is taking a picture of herself in a toilet while a picture of herself in a bathroom mirror
a man sitting on a post talking on a cell phone.	a man sitting on a cell phone
the a car that is stopped at a red light at an intersection	a car is light
a bicycle <unk> against a fence near some flood waters	a bicycle through a bicycle through a bicycle through a park next to a park next to a park bench near a bicycle stands

2.11. tabula. Tulkojuma rezultati trenējot modeli izmantojot visus BERT vektorus.



2.10. attēls. Loss-funkcijas vertības izmantojot visus BERT vektorus.

Kā redzams, rezultāti stripri atšķiras no 2.3.2. un līdzīgiem ekperimentiem. Tulkojums nav ideāls, bet tomēr ir tuvāks. Dažkārt teikuma konteksts ir tāds pats kā sakotnējam teikumam. Loss-funkcija tuvojas 0, teikumu atkodēšana no trenēšanas datiem gandrīz ideāli atkārto sākotnējo teikumu. Tāpēc, ka BERT trenējas uz citiem uzdevumiem, tas nav pielāgots tulkošanai, bet tomēr spēj izveidot pietiekami kvalitatīvu valodas modeli.

2.3.7. Visu tokenu izmantošana tulkošanas uzdevumam. XLM-R modelis.

Šis eksperiments atkārtos iepriekšēja ekperimenta idejas. Mainīsies tikai pretrenēts modelis. Eksperimenta ietvaros tiks izmantots jaunais XLM-R modelis, kuru nesen izlaida Facebook AI komanda. Šis modelis balstās uz (Conneau & Khandelwal et al, 2020) rakstu.

https://colab.research.google.com/drive/1T6ClG2TqYRd1BEq8dA00WTfA0yneXaE1

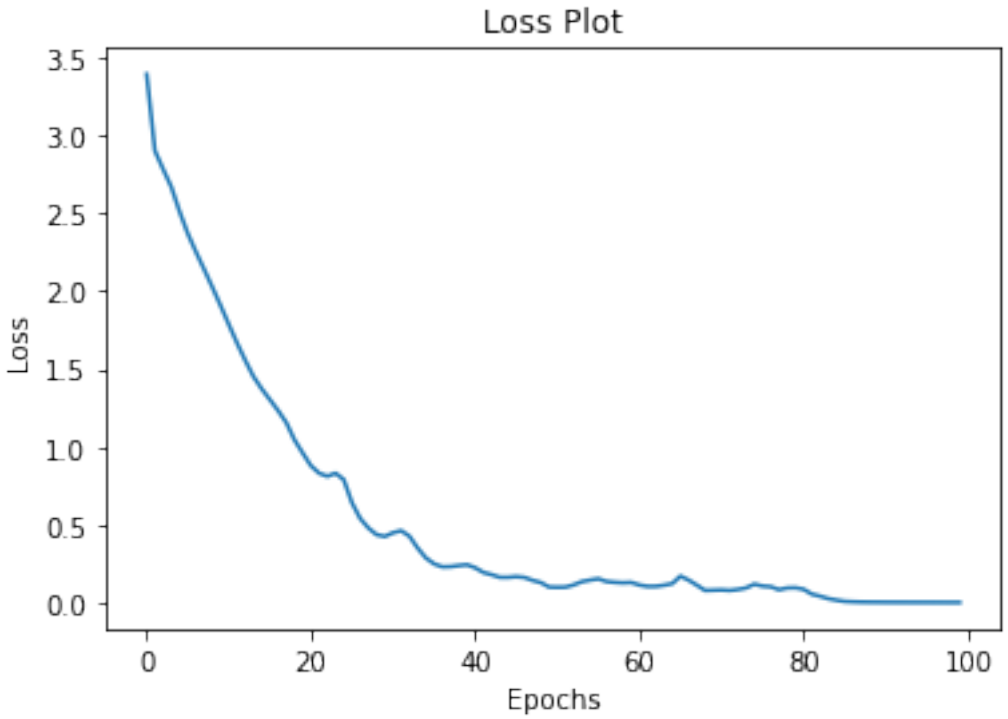
Pirmajā ekperimentā modelis tiek trenēts vektoru atkodēšanas uzdevumam. Tokeni tiek iegūti no angļu valodas tekstiem. Eksperimentā tiek izmantots XLM-R lielais modelis. Tā kā modelis ir izveidots ar Facebook AI komandu, tas ir veidots ar PyTorch. Modeli iespējams lejupielādēt no «Torch Hub» izmantojot komandu - torch.hub.load(). Kad modelis ir ielādēts, to vajag sagatavot ar komandu - eval(). Kad tas ir izdarīts, ir iespējams enkodēt teikumus. Teikumu enkodēšana notiek līdzīgi kā Keras tokenizatoram. Teikums ir sadalīts uz vārdu daļām un tie tiek aizvietoti ar unikālo ciparu no vārdnīcas. Talāk šie cipari tiek izmantoti, lai iegūtu vektoru. Vektoru iegūšana notiek izmantojot extract_features() komandu. Tā paņem

pēdējos slāņus un izrēķina vidējo vektoru katram tokenam. Pēc tam visas operācijas atkārtojas tāpat kā iepriekšējā ekperimentā. Vektori tiek papildināti līdz fiksētam izmēram, apgriezti un pārbaudīti.

Loss-funkcijas vērtības ir apkopotas 2.11. attēlā. Teikuma atkodēšanas rezultātu piemēri ir apkopoti tabulā 2.12.

Reals teikums	Atkodēts teikums
a kitchen with a refrigerator microwave dishwasher and sink	a shot of a kitchen and a microwave sink dishwasher and refrigerator
a man in a wheelchair and another sitting on a bench that is overlooking the water	a man with one on a wheelchair sitting by a skateboard at a table
sheep standing next to a stone wall in front of a tree	a bicycle in front of a sheep are in front of a stone building
a kitchen counter with cutting board knife and food	a kitchen with smoke knife and cutting board sick and vegetables are floating on it
a black and white cat sits near a window looking outside	a black and white cat

2.12. tabula. Teikuma atkodēšanas rezultati trenējot modeli izmantojot visus XLM-R vektorus.



2.11. attēls. Loss-funkcijas vērtības izmantojot visus XLM-R vektorus.

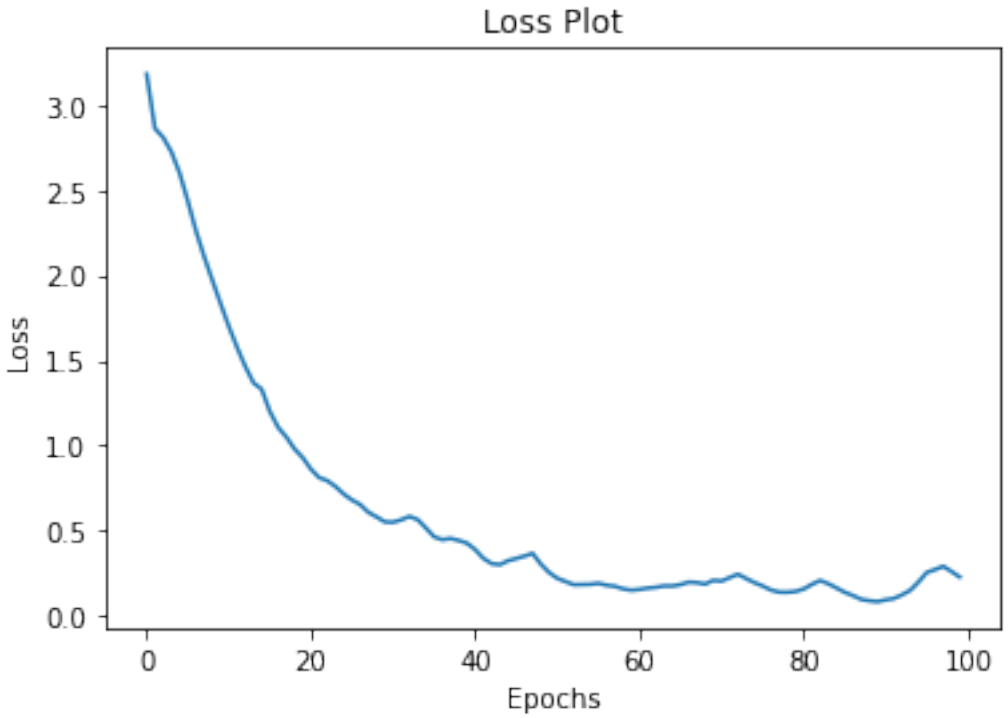
Loss-funkcijas beigas gandrīz ideāli atkārto sākuma teikumu un tā vērtība tuvojas nulei. Savukārt atkodēšanas rezultāti ir pietiekami labi. Daži vārdi ir lieki vai nepareizi, bet kopumā ir tuvi sākotnējam teikumam.

Otrajā ekperimenta daļā ir teikumu tulkojums no angļu uz latviešu valodu. Šis ekperiments pilnīgi atkārto pirmo daļu. Atšķiras tikai sākuma dati. Darbam tika ņemti latviešu valodas teikumi, tie tokenizēti un paņemti no tiem vektori. Kā arī kods ir papildināts ar jauno funkciju «translate()». Šī funkcija atkārto visus soļus, tāpat kā trenēšanā, bet tikai vienam teikumam. Teikums ir tokenizēts, pēc tam tiek iegūta vektoru matrica, un tā ir papildināta. Beigās ir izsaukta «evaluate()» funkcija, lai iegūtu rezultātus no trenēta LSTM modeļa.

Loss-funkcijas vērtības ir apkopotas 2.12. attēlā un teikuma tulkošanas rezultātu piemēri ir apkopoti tabulā 2.13.

Angļu valodas teikums	Latviešu valodas teikums
two lines of motorcycles driving down the road while a crowd watches	divi motociklisti brauc pa ceļu kamēr pūlis vēro
a picture taken from the driver seat of car at a stop sign	attēls no automašīnas vadītāja sēdekļa pie apstāšanās zīmes
a small wooden table covered with delicious vegetables	mazs koka <unk> kas karājas pa koka <unk> kas strādā pie <unk> dārzeņiem
restaurant staff taking a break near the bar	restorāna <unk> kas atrodas netālu no bāra
a group of people ski down the mountain slope	grupa cilvēku grupa cilvēku grupa cilvēku slēpoja lejup pa kalna nogāzi
a men's restroom with a poster of a woman over the urinals	pie sienas virs pisuāriem karājas sievietes <unk> virs pisuāriem karājas sievietes attēls ar plakātu par <unk>
full grown cat laying down and sleeping on top of a car	kaķis kas nogūlies un guļ uz mašīnas
a black motorcycle parked in front of trees	uz ielas novietota stāvvietā
a sink next to a large white door	izlietne pie lielām baltām durvīm
a clean steel industrial kitchen with minimal lighting	tīra tērauda <unk> virtuve ar <unk> apgaismojumu

2.13. tabula. Tulkojuma rezultati trenējot modeli izmantojot visus XLM-R vektorus.



2.12. attēls. Loss-funkcijas vērtības izmantojot visus XLM-R vektorus tulkojumu uzdevumam.

Dažos tulkojumos atkārtojas «<unk>» tokens. Tas ir tāpēc, ka vārdnīcas lielums ir ierobežots un palielinot to, šo tokenu biežums samazināsies. Tulkojuma rezultāti ir diezgan labi, daži vārdi ir izlaisti, bet kopumā teikuma konteksts ir pareizs.

2.3.8. Modeļu rezultāti tulkojumu uzdevumam izmantojot tokenu vektorus.

Izmantojot gatavus pretrenētus modeļus ir iespējams diezgan veiksmīgi tulkot tekstu, bet nepieciešams ņemt vērā zemāk minētos punktus:

- Tulkojuma precizitāte ir atkarīga galvenokārt no ieejas vektora vai matricas, kas ir padota uz LSTM;
- Teikuma vektori zaudē lielāku informācijas daļu un tos ir iespējams izmantot tikai dažos klasifikācijas uzdevumos, kur ir nepieciešama tikai kopēja informācija par teikumu vai vajag atšķirt vienu teikumu no otras;
- Izmantojot pretrenētus modeļus nav nepieciešams neliels datu apjoms vai aprēķināšanas jauda. Lai saprastu vai modelis un pieeja strādā, ir nepieciešami vismaz 1000 teikumi;
- Lai iegūtu pēc iespējas labāku valodas modeli ir svarīgi ar kādiem datiem un kādam uzdevumam tiek trenēts modelis. Tulkojumu uzdevumam ir nepieciešams izmantot modeļus, kas tiek trenēti uz vairākām valodām.

2.4. Anotācijas ģenerācija un tulkojumu uzdevumu apvienošāna vienā modelī.

Pēdējā ekperimenta ietvaros tika mēģināts apvienot attēlu matricu ar teikuma matricu un šiem datiem uztrenēt modeli. Eksperimenta mērķis ir apstiprināt vai noraidīt hipotēzi:

«Nav svarīgs no kāda avota tiek iegūta teikumu matrica, galvenais cik precīzi matrica reprezentē sākotnējo teikumu.»

Lai uztrenētu modeli uz dažādiem datiem, sākumā nepieciešams unificēt tos. Šim nolūkam ir atkārtoti apstrādāti MS-COCO dati. Dati ir pārveidoti JSON formātā. Formāts ir saraksts ar sekojošiem ierakstiem:

{

«en»: <anotācija angļu valodā>,

«lv»: <anotācija latviešu valodā>,

«image»: <saite uz attēlu>

}

Pimskods, kas apkopo datus ir pieejams šeit:

https://github.com/emukans/europarlament-clean/blob/master/coco.py

Pimskods, kas uztrenē modeli ir pieejams šeit:

https://colab.research.google.com/drive/13rBLZIrBoXB64KVNpipe8Rwgg0YUBg9z?usp=sharing

Šis eksperiments balstās uz iepriekšējo XLM-R tulkošanu un anotācijas ģenerēšanu ar InceptionV3. Datu unifikācija notiek sekojoši:

1. Lejupielādēt apkopotus datus no Google Storage;
2. Pievienot speciālus tokenus «<start>» un «<end>», kas tiek izmantoti iepriekšējos eksperimentos;
3. Lejupielādēt attēlus lokāli un pārbaudīt vai tie ir korekti;
4. Enkodēt angļu valodas anotācijas ar XLM-R modeli;
5. Pielietot lineārās transfomācijas iegūtām matricām, lai to izmērs būtu 64x2048. Matricas sākonējais izmērs ir mazāks, tāpēc vajag paplašināt matricu ar nulēm;
6. Enkodēt attēlus ar InceptionV3 modeli;
7. Uztrenēt modeli tāpat kā iepriekšējos eksperimentos.

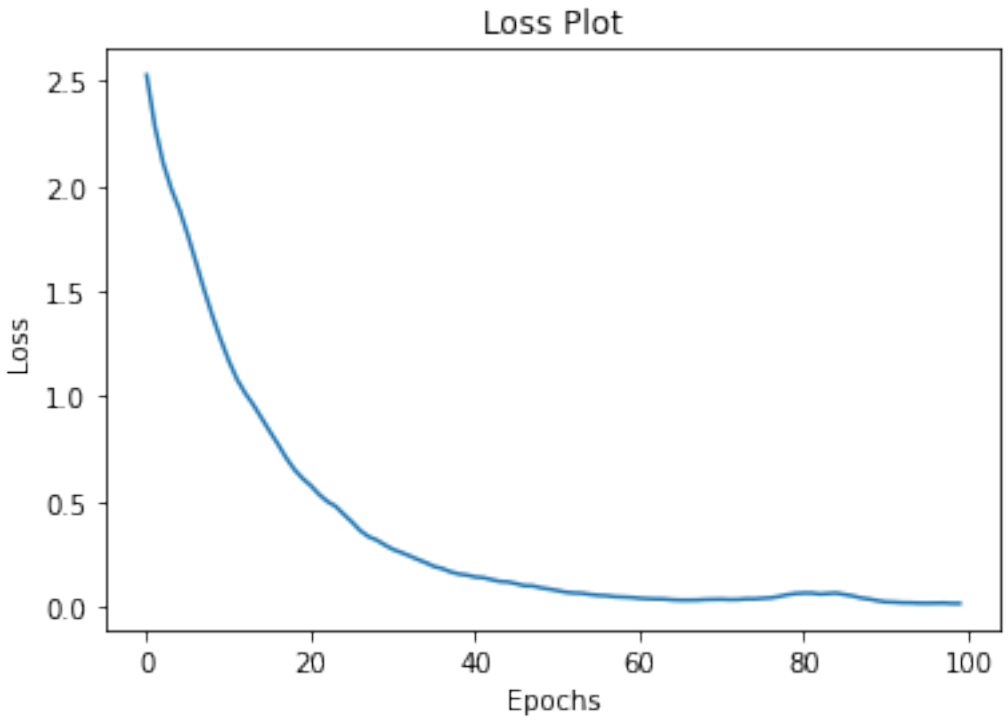
Datu struktūra LSTM modeļu trenēšanai ir parādīta 2.13. attēlā. Kā redzams, XLM-R matricas ir apvienotas ar InceptionV3 matricām. Tā kā abas matricas ir vienas anotācijas

dažādas reprezentācijas, latviešu valodas anotācijas atkārtojās divas reizes. Pirms trenēšanas dati ir samaisīti.

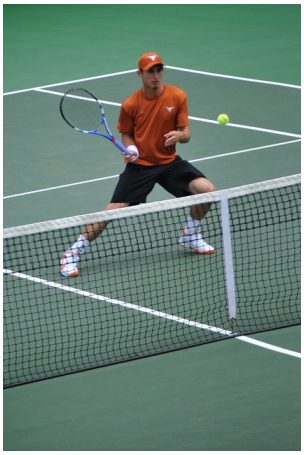
LSTM input	LSTM expected output
XML-R embedding	LV annotations
InceptionV3 embedding	LV annotations

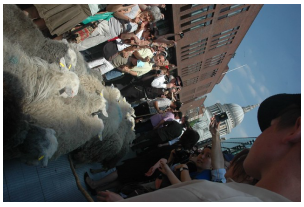
2.13. attēls. LSTM modeļu trenēšanas datu struktūra.

Tālāk trenēšana notika tāpat kā iepriekšējos eksperimentos. Trenēšana notika tikai uz 1800 anotācijām. 20% ir testa kopa un 80% ir trenēšanas kopa. Loss-funkcijas vērtības ir apkopotas 2.14. attēlā. Rezultatā modeļu ieejai iespējams padod teikumu angļu valodā vai padod attēlu un iegūt teikumu latviešu valodā. Līdz ar to ir iespējams salīdzināt kāda teikumu reprezentācija ir precīzāka - reprezentācija, kas iegūta ar InceptionV3 vai XLM-R. Salīdzinājumu rezultāti ir apkopoti 2.14. tabulā. Šie rezultāti ir ņemti tikai no testa kopas.



2.14. attēls. Loss-funkcijas vērtības izmantojot XLM-R un InceptionV3 matricas.

Reāls teikums	Tulkots teikums ar XLM-R	Tulkots teikums ar InceptionV3	Attēls
a man is playing tennis and is going to return the ball	vīrietis spēlē tenisu un gatavojas atgriezt bumbu	tenisa raketi	
three females impatiently waiting. two of them are on their cell phones	trīs sievietes <unk> gaida ka divas no <unk> <unk>	cilvēks kurš <unk> aiz diviem maziem malciņiem	
a piece of luggage is ready to go with cold wear on top for quick usage.	bagāžnieka gabals ir gatavs lai to ātri <unk>	uz šosejas barjeras	
a grassy hill side with some animals in the distance	zālaina kalna puse ar dažiem dzīvniekiem <unk>	šī ir tā bilde ar kokiem un cilvēkiem	

Reāls teikums	Tulkots teikums ar XLM-R	Tulkots teikums ar InceptionV3	Attēls
a woman holding a tennis racket on a court.	sieviete ar tenisa raketi uz tenisa raketes	meitene ir gatava trāpīt pa tenisa bumbiņu	
a living room displaying family photos and art.	dzīvojamā istaba kurā ir dzīvojamās istabas vidū	dzīvojamā istaba kurā attēlotas ģimenes fotogrāfijas un māksla	
sheep running through town all together in a crowd	aitas kas skraida pa pilsētu drūzmējas pūlī	aitas kas skraida pa pilsētu drūzmējas pūlī	
there is a green frisbee in the air with red chairs in the background	šī ir zooloģiskā dārza spoguli	gaisā ir zaļā frisbija ar sarkaniem krēsliem fonā	
a woman holding a red and white umbrella while standing in a cave	sieviete ar sarkanu un baltu lietussargu stāvēdama alā	vīrietis kas stāv ar diviem zariem	
two computer screens that are sitting on a desk	divi datora ekrāni kas atrodas uz galda	uz datora galda ir divi monitori un cits aprīkojums	

2.14. tabula. Tulkojuma modeļa trenēšanas rezultāti, izmantojot XLM-R un InceptionV3 matricas.

No sekojošiem rezultātiem nav iespējams viennozīmīgi pateikt kāds modelis uzģenerēja labāku anotāciju. Kopumā, abi modeļi dod pietiekami labu precizitāti un diezgan bieži rezultāti ir vienādi. Tas nozīmē, ka abi modeļi uzbūve savu valodas modeļi, kas prot tulkot tekstus atšķirīgi. Šajā gadījumā tulkojuma precizitāte ir aptuveni vienāda abiem modeliem, kas apstiprina eksperimenta hipotēzi.

SECINĀJUMI

Maģistra darba ietvaros ir izpētītas dažādas pieejas, lai pārtulkotu teikumu vai atkodētu teikumu tajā pašā valodā. Ir izstrādāti neironu mašīntulkotāji, kas spēj pārtulkot tekstu no angļu uz latviešu valodu. Rezultāti nav labāki un nav jēgas tos salīdzināt ar eksistējošiem risinājumiem, piemēram, tādiem kā Google Translate vai Tildes tulkotājs. Bet modelis spēj pietiekami precīzi iztulkot tekstu uz latviešu valodu, un teksts ir gramatiski pareizs, kā arī vārdu izvēle ir optimāla. Līdz ar to, darba secinājumi ir sekojoši:

- Pirmajā daļā tulkojuma rezultāti ir diezgan kvalitatīvi. Bet problēma ir datu trūkumā un datu dažādībā. Tā kā modeļis ir trenēts no jauna, lai uzlabotu rezultātus, ir nepieciešams lielāks un pēc iespējas dažādāks datu apjoms;
- Lai uztrenētu modeli, kas balstās uz pretrenēto modeli, nepieciešami lieli izrēķināšanās resursi. Colab resursu nepietiek, lai ilgi trenētu modeli uz lielā datu apjoma. Trenēšānas laikā tika meģināts uztrenēt modeli tikai uz MS-COCO anotācijas datiem, kas ir tikai 32 mB. Trenēšana notika ļoti ilgi un bieži tika pārtraukta resursu trūkuma dēļ. Lai uztrenētu pietiekami kvalitātīvu tulkotāju ir nepieciešami vismaz pāris simti megabaiti ar datiem;
- Teikumu vektoru izmantošana tulkošanas uzdevumam nav pieļaujama. Lietojot pretrenētus modeļus nepieciešams izmantot pēc iespējas vairāk informācijas par teikumu. Teikumu vektoru ir pieņemams izmantot tikai uzdevumos, kur nav svarīga precīza informācija par teikumu. Piemēram, teikumu klasifikācija, jautājumu atbildēšana u.c. Lai iegūtu pēc iespējas kvalitatīvako teikumu vektoru ir nepieciešams precizēt pretrenētu modeli, tāpat ka ir izdarīts S-BERT modelī;
- «Transformer» arhitektūra spēj veiksmīgi tulkot tekstus. Tas ir pieradīts pirmajā ekperimentā, kad ir uzbūvēts modelis no jauna un otrajā ekperimentā, kad izmantoja «XLM-R» modeli, kas arī izmanto «Transformer» arhitektūru;
- Iegūti modeļi, kas balstas uz «Transformer» arhitektūru aptver uzreiz visu teikumu. Tas nozīmē, ka modelis nezaudē kontekstu un spēj tulkot teikumu jebkurā garumā nezaudējot kvalitāti. Protams, tas ierobežo teksta garumu. Piemēram, darbā tika iestatīts ierobežojums uz 64 tokeniem, lai būtu iespējams uztrenēt LSMT modeļi. Bet izmantojot šo pieeju reālos modeļos, tekstu ir iespējams sašķelt un palielināt tokenu skaitu, piemēram, līdz 1024 tokeniem.

- Kāds ir datu tips un kāds modeļis tiek izmantos, lai iegūtu teikuma reprezentāciju (matricu vai vektoru) nav svarīgs. Būtiska ir tikai tā teikumu matrica, kas pēc iespējas kvalitatīvāk reprezentē teikumu kopējā datu mākonī. Lai teikumi, kas ir līdzīgi pēc konteksta atrastos tuvāk.

PRIEKŠLIKUMI

Darbu iespējams attīstīt tālāk, veicot sekojošos soļus:

1. Savākt pēc iespējas vairāk datu no dažādiem avotiem un par dažāadm tēmām, lai modelis nebūtu virzīts uz specifiskām tēmām. Pašlaik pieejamas tikai MS-COCO anotācijas, Eiroparlamenta sesijas dati un Eiropas Investīcijas bankas sesijas dati. Šo datu korpusu pietiek, lai uztrenētu precīzu modeļi tikai specifiskiem tekstiem;
2. Veikt trenēšanu uz visiem datiem. Darbā tika izmantots Colab modeļu trenēšanai. Tas ierobežo modeļu lielumu un pašu trenēšanu. Līdz ar to nebija iespējams uztrenēt modeli uz visiem pieejamiem datiem, izmantojot pretrenētus modeļus. Trenēšana notika ļoti ilgi un pieprasīja nepārtrauktu procesa monitoringu;
3. Salīdzināt iegūtos modeļa tulkojumus ar esošiem tulkotājiem - Tildes tulkotāju un Google translate. Salīdzināšanai izmantot kvalitatīvus indikatorus tādus kā BLEU.

IZMANTOTA LITERATŪRA

1. Alammar, «The Illustrated Transformer», 2018. Pieejams: https://jalammar.github.io/illustrated-transformer/

2. Al-Rfou et al., «Character-Level Language Modeling with Deeper Self-Attention», 2019. Pieejams: https://arxiv.org/pdf/1808.04444.pdf

3. Bahdanau et al., «Neural Machine Translation by Jointly Learning to Align and Translate», 2014. Pieejams: https://arxiv.org/pdf/1409.0473.pdf

4. Cho et al., «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation», 2014. Pieejams: https://arxiv.org/pdf/1406.1078.pdf

5. Chiu et al., «State-of-the-art Speech Recognition With Sequence-to-Sequence Models», 2018. Pieejams: https://arxiv.org/pdf/1712.01769.pdf

6. Conneau & Khandelwal et al, «Unsupervised Cross-lingual Representation Learning at Scale», 2020. Pieejams: https://arxiv.org/pdf/1911.02116.pdfs

7. Dai et al., «Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context», 2019. Pieejams: https://arxiv.org/pdf/1901.02860.pdf

8. Denis Antyukhov, «Improving sentence embeddings with BERT and Representation Learning», 2020 29 Februārī. Peejams: https://towardsdatascience.com/improving-sentence-embeddings-with-bert-and-representation-learning-dfba6b444f6b

9. Devlin et al., «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding», 2019. Pieejams: https://arxiv.org/pdf/1810.04805.pdf

10. Google Developers, «Image captioning with visual attention». Pieejams: https://www.tensorflow.org/tutorials/text/image_captioning

11. Luong et al., «Effective Approaches to Attention-based Neural Machine Translation», 2015. Pieejams: https://arxiv.org/pdf/1508.04025.pdf

12. Sutskever et al., «Sequence to Sequence Learning with Neural Networks», 2014. Pieejams: https://arxiv.org/pdf/1409.3215.pdf

13. Vaswani et al., «Attention Is All You Need», 2017. Pieejams: https://arxiv.org/pdf/1706.03762.pdf

Maģistra darbs: **Alternatīvas neironu mašīntulkošanas arhitektūras**

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____
(Vadītāja paraksts)

Darbs iesniegts maģistrantūras sekretariātā _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: Dr. dat. , Mārcis Pinnis
(Akad. amats, zin. grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)