TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Andmeteaduse õppekava

Raul Niit

BERT mudeli kohandamine eesti keelele

Magistritöö (15 EAP)

Juhendajad: Sven Laur, DSc (Tech)
Hendrik Šuvalov, MSc

Tartu 2023

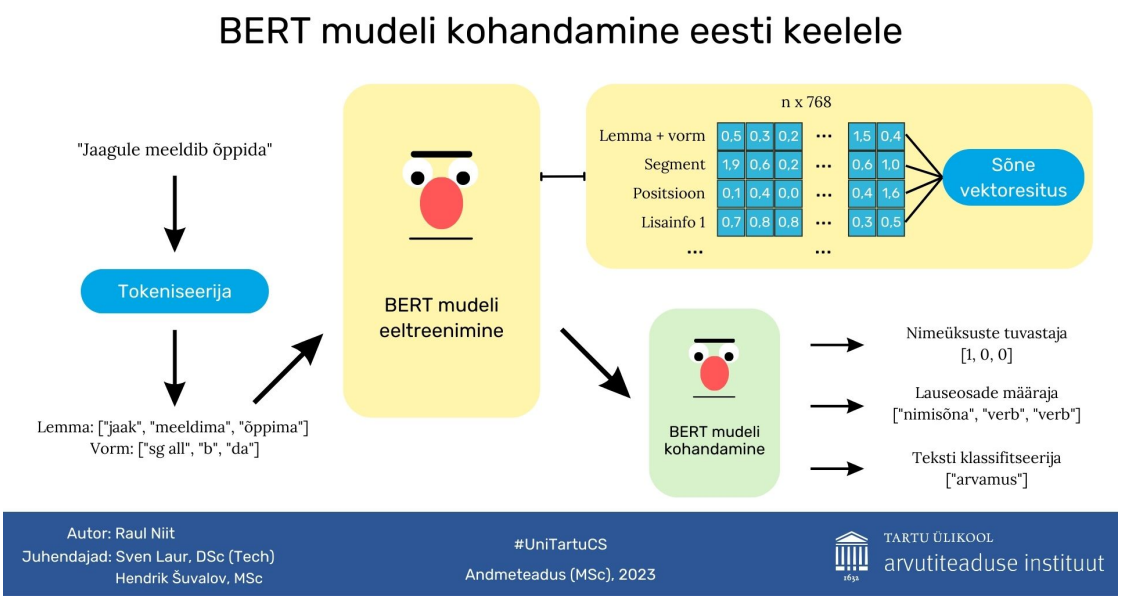
BERT mudeli kohandamine eesti keelele

Lühikokkuvõte:
Keelemudelite kiire areng on muutnud arvutid meie elus osavateks inimkeele kasutajateks, mille abil on tänapäeval võimalik lihtsa vaevaga lahendada mitmeid erinevat tüüpi keeleülesandeid, olgu selleks siis tekstide tõlkimine, klassifitseerimine või uue teksti genereerimine. Aastal 2018 Google teadlaste poolt loodud keelemudel BERT on tänaseni tänu oma võimsale arhitektuurile ja avatud lähtekoodile üks populaarsemaid keelemudelid. Mudeli täiustamiseks on loodud ka konkreetse keele põhiseid BERT mudeleid nagu aastal 2020 loodud EstBERT, mis on kohandatud eestikeelsete ülesannete jaoks. Magistritöö eesmärk on muuta BERT mudeli arhitektuuri nii, et see võimaldaks mudelis kasutada täiendavat morfoloogilist infot sisendi kohta nagu sõnade lemmad ja vormid. Töös treenitakse muudetud arhitektuuriga mudel välja ning analüüsitakse mudeli suutlikkust neljal keeleülesandel.

Võtmesõnad:
Loomuliku keele töötlus, BERT, EstBERT

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine
P176 Tehisintellekt

Visuaalne kokkuvõte:



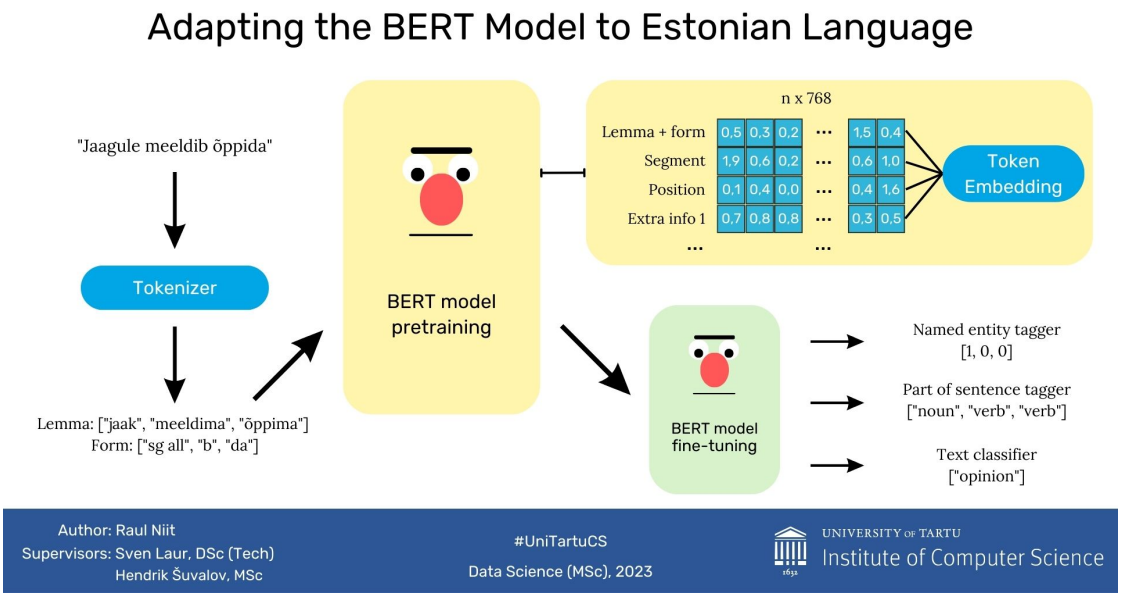
Adapting the BERT Model to Estonian Language

Abstract:
The rapid development of natural language models has turned computers into skilled users of human language, which can be used to solve different types of language tasks such as translation, classification or generation of text. The BERT language model which was created by Google researchers in 2018 is still one of the most popular natural language models today thanks to its powerful architecture and open source framework. BERT models based on a specific language have also been created, such as EstBERT created in 2020, which is adapted specifically for tasks in the Estonian language. The aim of this master's thesis is to change the architecture of the BERT model so that additional morphological information about the input such as lemmas and forms could be used in the model. In this work the modifications of the model are implemented and the performance of the model is analyzed on four natural language tasks.

Keywords:
Natural language processing, BERT, EstBERT

CERCS:
P170 Computer science, numerical analysis, systems, control
P176 Artificial intelligence

Visual abstract:



Sisukord

1 Sissejuhatus

Loomuliku keele töötlus on arvutiteaduse haru, mis tegeleb loomuliku keele ehk inimkeele mõistmise ning analüüsimisega arvutites. Loomuliku keele töötluse arendamine on oluline, sest see võimaldab arvutite abil lahendada inimkeelt käsitlevaid ülesandeid ja probleeme nagu tekstilisel kujul oleva info automaatne süstematiseerimine või petukirjade tuvastamine meilides. Tänapäeval kasutatakse loomuliku keele töötluseks peamiselt masinõppel ja närvivõrkudel põhinevaid keelemudeleid nagu GPT-3 ja BERT (Ruder, 2018).

BERT mudel on loomuliku keele töötluse mudel, mille töötasid välja Google teadlased 2018. aastal. BERT mudeli näol on tegemist kahefaasilise mudeliga, kus esmalt eeltreenitakse mudelit märgendamata andmetel ning seejärel kohandatakse mudelit märgendatud andmestikul spetsiifilise ülesande jaoks. Taoline mudeli ülesehitus võimaldab mudelit rakendada erinevate ülesannete tarbeks, mida inimkeelelisetel tekstidel lahendada saab nagu näiteks teksti rubriigi klassifitseerimine või nimeüksuste tuvastamine tekstis. Avaldamise hetkel edestas BERT mudel teisi keelemudeleid mitmetel keeleülesannete jaoks loodud testandmestikel (Devlin *et al.*, 2018).

Kuna maailmas on palju erinevaid keeli, on ka BERT mudeleid loodud erinevate keelte tarbeks. Eesti keele jaoks loodud EstBERT on treenitud ainult eestikeelsetel tekstiandmetel, millest tulenevalt annab mudel täpsemaid tulemusi eestikeelsetel keeleülesannetel kui mitmekeelsetel tekstidel treenitud mudelid (Tanvir *et al.*, 2020). Kuna käesolevas töös vaadeldakse niisamuti eestikeelseid keeleülesandeid, on EstBERT mudeli loomisel läbiviidud protsessid aluseks täiendatud mudeli loomisele.

Magistritöö eesmärk on uurida, kuidas saab olemasolevaid reeglipõhiseid meetodeid kasutada BERT mudeli täiendamiseks ning kuidas lisada BERT mudeli sisendile märgendusülesandeid lihtsustavaid signaale. Töös muudetakse BERT mudeli arhitektuuri esimesi mooduleid ehk mudeli tokeniseerijat ning mudeli vektoresitusi. Mudeli sisendtekstist eraldatud sõnedele (*token*) lisatakse Pythoni teegi EstNLTK abil morfoloogiline analüüs sõna algvormi ehk lemma ja sõna vormi ehk käände või pöörde näol. Morfoloogilise info kaasamine võiks hüpoteetiliselt aidata kaasa mudeli täpsemale ennustamisele. On näidatud, et ka eestikeelsete keeleülesannete lahendamisel annab BERT mudelile morfoloogilise info lisamine paremaid tulemusi (Klemen, Krsnik ja Robnik-Šikonja, 2020). Täiendatud mudel treenitakse välja ning analüüsitakse mudeli suutlikkust erinevatel keeleülesannetel.

Töö teises peatükis tutvustatakse BERT mudeli toimimist ning eeltreenimise ja kohandamise protsesse. Töö kolmandas peatükis seletatakse lahti BERT mudelile implementeeritud muudatused. Neljandas peatükis esitatakse treenitud mudelite tulemused ning

analüüsitakse täiendatud mudeli võimekust keeleülesannete lahendamisel. Töö lisas on välja toodud detailsemad treenitud mudelite tulemused.

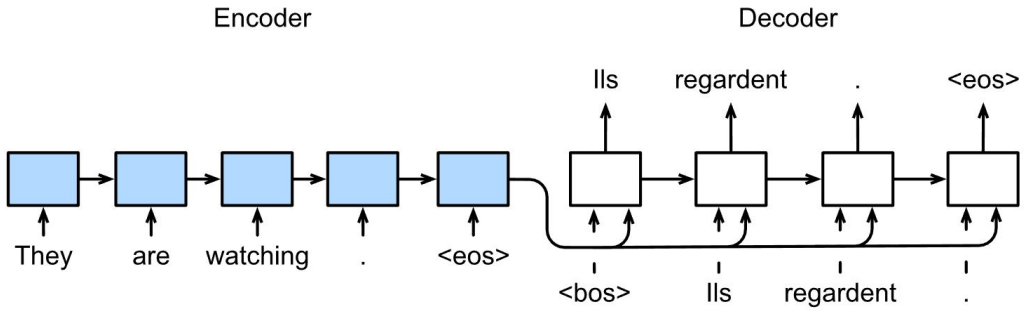
2 BERT mudel

BERT on 2018. aastal Google teadlaste poolt loodud loomuliku keele töötluse mudel, mida saab kasutada erinevate keeleülesannete lahendamiseks. BERT mudeli näol on tegemist baasmudeliga, mille eeltreenimisel on mudeli ülesandeks leida sobiv sõne lünga asemele, mida kutsutakse Cloze ülesandeks (Taylor, 1953). Formaalselt kasutati sarnast ülesannet keelemudelites esimest korda 2013. aastal WORD2VEC meetodites (Mikolov *et al.*, 2013). Eeltreenimise käigus õpib mudel iga sõne kohta ära vektoresituse ning seeläbi ka iga sõne semantilise tähenduse. Mudeli täiendaval treenimisel ehk kohandamisel on võimalik õpetada mudelit kasutama semantilise tähendusega vektoresitusi lahendamaks erinevaid keeleülesandeid nagu tekstide klassifitseerimine või lauseosade märgendamine tekstis (Devlin *et al.*, 2018).

Vektoresituste kasutamine inimkeele sõnade esitamiseks on oluline, kuna see võimaldab mudelite treenimisel kasutada erinevaid masinõppe meetodeid, sealhulgas närvivõrke. Idee kasutada loomuliku keele töötluse mudelis närvivõrkude abil õpitud vektoresitusi sõnade jaoks pärineb aastast 2011 (Collobert *et al.*, 2011) ning seda lähenemist arendasid edasi taaskord Google teadlased, kui 2013. aastal tutvustati WORD2VEC meetodeid (Mikolov *et al.*, 2013). Sõnade konteksti õppimiseks pakuti välja kaks varianti - *continuous Bag-of-Words* mudel, kus mudeli ülesanne on ennustada sõna sellele vahetult eelnevate ja järgnevate sõnade abil ning *continuous Skip-gram* mudel, kus mudelil tuleb etteantud sõna abil ennustada sõnale tekstis eelnevaid ja järgnevaid sõnu. See tähendab, et WORD2VEC meetoodika puhul sõltub sõna vektoresitus sõnale vahetult eelnevatest ja järgnevatest sõnadest, täpsemalt nende sõnade vektoresitustest.

Keeleülesannete lahendamine mudelite abil nõudis arenguid ka mudelite ülesehituses. Paljud keeleülesanded nagu tekstide tõlkimine või küsimusele vastuse genereerimine on oma olemuselt sellised, kus sisendjada põhjal tuleb genereerida väljundjada, mille pikkus pole seejuures ette teada. Taoliste ülesannete jaoks arendati välja SEQ2SEQ mudel, mis kasutab samuti sõnede vektoresitusi ning kodeerija-dekodeerija (*encoder-decoder*) arhitektuuri (Sutskever, Vinyals ja Le, 2014) . Nii kodeerija kui ka dekodeerija on rekurrentsed närvivõrgud. Kodeerijas saab iga närvivõrgu element (*cell*) sisendiks sisendjada liikme ning eelneva närvivõrgu elemendi väljundi. Kodeerija väljundiks on sisendjada põhjal leitud kodeeritud vektor. Dekodeerijas leitakse kodeeritud vektori alusel väljundjada, kus iga väljundjada ühik väljastab sõne nii kaua kuni mõni ühikutest väljastab väljundi lõpu sõne (<eos>). Kodeerija-dekodeerija arhitektuur on esitatud joonisel 1.

Siinkohal mõeldakse sõne all tokeniseeritud sõna (*token*). Tokeniseerimine on teksti kui märgijada jaotamine sõnedeks, mis võivad aga ei pea olema loomuliku keele sõnad. Kõikide sõnede loetelu moodustab mudeli sõnastiku. Tokeniseerimine on oluline, kuna keelemudeli jaoks on kõikide keeles leiduvate sõnade vektoresituste treenimine



Joonis 1. Kodeerija-dekodeerija arhitektuur (Zhang *et al.*, 2021). Joonisel tähistavad <bos> ja <eos> vastavalt lause alguse sõne (*beginning of sentence*) ja lause lõpu sõne (*end of sentence*). Sinised ristkülikud tähistavad kodeerija närvivõrgu elemente ning valged ristkülikud dekodeerija närvivõrgu elemente.

ja meeleshoidmine küllaltki ressursimahukas. Seetõttu on tarvilik leida tasakaal - kui sõnesid on mudeli sõnastikus liiga vähe ei suuda mudel piisavalt hästi selgeks õppida keele semantikat, samas mida rohkem on sõnesid mudeli sõnastikus, seda arvutuslikult keerukam on mudel. Tokeniseerimisest räägitakse lähemalt alapeatükis 2.1.

BERT mudel on modifitseeritud transformer mudel, mis põhineb Vaswani *et al.*, 2017 avaldatud artiklil. Transformer mudel on oma olemuselt küll kodeerija-dekodeerija arhitektuuriga, kuid kasutab rekurrentsete närvivõrkude asemel enesetähelepanu (*self-attention*) mehhanismi (Devlin *et al.*, 2018). Paljud tänapäevased keelemudelid põhinevad transformeri arhitektuuril (*The Hugging Face Course* 2022). Sealjuures eristatakse vastavalt ülesehitusele kolme tüüpi mudeleid:

1. Ainult kodeerija mudelid: BERT, RoBERTa, DeBERTa, ...
2. Ainult dekordeerija mudelid: GPT-3, PaLM, XLNet, ...
3. Kodeerija-dekodeerija mudelid: T5, BART, ...

Lisaks unikaalsele arhitektuurile peitub BERT-tüüpi mudelite eripära kahesuunalises sõnede esituste õppimises. See tähendab, et erinevalt ühesuunalistest keelemudelitest nagu OpenAI GPT mudelid, kus ennustamisel käsitleb mudel vaid sõnele eelnenud sõnesid, siis BERT mudelis käsitletakse kogu mudeli sisendit korraga, võimaldades mudelil iga sõne konteksti õppimiseks kasutada ka sisendis tagapool esinevaid sõnesid (Devlin *et al.*, 2018).

Avaldamise hetkel saavutas BERT parimaid skoore mitmetel keelemudelite testimiseks loodud andmestikel ja ülesannetel nagu GLUE ja SQuAD 2.0. (Devlin *et al.*, 2018)

Tabel 1. BERT, RoBERTa ja DeBERTa võrdlus erinevatel ülesannetel ja andmestikel (He *et al.*, 2021). SQuAD v2.0 (Rajpurkar, Jia ja Liang, 2018), RACE (Lai *et al.*, 2017) ja SWAG (Zellers *et al.*, 2018) on andmestikud küsimustele vastuste leidmiseks konteksti põhjal, MNLI-m/mm (Wang *et al.*, 2019) on ülesanne, kus lausete paar tuleb klassifitseerida kaasarääkivaks, vasturääkivaks või neutraalseks, ning NER tähistab nimeolemite märgendamise ülesannet, mille jaoks kasutati CoNLL-2003 andmestikku (Tjong Kim Sang ja De Meulder, 2003). Acc ja EM tähistavad mudeli täpsust ning F1 tähistab mudeli f1-skoori.

Mudel	Param	Andmete maht	MNLI-m/mm Acc	SQuAD v2.0 F1/EM	RACE Acc	SWAG Acc	NER F1
BERT-LARGE	334M	16GB	86.6/-	81.8/79.0	72	86.6	92.8
DeBERTa-LARGE	345M	78GB	91.1/91.1	90.7/88.0	86.8	90.8	93.8
RoBERTa-LARGE	355M	160GB	90.2/90.2	89.4/86.5	83.2	89.9	93.4

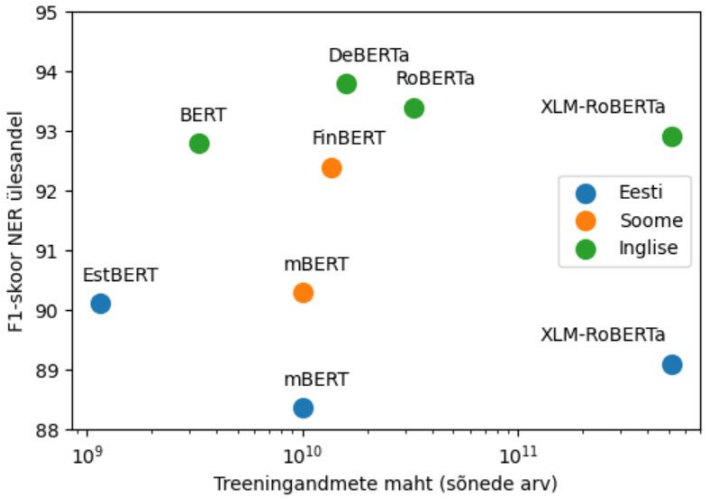
BERT mudeli võimekusest lähtuvalt hakati mudelit edasi arendama. Kui BERT mudeli treenimiseks kasutati andmestikku 3,3 miljardi sõnega, siis 2019. aastal loodud RoBERTa mudel treeniti ligi 10 korda suuremal andmestikul, sealjuures muudeti ka mudeli treenimist ning optimiseeriti treenimise parameetreid, mille tulemusena edestas RoBERTa mudel BERT mudelit mitmetes keeleülesannetes. (Liu *et al.*, 2019). Kontrollkatsena treeniti välja ka BERT mudeliga sarnase andmemahuga RoBERTa mudel, mis kinnitas, et muutused mudeli arhitektuuris ja treenimisprotsessis olid mõjukad.

2021. aastal arendatud DeBERTa mudelis muudeti enesetähelepanu mehhanismi ning võeti arvesse ka sõnede positsiooni lauses, saavutades täiendatud meetoditega veelgi paremaid tulemusi kui samadel tingimustel treenitud RoBERTa mudel (He *et al.*, 2021). Kui originaalse BERT mudeli suurem versioon hõlmas endas 340 miljonit parameetrit, oli DeBERTa mudeli suurimas versioonis juba ligi 1,5 miljardit parameetrit (He *et al.*, 2021). Tulemuste puhul tuleb silmas pidada, et mudeli treenimisel on palju erinevaid muutujaid nagu treeningandmete maht ja kvaliteet, parameetrite arv mudelis, mudeli kohandamise parameetrid ja nii edasi. Mudeleid võrdlevaid kontrollkaitseid (*ablation study*), kus kõik treeningtingimused on identsed, pole autorile teadaolevalt keegi avaldanud. Kolme mudeli sarnaste parameetrite arvuga versioonide võrdlus on esitatud tabelis 1.

Originaalis on BERT mudel treenitud ingliskeelsete tekstide põhjal. BERT mudeli autorid treenisid välja ka mBERT mudeli, kasutades selleks 104 kõige populaarsema keele vikipeedia artikleid (Devlin, 2018). Küll aga varieerub mBERT mudeli suutlikkus erinevate keelte ülesannete lahendamisel, saades kehvemaid tulemusi just väiksemate keelte seas (Wu ja Dredze, 2020). Varasemalt kasutatigi eestikeelsete keeleülesannete jaoks multikeelset XLM-RoBERTa mudelit (Kittask, Milintsevich ja Sirts, 2020), kuni loodi ainult eesti keele tekstidel treenitud EstBERT mudel (Tanvir *et al.*, 2020).

EstBERT mudeli treenimisel kasutati 2017. aasta eesti keele ühendkorpust, kokku 3,3 miljoni dokumendi tekste, mis sisaldasid endas 75,7 miljonit lauset ja 1,154 miljardit sõne (Koppel ja Kallas, 2018). Võrdluseks sisaldas originaalne ingliskeelsetel tekstidel eeltreenitud BERT mudeli treeningandmestik 3,3 miljardit sõne (16 GB), DeBERTa ja RoBERTa teeningandmestikkude mahud olid vastavalt 78 GB ja 160 GB. Teiste keelte kontekstis paigutub EstBERT mudeli eeltreenimise andmestiku mahult pigem väiksemate keelte hulka - soomekeelse FinnBERT mudeli eeltreenimise andmete maht oli 13,5 miljardit sõne (Virtanen *et al.*, 2019), kuid näiteks läti keelse LVBERT mudeli puhul oli see 0,5 miljardit sõne (Znotiņš ja Barzdins, 2020). Seega on eesti keelele oma BERT mudeli eeltreenimine igati asjakohane - EstBERT mudel saavutab mitmetes keeleülesannetes paremaid tulemusi kui multikeelsed mudelid (Tanvir *et al.*, 2020).

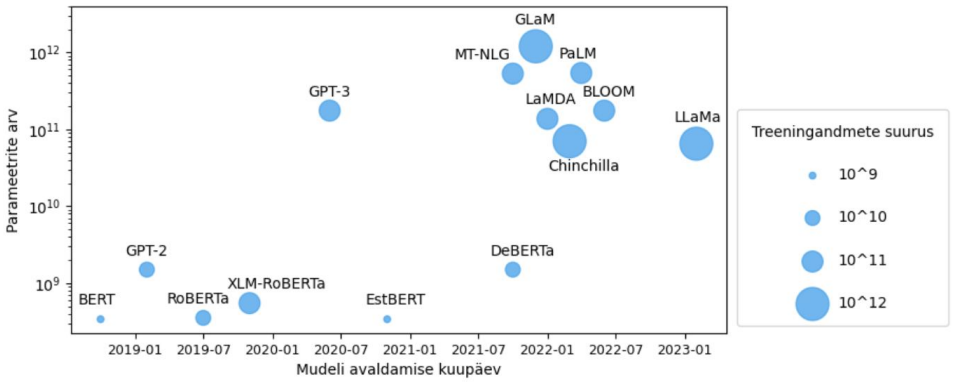
Üheks puuduseks eestikeelse mudeli treenimisel on vähene eestikeelsete tekstide olemasolu. Jooniselt 2 on näha, et treeningandmete mahult jääb EstBERT märgatavalt alla ingliskeelsetele, soomekeelsetele ning multikeelsetele mudelitele. Vähene treeningandmete hulk võib mõjutada mudeli võimekust erinevate keeleülesannete lahendamisel. Tänapäevaste keelemudelite võrdlemist raskendab asjaolu, et mitmete mudelite kohta ei avaldata piisavalt informatsiooni või ei testita neid standardsetel testandmestikel.



Joonis 2. BERT mudelite võrdlus. EstBERT mudeli jaoks kasutatav treeningandmete maht on oluliselt väiksem mitmetest teistest BERT mudelitest, mis võib olla takistuseks paremate tulemuste saavutamisel olulistes keeleülesannetes nagu nimeolemite märgendamine (NER).

Lisaks BERT-tüüpi mudelitele on laialdaselt kasutusel ka OpenAI arendatud GPT mudelid mille parameetrite arv küündib GPT-3 mudeli puhul 175 miljardini (Brown *et al.*,

2020). Samas on Deepmind teadlaste arvates keelemudelite puhul oluline just mudeli treenimiseks kasutatud andmete maht ja treenimise aeg, näidates, et nende mudel CHINCHILLA, millel on 70 miljardit parameetrit, saavutab mahukamal treenimisel paremaid tulemusi kui GPT-3 mudel (Hoffmann *et al.*, 2022). Kui originaalse BERT mudeli treeningandmestik sisaldas 3,3 miljardit sõne (Devlin *et al.*, 2018), siis CHINCHILLA mudeli treeningandmete maht oli 1,4 triljonit sõne (Hoffmann *et al.*, 2022). Keelemudelide suuruste võrdlus on esitatud joonisel 3.



Joonis 3. Suurte keelemudelite võrdlus. Tänaseks on loodud keelemudeleid, mille parameetrite arv ning treeningandmete maht on tuhandetes kordades suurem kui BERT mudelil.

Kuigi BERT-tüüpi mudelid on tänseks oma suuruselt ja võimekuselt jäänud alla mitmetele mudelitele, on need endiselt populaarsed. Üheks põhjuseks on juba eelnevalt mainitud mudeli arhitektuur, mis võimaldab mudelil treenimisel kasutada märgendamata andmeid ning teisalt arvestada sõnede konteksti õppimisel kõikide teiste sõnedega sisendis. Ilmselt üheks suurimaks eeliseks on aga BERT mudeli kättesaadavus ning võimalus mudelit oma andmetele ja ülesannetele ise kohandada, mida paljude joonisel 3 esitatud keelemudelite puhul pole võimalik teha, sest nende mudelite lähtekoodi pole avaldatud. BERT mudel on implementeeritud Pythoni paketis Transformers (Wolf *et al.*, 2020), mida ka käesolevas töös mudeli rakendamiseks kasutatakse.

Transformers pakett sisaldab endas lisaks BERT-tüüpi mudelitele ka mitmete muude populaarsete keelemudelite nagu GPT-2 või T5 implementatsioone (Wolf *et al.*, 2020). Läbi paketi vahendite on võimalik igaühel mudeleid oma arvutis kasutada ja treenida, järgides paketi dokumentatsioonis toodud juhiseid. Küll aga tuleb silmas pidada, et keelemudelite eeltreenimine nõuab üldjuhul palju resursse nagu on välja toodud tabelis 2. Sealjuures on BASE lõpuga mudelid kõige väiksemate parameetrite arvudega versioonid loodud BERT mudelitest. Seetõttu võib väita, et BERT-tüüpi mudelite eeltreenimine on

tavakasutaja arvutis teoreetiliselt võimalik, kuid see võib sõltuvalt arvuti võimekustest väga kaua aega võtta, näiteks ühe GPU peal võtaks EstBERT-base mudeli eeltreenimine hinnanguliselt aega 64 päeva ehk ligikaudu 2 kuud.

Tabel 2. BERT-tüüpi mudelite eeltreenimise kestus ja kulu.

Mudel	Jõudlus	Kestus	Kulu
EstBERT-base	4 X NVIDIA V100 GPU	16 päeva	1536 GPU tundi
BERT-base	4 X Cloud TPU	4 päeva	384 TPU tundi
DeBERTa-base	64 X NVIDIA V100 GPU	4 päeva	24576 GPU tundi
RoBERTa-base	1024 X NVIDIA V100 GPU	1 päev	15360 GPU tundi

Seevastu on mudeli kohandamine märksa vähem resursse nõudev protsess. Praktikas võetaksegi tihti olemasolev eeltreenitud mudel ning kohandatakse mudel oma andmestikule ja ülesandele. Paljud eeltreenitud mudelid on kättesaadavad Hugging Face platvormil (*Hugging Face* 2023), sealhulgas ka kõik nimetatud BERT-tüüpi mudelid. Keelemudelite eeltreenimist ja kohandamist kiirendab oluliselt GPU või TPU kasutamine.

2.1 Mudeli eeltreenimine

Alapeatükid 2.1 ja 2.2 on kirjutatud Jacob Devlini, Ming-Wei Changi, Kenton Lee ja Kristina Toutanova poolt 2018. aastal avaldatud artikli *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (Devlin *et al.*, 2018) ning Pythoni paketi Transformers dokumentatsiooni (Wolf *et al.*, 2020) põhjal, kui just pole viidatud muule allikale.

Mudeli treenimisel eristatakse kahte osa - eeltreenimine (*pre-training*) ja kohandamine (*fine-tuning*). Eeltreenimisel õpetatakse mudelile kahte ülesannet - lünkteksti täitmine (*MLM, Masked-Language Modeling*) ning järgmise lause ennustamine (*NSP, Next Sentence Prediction*), mis on lahti seletatud allpool. Eeltreenimise eesmärk on õppida selgeks sõnede vektoresitused ning muud mudeli parameetrid, mille läbi tajub mudel keeles esinevate sõnede semantilist tähendust. Kohandamise käigus õpib mudel eeltreenitud mudeli väljundi põhjal lahendama spetsiifilist keeleülesannet, näiteks klassifitseerima tekste või leidma tekstist küsimuse põhjal vastuseid. Võrreldes eeltreenimisega kasutatakse mudeli kohandamiseks kordades väiksemaid andmestikke, kuna tänu eeltreeningule on mudelil juba teatav keeleline arusaam olemas. Seetõttu saab oluliselt vähendada ülesande lahendamiseks vajaminevat märgendatud treeningandmete mahtu, mis on oluline, kuna

märgendatud andmete loomine nõuab palju ressursse.

BERT mudeli eeltreenimine algab korpuse tekstide tokeniseerimisest, mille käigus mudeli sisend, milleks on sõna, lause, tekst või nende paar, tükeldatakse sõnedeks vastavalt mudeli sõnastikule. Tehnilises mõttes on sõne mistahes *UTF-8* kodeeringus märgijada. Kuna tokeniseerija toimetab *UTF-8* kodeeringuga, siis töökindluse tagamiseks tuleks ka sisend vajadusel eelnevalt ümber kodeerida.

Mudeli sõnastik sisaldab endas kõige sagedasemaid sõnu, sõnaosi ja ka üksikuid sümboleid valitud keelekorpusest. Kui mõnda tokeniseeritud sõne sõnastikus ei esine, märgitakse see tundmatuks sõneks sõnastiku erisõne [UNK] näol, mida aga tokeniseerija olemuse tõttu juhtub väga harva. BERT kasutab vaikimisi *Wordpiece* tokeniseerimist, mille loojad tõid välja, et optimaalne sõnede arv sõnastikus jääb 8000 ja 32000 sõne vahele (Wu *et al.*, 2016). Vaikimisi on BERT mudeli sõnastikus 30522 sõne, mis hõlmab endas 522 ühesümbolililst sõne ning 30000 sõnastiku loomisel lisatud sõne.

Wordpiece tokeniseerija alustab sõnastiku loomist sellest, et lisab sõnastikku kõik üksikud sümbolid, mis saadakse etteantud korpuse kõikide sõnade tükeldamisel. Sealjuures eristatakse sõne algustähte ja sõne muid sümboleid, lisades muudele sümbolitele eesliite '##' (sõna 'koer' tükeldatakse sõnedeks 'k', '##o', '##e' ja '##r'). Seejärel leitakse kõikidele võimalikele sõnastikus olemasolevatele sõnede paaridele

$$\text{skoor} = \frac{n_{ij}}{n_i \times n_j} \ ,$$

kus n_i ja n_j tähistavad vastavalt i-nda ja j-nda sõne sagedust korpuses ning n_{ij} tähistab i-ndast ja j-ndast sõnest koosneva paari sagedust korpuses. Tegemist on standardse üleesindatuse skooriga, kus lugejas on paari vaadeldud sagedus ning nimetajas on paari oodatav sagedus. Sõnastikku lisatakse sõne, mille skoor on suurim, ning protsessi korratakse kuni sõnede arv sõnastikus on jõudnud etteantud suuruseni. (*The Hugging Face Course* 2022)

Protsessi eesmärk on leida tekstikorpusele sõnastik, mis oleks võimalikult representatiivne keeles kasutatavate sõnade ja sõnaosade suhtes. Tihtiesinevad sõnad on sõnastikus esindatud täiskujul, harvemini esinevad sõnad on enamasti jaotatud loogilisteks osadeks, sagedasti sõna tüveks ning sõna lõpuks. Väga harva esinevad sõnad võivad jaotuda täielikult eraldiseisvateks sümboliteks, mis aga pole suur probleem, kuna neid sõnu esinebki keeles võrdlemisi vähe. Sõnastiku loomisel on sisuliselt tegemist Huffmani kodeerimisega (Huffman, 1952), selle erinevusega, et suund on erinev - Huffmani kodeerimisel ühendatakse esmalt kaks kõige väiksema sagedusega sümbolit.

Wordpiece tokeniseerija ülesehitus võimaldab toime tulla erinevate keelte iseärasustega. Eesti keeles muudab kääne enamasti sõna lõppu. Vaatleme korpuses esinevat sõna 'tigudega'. Kui korpuse tekstid on loodusteemalised, võib sõna 'tigudega' mahtuda ka mudeli sõnastikku. Koondkorpuse puhul on tegemist ilmselt väga harva esineva sõnaga, seega tuleb sõna tükeldada. Sõnaosa 'tigu' on sagedamini esinev sõna ning võib korpuses piisavalt palju esineda, et see mahuks mudeli sõnastikku - kui mitte, siis tükeldatakse sõna 'tigu' veel väiksemateks tükkideks. Sõnaosa 'dega' on aga tihti esinev käändelõpp, mis esineb paljudes sõnades: 'karpidega', 'lilledega', 'vanadega', ja nii edasi. Seega on sõne '##dega' suure tõenäosusega mudeli sõnastikus olemas ning sõna 'tigudega' tükeldatakse sõnedeks 'tigu' ning '##dega'. Nii säilivad sõna tüvi ning sõna kääne, mis annavad mudelis edasi sõna olemuse piisava täpsusega.

Sisendi tokeniseerimisel jaotatakse sisend esmalt tühikute ning punktuatsiooni sümbolite kohtadelt sõnedeks, kusjuures punktuatsioon lisatakse samuti tokeniseerija väljundisse, kuid tühikud mitte. Tokeniseerija on tundlik tühikute paigutuse suhtes, kuna ilma nendeta ei suuda tokeniseerija sisendi sõnesid omavahel eristada. Punktuatsiooni puudumine ei häiri tokeniseerija tööd, kuid see tähendab, et punktuatsiooni tähistavaid sõnesid ei leidu ka mudeli sisendis, mistõttu mudeli täpsus keeleülesannete lahendamisel võib kahaneda. Sealjuures on leitud, et ebaolulise punktuatsiooni nagu lauselõpumärkide puudumine BERT mudeli tööd ei mõjuta, kuid näiteks komade puudumine, mis kannavad laiemat informatsiooni nagu sõnade kokkukuulumine lauses teevad mudelit ebatäpsemaks (Ek, Bernardy ja Chatzikyriakidis, 2020).

Peale sõnede eraldamist vaadatakse iga sõne märgijadana eraldi - kui märgijada leidub tokeniseerija sõnastikus, leitakse sellele sõnastikust vastav indeks ehk järjekorranumber. Kui esialgset märgijada tokeniseerija sõnastikus ei leidu, otsitakse sõnastikust märgijada, kus esialgse märgijada lõpust on eemaldatud üks sümbol. Sümbolite eemaldamist märgijada lõpust korratakse kuni saadakse sõne, mis leidub mudeli sõnastikus. Seejärel vaadatakse sama loogika alusel läbi esialgsest märgijadast järelejäänud sümbolite jada ning jätkatakse protsessi kuni kõik esialgse märgijada sümbolid on käsitletud.

Näitlikustamiseks oletame, et sisendis leiduv sõna 'koerakuut' ei esine mudeli sõnastikus, kuid sõned 'koera' ja 'kuut' esinevad ning paiknevad sõnastikus kohtadel 123 ja 5555. Tokeniseerija üritab leida sõnastikust sõnesid 'koerakuut', 'koerakuu', 'koeraku' ja 'koerak', kuid vastet ei leita. Jõudes märgijadani 'koera', mis sõnastikus tõepoolest leidub, leitakse sellele sõnastikust indeks 123. Edasi vaadatakse allesjäänud sisendi märgijada 'kuut', millele leitakse sõnastikust vastena indeks 5555. Tulemusena on sõnast 'koerakuut' lisatud tokeniseerija väljundisse indeksid 123 ja 5555.

Lisaks sõnede tükeldamisele lisab tokeniseerija sisendile ka erisõned (*special tokens*):

[CLS] tähistab sisendi algust, [SEP] tähistab sisendi lõppu (paarissisendi korral nii esimese osa kui ka teise osa lõppu) ning [UNK] tähistab tundmatut sõne (nt. võõrkeelset tähte või sõna, mida sõnastiku loomisel korpuses ei esinenud).

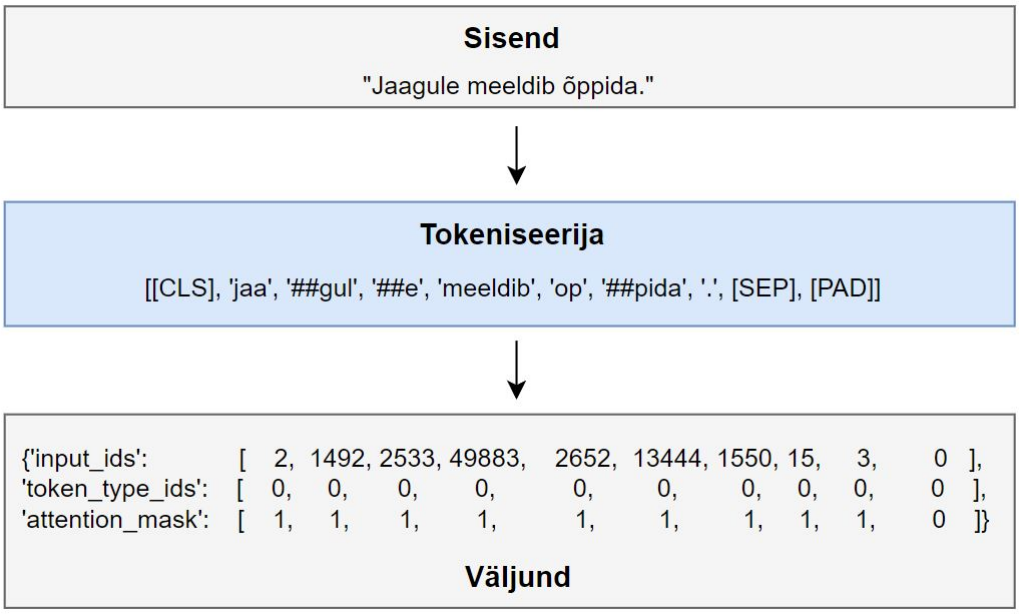
Kuna mudeli treenimisel on vajalik, et kõik sisendid oleksid sama pikkusega n, siis lühematele sisenditele lisatakse lõppu veel vajalikul arvul erisõnesid [PAD]. Kui mõni sisend on peale tokeniseerimist pikkusega k, kus $k > n$, lõigatakse sisendi lõpust nii palju sõnesid ära, et sisend oleks pikkusega n. Paarissisendi puhul kui sisendi esimene osa on pikkusega k_1 ning sisendi teine osa on pikkusega k_2, kus $(k_1 + k_2) > n$, siis eemaldatakse sõnesid sisendist ühekaupa. Sealjuures eemaldatakse sõne alati pikemast sisendi osast - kui $k_1 > k_2$, eemaldatakse sõne esimesest sisendi osast ning vastupidi, kui $k_2 > k_1$, eemaldatakse sõne teisest sisendi osast. Sõnesid eemaldatakse sisendist kuni $(k_1 + k_2) = n$. Eemaldatud sõnesid tokeniseerija väljundisse ei kaasata.

BERT mudeli esmasel loomisel treeniti kaks erineva pikkusega mudelit - BERT-BASE ja BERT-LARGE, kus sisendi pikkusteks olid vastavalt 128 ja 512 sõne. See tähendab muuhulgas seda, et kui hiljem rakendada mudelit konkreetse ülesande lahendamiseks ning sisendi pikkus on suurem kui n sõne, siis mudel väljastabki ennustuse vaid sisendi esimesele n sõnele. Praktikas võib olla oluline, et kogu sisend säiliks, näiteks nimeolemite märgendamisel, kui on tarvis saada ennustus kõikide teksti sõnade kohta. Sel juhul on üks võimalus jupitada sisendit lühemateks osadeks nii, et kõik sisendandmed oleksid tokeniseerimisel lühemad kui n sõne. Teine võimalus on võtta kasutusele pikemate sisendite jaoks loodud mudelid, näiteks nagu LONGFORMER, mis põhineb ROBERTA mudelil ja mille sisendi maksimaalseks pikkuseks on 4096 sõne (Beltagy, Peters ja Cohan, 2020).

Tokeniseerija väljund sisaldab endas kolme vektorit - tokeniseerija tulemusel leitud sõnede indekseid (*input IDs*) ehk järjekorranumbreid tokeniseerija sõnastikus, binaarset segmendi vektorit (*token type IDs*), mis määrab, kas sõne on paarissisendi esimeses või teises osas, ning binaarset tähelepanu vektorit (*attention mask*), mis eristab sõnesid ja *[PAD]* erisõneseid. Joonisel 4 on esitatud tokeniseerimine näitelause põhjal.

Tokeniseeritud sisendi igale sõnele leitakse seejärel esialgne staatiline vektoresitus (*embedding*), mis koosneb kolmest osast - sõne esitus (*token embedding*), segmendi esitus (*segment embedding*) ning positsiooni esitus (*position embedding*). Sõne esitus on unikaalne vektor igale sõnele mudeli sõnastikus. Segmendi esitus on vektor, mis on määratud vastavalt sellele, kas vaadeldav sõne on paarissisendi esimeses või teises osas - üheosalise sisendi puhul on see vektor iga sõne puhul sama. Positsiooni esitus on vektor, mis on määratud vastavalt sõne positsioonile sisendis.

Nii BERT mudelis kui ka WORD2VEC *continuous Bag-of-Words* mudeli vektoresituste

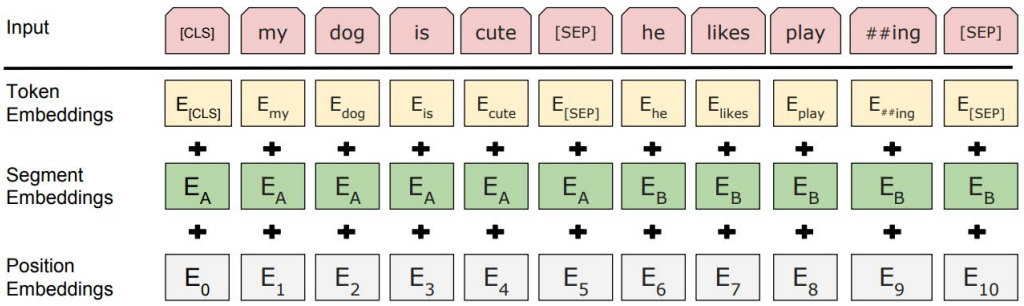


Joonis 4. BERT tokeniseerimine. Antud juhul on määratud mudeli sisendi pikkuseks 10, mistõttu lisatakse tokeniseerimisel üks [PAD] erisõne.

õppimine põhinevad sõnede ennustamisel teiste sisendi sõnede põhjal. Kui WORD2VEC metoodika vaatab vektoresituse õppimiseks vaid fikseeritud suurusega sõne ümbrust, üldjuhul 5 või 10 sõne mõlemale poole vaadeldavast sõnest, siis BERT mudel kasutab vektoresituste õppimiseks ära absoluutselt kõiki sisendi sõnesid läbi enesetähelepanu mehhanismi. Samuti suudab BERT mudel läbi positsiooni esituste leidmise arvestada sõna asukohaga lauses. Seeläbi kajastavad BERT mudeli esitused iga sõne semantilist tähendust täpsemini kui WORD2VEC metoodikal leitud esitused.

Kõik nimetatud esituste vektorid on määratud olema võrdse pikkusega, vaikimisi BERT-BASE mudelis 768 elementi ja BERT-LARGE mudelis 1024 elementi, mis on valitud vastavalt ingliskeelse mudeli treenimise empiirilistele tulemustele. Lõplik vektoresitus igale sõnele saadakse kolme esituse kokkuliitmisel. Joonisel 5 on näitena toodud kaheosalise sisendi vektoresituse leidmine, kus E_x tähistab vastavat esituse vektorit.

Positsiooni ja segmendi esituste lisamine on oluline, kuna transformeri arhitektuuriga mudelid ei käsitle sisendi sõnesid eraldi vaid kõiki sisendi sõnesid korraga, kasutades selleks juba mainitud enesetähelepanu (*self-attention*) mehhanismi. Seetõttu ei oleks ilma positsiooni ja segmendi esituste lisamiseta mudelil teavet selle kohta, kus konkreetne sõne sisendi suhtes paikneb. Sõna asukoht lauses või tekstis on aga keeleülesannete seisukohalt üsna oluline info.



Joonis 5. BERT vektoresituse leidmine (Devlin *et al.*, 2018).

Tokeniseeritud sisendi põhjal leitud vektoresitused on sisendiks mudeli kodeerijaosale, kus toimub mudeli eeltreenimine kahe ülesande põhjal.

NSP ülesanne. Mudeli eeltreenimisel on mudeli sisendiks lausete paar koos märgendustega: lause A ja lause B. Järgmise lause ennustamise õpetamiseks on laused valitud selliselt, et 50% juhtudel järgneb lause B korpuse tekstis lausele A (märge IsNext) ning ülejäänud 50% juhtudel on lause B suvaline korpuses esinev lause (märge NotNext). Mudeli ülesanne on ennustada, kas lause A järgneb lausele B või mitte.

NSP ülesannet on loomulikult võimalik lahendada ka lihtsamate meetoditega. Näiteks võib kasutada lineaarset klassifitseeriat, mis on defineeritud üle *one-hot-encodingu* abil leitud dokumendi sõnasageduste või Word2Vec esitusvektori. NSP ülesande osas puudub konsensus lihtsama baasmeetodi osas, millega BERT mudelit antud ülesande lahendamisel võrrelda saaks.

MLM ülesanne. Lünktesti ülesande õpetamiseks valitakse juhuslikult 15% tokeniseeritud sisendi sõnede positsioonidest. Valitud positsioonidel asuvad sõned maskeeritakse järgneva loogika alusel:

- 80% juhtudest asendatakse valitud sõne erisõnega [MASK]
- 10% juhtudest asendatakse valitud sõne juhusliku sõnega, mis on valitud ühtlase jaotusega mudeli sõnastiku sõnede hulgast
- 10% juhtudest jäetakse valitud sõne samaks

Mudeli ülesanne on igale maskeeritud sõnele ennustada sõne mudeli sõnastikust. See tähendab, et kui mudeli sõnastiku suurus on n, siis ennustab mudel igale maskeeritud sõnele indeksi vahemikust 0 kuni $(n-1)$, mis vastab ennustatava sõne indeksile mudeli

sõnastikus. Mudel kasutab ennustamiseks kogu sisendi konteksti, kus ennustatavad sõned on maskeeritud. Kui maskeerimisel kasutada 100% juhtudest [MASK] erisõne tekib ebakõla mudel eeltreenimise ning mudeli kohandamise vahel, sest kohandamisel ei ole mudeli ülesanne enam lünkteksti täitmine vaid näiteks teksti klassifitseerimine. See aga tähendab, et sisendteksti enam ei maskeerita ning mudeli sisendis pole ühtegi [MASK] erisõne, mis on võrreldes eeltreenimisega järsk muutus. Et selle muutuse mõju vähendada maskeeritakse 10% juhtudest sõne iseendaga ning 10% juhtudest sõne suvalise sõnega mudeli sõnastikust.

Maskeerimise osakaalud 80/10/10 valiti originaalse BERT mudeli arendamise kontrollkatsetuste tulemusel, kus vastav osakaalude jaotus andis keeleülesannete lahendamisel parimaid tulemusi. Sealjuures andsid näiteks maskeerimisstrateegiad 100/0/0 ja 0/0/100 pool protsenti kehvemaid tulemusi peale mudeli kohandamist nimeolemite märgendamise ülesandele. Kuna valik tehti ingliskeelsete tekstide põhjal, siis ei ole garaneeritud, et osakaalude jaotumine 80/10/10 on optimaalne ka teiste keelte jaoks.

BERT mudeli headust MLM ülesande lahendamisel võiks võrrelda WORD2VEC *continuous Bag-of-Words* meetodiga. BERT mudelil on lisaks ülaltoodud vektoresituse leidmisele ka muid eeliseid WORD2VEC metoodika ees, näiteks võivad BERT mudeli eeltreenimisel olla ka järjestikused ning lähestikku paiknevad sõned maskeeritud. Samas on WORD2VEC metoodika treenimine kiirem ning seega oleks selle näol tegemist hea võrdluse baasiga, millega teisi MLM ülesannet lahendavaid mudeleid saaks kõrvutada.

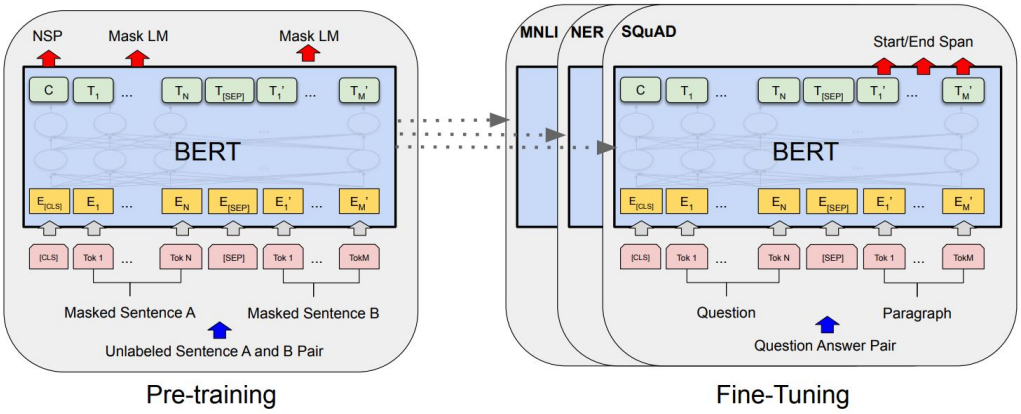
Originaalne BERT mudeli artikkel (Devlin *et al.*, 2018) kirjeldab maskeerimist staatilisena ehk maskeerimine tehakse vaid üks kord ning igal eeltreenimise epohhil kasutatakse samu maskeeritud sõnesid. ROBERTA mudeli artikkel (Liu *et al.*, 2019) aga toob ühe täiendusena välja dünaamilise maskeerimise, kus igal eeltreenimise epohhil tehakse uus valik sõnedest mida maskeeritakse. Dünaamiline maskeerimine teeb mudeli eeltreenimise stabiilsemaks, kuna vähendab juhusliku valiku mõju. BERT mudeli implementatsioonis Pythoni teegis Transformers kasutatakse ainult dünaamilist maskeerimist.

Eeltreenimisel peab mudel korraga ennustama nii seda, kas sisendina antud lausete paari teine lause järgneb päriselt esimesele ning ka kõiki maskeerimiseks valitud sõnesid, täpsemalt nende indeksit sõnastikus. Mõlema ülesande puhul arvutatakse eraldi kadu mudeli ennustuste ja tõeliste märgendite vahel kasutades ristentroopia (*cross-entropy*) kaofunktsiooni. Kogu mudeli kadu on kahe ülesande kadude summa ning selle alusel muudetakse mudeli kaale ja parameetreid läbi tagasilevi algoritmi. Mudeli täpsust saab hinnata nii treenimisandmete kao kui ka valideerimisandmete kao abil. Mudeli eeltreenimise implementatsiooniks loodud `Trainer` klassis tuleb sealjuures treeningandmestik ning valideerimisandmestik ise valida. Lisaks on võimalik valida erinevaid eeltreenimise

parameetreid nagu sammude/epohhide arv, ploki (*batch*) suurus või õpisamm (*learning rate*).

2.2 Mudeli kohandamine

BERT mudeli kohandamisel võetakse aluseks eeltreenitud BERT mudel ning treenitakse seda täiendavalt ülesandespetsiifilisel andmestikul. Lisaks mudeli erinevale treeningülesandele peitub eeltreenimise ja kohandamise vahe ka andmete mahus. Kohandamine on võrreldes eeltreenimisega vähem mahukam ning kiirem protsess. Kui ESTBERT mudeli eeltreenimiseks kasutati andmestikku 1,1 miljardi sõnega, siis näiteks ESTBERT mudeli kohandamisel nimeolemite märgendamiseks kasutati treeningandmestikku, milles oli 156 tuhat sõne, mida on võrreldes eeltreenimise andmetega ligi 7000 korda vähem (Tanvir *et al.*, 2020). Mudeli eeltreenimise ja kohandamise erinevus küsimustele vastamise ülesande näitel on näidatud joonisel 6.

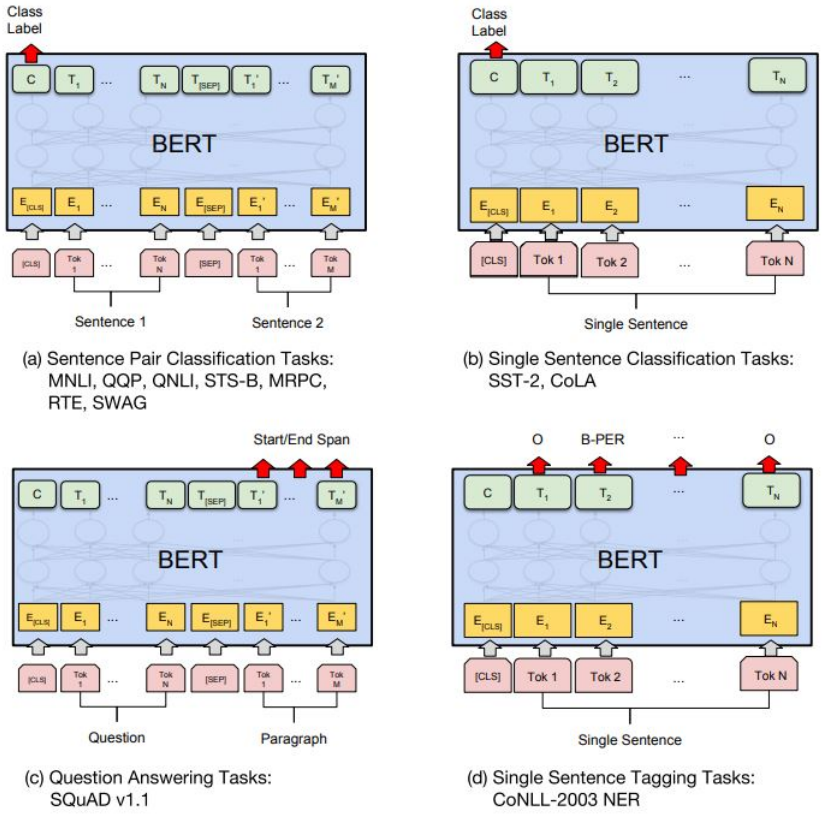


Joonis 6. BERT eeltreenimine ja kohandamine (Devlin *et al.*, 2018)

Erinevate keeleülesannete jaoks muudetakse mudeli pead (*head*), et see vastaks ülesande olemusele. Eeltreenitud mudelil on samuti pea, mis ongi mõeldud MLM ja NSP ülesanneteks, kus mudeli kodeerijaosa väljundi põhjal leitakse ennustused mõlemale ülesandele. Kodeerijaosa väljund sisaldab iga sisendi tokeniseeritud sõne kohta vektorit, mille suurus on määratud mudeli peidetud kihi suurusega. BERT-BASE mudelis on peidetud kihi suurus 768.

Näiteks ülesannete jaoks nagu nimeolemite või sõnaliikide märgendamine, kus iga sõna kohta peab mudel tegema ennustuse, on olemas vastav *BertForTokenClassification* mudel, mille pea suunab kodeerijaosa väljundi läbi lineaarkihi. Lineaarkiht muudab kodeerijaosa

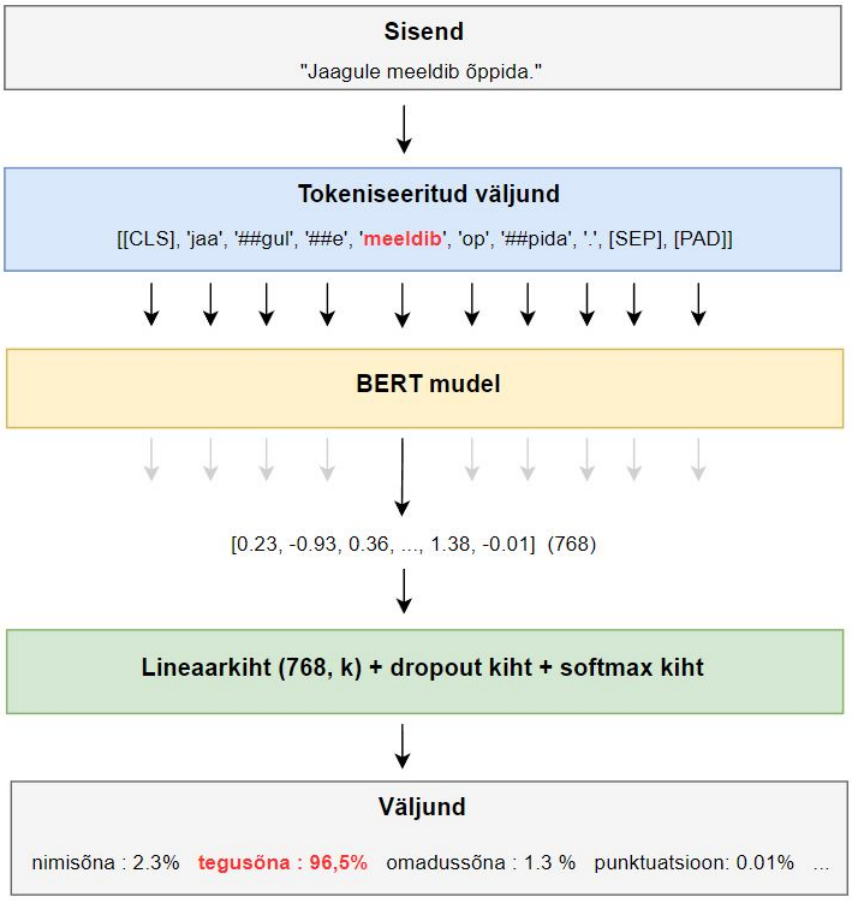
väljundi iga sõne k elemendilise vektori l-pikkuseks vektoriks, kus k on mudeli peidetud kihi suurus ning l on ennustatavate klasside arv. Mudeli kohandamisel kasutatatakse taaskord ristentroopia kadu, mis leitakse lineaarkihi väljundi ning tõeliste märgenduste vahel. Peale mudeli kohandamist saab mudelit kasutada ennustamiseks, suunates tokeniseeritud sisendi läbi mudeli ning leides lineaarse kihi väljundite hulgast maksimaalse väärtuse (*hard classification*. Teine võimalus on lineaarkihi väljundid suunata läbi *softmax*-kihi, et leida igale sõnele tõenäosus igasse klassi kuulumise kohta (*soft classification*).



Joonis 7. BERT kohandamine erinevatele ülesannetele (Devlin *et al.*, 2018). (a) Kaheosalise sisendi klassifitseerimise ülesanded, (b) Üheosalise sisendi klassifitseerimise ülesanded, (c) Küsimustele vastamise ülesanded, (d) Üheosalise sisendi sõnede märgendamise ülesanded.

Kui vaadelda üledandeid nagu rubriigi klassifitseerimine või spämmi tuvastamine, kus iga sõna kohta pole ennustust vaja, vaid kogu teksti kohta on vaja ühte ennustust, siis selleks kasutatakse kodeerijaosa väljundi esimese sõne ehk erisõne [CLS] väljundit. Vastav mudel selle jaoks on *BertForSequenceClassification*, mille pea kasutab klasside ennustamiseks samuti lineaarkihti. Kuna BERT mudeli lähtekood on vabalt kättesaadav,

saab mudeli pea ka ise disainida ja mudeli kohandamiseks kasutada, kus muuhulgas saab muuta mudeli pea kihte ning kohandamise parameetreid. Erinevatele ülesannetele kohandatud mudelid on esitatud joonisel 7 ning kohandatud mudeli ennustamise näide joonisel 8.



Joonis 8. BERT kohandatud mudel lauseosa märgendajana. Sõne 'meeldib' leidub mudeli sõnastikus ning seega esineb ka peale tokeniseerimist sõnena 'meeldib'. BERT-BASE mudelis leitakse sellele 768 elemendiline vektor, mis saadetakse läbi lineaarkihi, *dropout* kihi ning *softmax* kihi. Selle tulemusena leitakse sõnele 'meeldib' tõenäosus igasse klassi kuulumise kohta, antud juhul tuleb suurim tõenäosus klassile 'tegusõna'. Joonisel tähistab k ennustatavate klasside arvu ehk erinevaid võimalikke lauseosasid, mida mudel saab ennustada.

3 Täiendatud mudel

Selles peatükis kirjeldatakse BERT mudelile tehtud täiendusi. Täienduste põhiidee on rikastada mudeli sisendit morfoloogilise analüüsi abil kasutades selleks Pythoni teegi EstNLTK (Laur *et al.*, 2020) võimalusi. Kuna täienduste tulemusena muutub mudeli tokeniseerija väljundi kuju, tuleb kõik ülejäänud mudeli osad viia sellega vastavusse. Täiendatud mudeli arendamiseks ja eeltreenimiseks kasutatakse korpusena 2017. aasta eesti keele ühendkorpust (Koppel ja Kallas, 2018), mida kasutati ka ESTBERT mudeli puhul (Tanvir *et al.*, 2020). Implementeerimiseks tehakse koopia Pythoni teegi Transformers Githubi repositooriumist (*Transformers* 2023) ning muudetakse sealseid faile. Selline lähenemine on tarvilik teegi keerukuse tõttu ning tulevikuarendusena tuleks selguse huvides kasutada originaalset Transformers teeki defineerides üle vaid vajalikud klassid ja failid.

Tokeniseerimine. Tokeniseerija muutmise motivatsioon tuleneb asjaolust, et BERT mudel kasutab sõnede esituste õppimisel ka keeles väga harva esinevaid sõnu, mis ei ole ilmselt statistiliselt efektiivne ning võib mõjutada ülejäänud sõnede semantilise tähenduse õppimist. Sõne tähendus keeles ei sõltu sõne käändest - 'hobune', 'hobustele' ja 'hobuseta' viitavad kõik samale olendile. Siit ka hüpotees: eesti keele kui grammatiliselt mahuka keele jaoks on tarvilikum käsitleda sõnede algvormi ning käänet või pööret keelemudelis eraldi komponentidena.

Teisalt töötab *Wordpiece* tokeniseerija mõnes mõttes täiendatud tokeniseerijale sarnaselt - kuna keeles tihedamini esinevad sõnaosad kaastakse mudeli sõnastikku, siis sisaldub sõnastikus suur osa sagedatest sõnajuurtest ning ka käände- ja pöördelõppudest. Seega alternatiivne hüpotees on, et *Wordpiece* tokeniseerija juba arvestab eesti keele grammatiliste nüansidega ning uus tokeniseerija ei muuda mudelit paremaks. Hüpoteesi kontrollimiseks tuleks võrrelda erineva tokeniseerijaga mudelite suutlikkust keeleülesannetel.

Täiendatud mudeli tokeniseerija muudab esmalt sisendi teksti EstNLTK `Text` objektiks ning lisab sellele morfoloogilise analüüsi kihi. EstNLTK `Text` objekti loomisel lisatakse sisendile lausete, sõnede, liitsõnede ning sõnade kihid. Morfoloogilise analüüsi kiht leiab iga sõnade kihis oleva sõna kohta muuhulgas ka sõna lemma ehk algvormi ja sõna vormikoodi ehk sõna käände või pöörde tähise. Iga sisendi sõna lemmast ja vormist moodustatud paar lisatakse tokeniseerija väljundisse. See tähendab, et võrreldes BERT mudeliga, mille tokeniseerija väljundis on ühemõõtmeline järjend sõnede indeksitest, on uue mudeli tokeniseerija väljundis kaheelemendiliste ennikute järjend, kus iga enniku esimene liige viitab sõne lemmale ning enniku teine liige sõne vormile. Uue tokeniseerija tööpõhimõtet seletatakse näite põhjal joonisel 9.

Kui originaalne BERT mudel kasutab *Wordpiece* tokeniseerijat, mille sõnastiku sõned on oma olemuselt sõnad, sõnaosad või üksikud sümbolid, siis täiendatud mudelil on kaks sõnastikku - üks lemmade ning teine vormide jaoks. Vormide sõnastik on kõikehõlmav, kuna erinevaid võimalikke vormikoode on kokku vaid 106[1]. Lemmade sõnastik sisaldab endas korpuse 50000 kõige sagedasemat lemmat, mis on suuruselt võrdväärne EstBERT mudeli sõnastikuga. Lemmade sõnastiku loomisel filtreeriti korpusest välja tekstid, mis ei olnud eestikeelsed. Mõlemad sõnastikud sisaldavad endas ka erisõnesid [PAD], [UNK], [CLS], [SEP] ja ˇMASKˇ. Maskeerimissõne ei kasuta kantsulgusid eesmärgil, et EstNLTK vahendid käsitleksid seda ühe sõnena, st kui maskeerimissõne oleks [MASK], tükeldataks see `Text` objekti loomisel sõnedeks '[', 'MASK' ja ']'.

EstNLTK morfoloogiline analüüs väljastab mõningatel juhtudel sõnale mitu lemmat ja vormi - sellisel juhul leiab uus tokeniseerija juhusliku indeksi ning valib sõna morfoloogilisest analüüsist vastava indeksiga lemma ja vormi. Lisaks käsitleb tokeniseerija eraldi liitsõnu, mille lemmasid ei leidu mudeli sõnastikus. See tähendab, et sagedasti esinevate liitsõnade lemmad on mudeli sõnastikus olemas, kuid keeles harvemini esinevate liitsõnade korral otsitakse sõnastikust liitsõna lemma asemel liitsõna juurte lemmasid. Liitsõna juured on samuti EstNLTK morfoloogilise analüüsi kihis olemas. Kui tokeniseerija ei leia juure lemmat mudeli sõnastikust, tähistatakse see tokeniseerija väljundis erisõnega [UNK]. Liitsõnade indikeerimiseks leiab tokeniseerija iga väljundi sõne kohta ka kaheksaelemendilise binaarse vektori, mille iga positsioon tähistab kanalit sõne omaduse edasikandmiseks. Mudeli implementatsioonis väärtustatakse vektori elemente vastavalt tabelile 3.

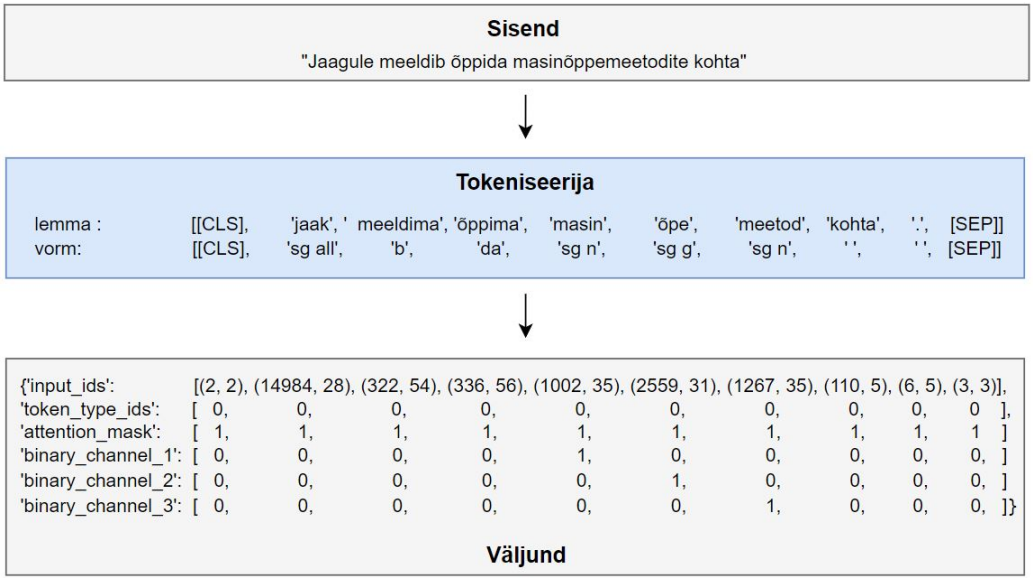
Tabel 3. Binaarsete kanalite väärtused täiendatud mudelis.

Positsioon	**Väärtus**
1	1, kui sõne on liitsõna esimene juur, mujal 0
2	1, kui sõne on mõni liitsõna keskmistest juurtest, mujal 0
3	1, kui sõne on liitsõna viimane juur, mujal 0
4-8	alati 0, reserveeritud tulevikuarenduste jaoks

Vaatleme lemmat 'raudteejaam'. Kui see leidub mudeli lemmade sõnastikus, on sõnale vastav binaarne vektor (0,0,0,0,0,0,0,0). Kui lemmat mudeli lemmade sõnastikus ei leidu, vaatab tokeniseerija eraldi juursõnesid 'raud', 'tee' ja 'jaam'. Oletame, et need kolm sõna leiduvad mudeli sõnastikus. Sellisel juhul lisab tokeniseerija mudeli väljundisse kolm ennikut millele vastavad binaarsed vektorid on 'raud' – (1,0,0,0,0,0,0,0), 'tee' –

[1]Vormikoodide tähendused on esitatud lehel `https://cl.ut.ee/ressursid/morfo-systeemid/`

(0,1,0,0,0,0,0,0) ning ’jaam’ – (0,0,1,0,0,0,0,0).



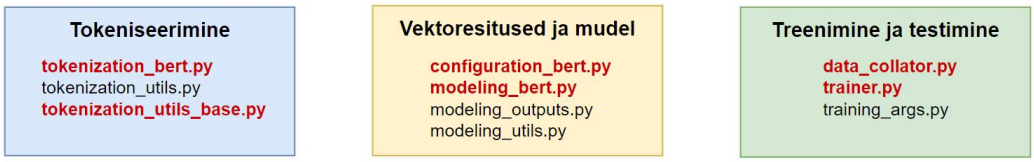
Joonis 9. Täiendatud tokeniseerija. Iga sõna kohta leitakse EstNLTK vahenditega lemma ning vorm. Sõnastikus ei esine liitsõna ’masinõppemeetodid’ lemmat, küll aga esinevad sõnastikus liitsõna juurte lemmad - ’masin’, ’õpe’ ja ’meetod’.

Töö koodis on implementeeritud tokeniseerija põhilised muudatused järgnevad:

1. fail `bert_tokenization.py`
   (a) funktsioon `_tokenize` - tokeniseerimise loogika
   (b) funktsioon `_convert_token_to_id` - sõnede ning sõnastiku indeksite vaheline teisendus
   (c) funktsioon `_convert_id_to_token` - sõnastiku indeksite ja sõnede vaheline teisendus
2. fail `tokenization_utils_base.py`
   (a) funktsioon `prepare_for_model` - tokeniseerija väljundi kujuga ning binaarsete kanalitega arvestamine
   (b) funktsioon `_pad` - erisõne [PAD] lisamise loogika kohandamine uuele tokeniseerijale

Binaarsete kanalite 4-8 aktiveerimiseks tuleb töö koodi faili `bert_tokenization.py` funktsioonis `_tokenize` tekitada kanalitele väärtustamise loogika ning lisada kanalitele

esitused faili `modeling_bert.py` `BertEmbeddings` klassis analoogselt olemasolevatele kanalitele 1-3. Joonisel 10 on esitatud mudeli kolm peamist moodulit koos põhiliste failidega, kus vastavad moodulid implementeeritakse.

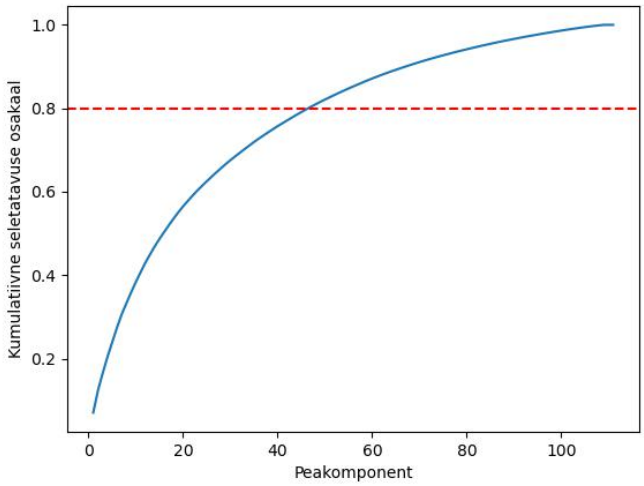


Joonis 10. Mudeli põhilised moodulid ning neid implementeerivad failid. Punasega on märgitud failid, milles tehti täiendatud mudeli implementeerimisel muudatusi.

Vektoresitus. BERT mudel leiab igale tokeniseerija väljastatud sõnele vektoresituse sõne esituse, segmendi esituse ning positsiooni esituse liitmisel. Täiendatud mudeli tokeniseerija väljund sisaldab sõne asemel sõne lemmat ja vormikoodi, seega on ka vektoresituse leidmisel tehtud varasemast sõne esitusest kaks komponenti - lemma esitus ning vormi esitus. Kui BERT-BASE mudelis on kõikide esituste pikkus fikseeritud 768 elemendini, siis täiendatud mudelis on lemma esituse pikkus 720 elementi ning vormi esituse pikkus 48 elementi. Lemma esitus ning vormi esitus kleebitakse omavahel kokku, saades 768 elemendi pikkune vektor.

Vormi esituse pikkus valiti kasutades eeltreenitud mudeli vormiesitustel peakomponentide analüüsi. Analüüsitavas mudelis oli nii lemma esituse kui ka vormi esituse pikkus 768 elementi ning need esitused liideti koos segmendi esituse ning positsiooni esitusega, et saada lõplik sõne esitus. Mudelit eeltreeniti 8 päeva ühe graafikaprotsessori peal, mille jooksul tehti 300 000 treenimise sammu. Eeltreenitud mudeli vormiesituste peakomponentanalüüsil selgus, et 46 peakomponenti seletavad ära 80% vormiesituste variatsioonist, mida on visualiseeritud joonisel 11.

Peale lõpliku vektoresituse leidmist liigub see mudeli enesetähelepanu mehhanismi, kus vektoresitus jaotatakse võrdselt enesetähelepanu peade vahel ära. Mitmepealine ülesehitus võimaldab enesetähelepanu mehhanismil õppida erinevat tüüpi seoseid sisendi sõnade vahel. Näiteks mõni enesetähelepanu pea võib kajastada kõrvuti asetsevate sõnade vahelisi seoseid ning mõni teine pea sisendi eri otstes asetsevate sõnade vahelisi seoseid, osad pead võivad keskenduda semantilistele seostele ning teised pead süntaktilistele seostele sisendi sõnade vahel. Peade arvu suurendamisel suureneb ka mudeli õpitavate parameetrite arvu, seega on oluline enesetähelepanu peade arv optimiseerida. BERT-BASE mudelis on enesetähelepanu peade arv vaikimisi 12.



Joonis 11. Peakomponentide analüüsi tulemus. Graafikul on sinise joonega toodud peakomponentide kumulatiivne vormiesituste variatsiooni seletatavuse osakaal ning punasega 80% osakaalu piir.

Esiaglses implementatsioonis hoiti lemma esituse pikkus 768 elementi ning sellele kleebiti otsa vormi esitus. Kuna vektoresituse kogupikkus peab mudeli töötamiseks olema mudeli enesetähelepanu peade arvu ehk 12-kordne, siis valiti vormi esituse pikkuseks 48. Mudeli lõplikus implementatsioonis fikseeriti vektoresituse kogupikkuseks 768 ning seega ei ole enam vajalik, et vormi esitus oleks enesetähelepanu peade arvu kordne, sest suurendades vormi esituse pikkust lüheneb selle arvelt lemma esituse pikkus.

Lisaks segmendi ja positsiooni esitustele õpitakse täiendatud mudelis ka iga lisatud binaarse kanali kohta ära esitus ning liidetakse need teiste esitustega. Implementeeritud mudelis kasutati kolme binaarset kanalit, seega lõplik vektoresitus sõnele saadakse kokkukleebitud lemma ja vormi esituse, segmendi esituse, positsiooni esituse ning kolme binaarse kanali esituste kokkuliitmisel.

Töö koodis on vektoresitustega tehtud muudatused implementeeritud klassis `BertEmbeddings` failis `modelling_bert.py`. Kõik ülejäänud muudatused erinevatele mudeli osadele ja failidele on tehtud selleks, et mudeli implementatsioon suudaks arvestada tokeniseerijale ning vektoresitustele tehtud muudatustega. Sealjuures on suurimaks muudatuste põhjustajaks tokeniseerija väljundi uus kuju, millega tuleb arvestada nii vektoresituste loomisel kui ka mudeli treenimisel. Täiendatud mudeli lõplik implementatsioon on avalikult kättesaadav töö Githubi repositooriumis [2].

[2] `https://github.com/raulniit/transformers`

4 Täiendatud mudeli treenimine ja rakendamine

Täiendatud mudelit eeltreeniti Tartu Ülikooli HPC (*High Performance Computing*) keskuses (University of Tartu, 2018) ning selleks kasutati ühte NVIDIA Tesla V100 graafikaprotsessorit. Eeltreenimine toimus 8-päevalise ajalimiidi tõttu kahes osas, kokku 16 päeva, mille jooksul jõudis mudel treenida 500 000 sammu. Mõju hindamiseks salvestati mudeli seis iga 50 000 sammu tagant. Eeltreenimiseks kasutati ainult MLM ülesannet, sest RoBERTa mudeli analüüsimisel selgus, et NSP ülesanne ei mõjuta mudeli tulemusi (Liu *et al.*, 2019). Mudeli esialgne konfiguratsioon on sama, mis orginiaalsel BERT-BASE mudelil, välja arvatud mudeli sõnastiku suurus.

Kuna tokeniseerimine ise on samuti ajamahukas protsess tokeniseeriti enne mudeli treenimist esmalt ära kõik korpuse tekstid ning kirjutati need JSON kujul failidesse. Sealjuures on tokeniseeritud andmete maht oluliselt suurem, kui tavalisel BERT mudelil, sest uue tokeniseerija väljund sisaldab varasema ühe sõne indeksi asemel iga sõne kohta lemma ning vormi indeksit. Lisaks suurendab andmete mahtu binaarsete kanalite olemasolu tokeniseerija väljundis. Kokku on tokeniseeritud korpuse maht 374 gigabaiti.

Mudelite kohandamiseks kasutati mugandatud Transformers teegi näitekoode. Näitekoodides kastutatakse BERT mudelitele loodud kiiremaid, '*fast*' versioone tokeniseerijatest, mis põhinevad programmeerimiskeelel Rust. Kiired tokeniseerijad võimaldavad muuhulgas tokeniseeritud väljundi põhjal leida, millistest sõnadest sisendi tekstis vastavad sõned tokeniseerija väljundisse tekkisid. See on oluline just märgendamisülesannete korral, kus on vaja viia treeningandmestiku märgendused vastavusse tokeniseerija väljundiga. Täiendatud tokeniseerijal selline omadus puudub. Seetõttu tuli iga sisendi tokeniseerimisel tokeniseerida ka sisendi iga sõna eraldi ning hoida meeles, millised sõned iga sõna tokeniseerimisel tekivad.

Stabiilsuse tagamiseks tuli välja lülitada ka täiendatud tokeniseerija juhuslik ühestamine ehk kui EstNLTK analüüs leidis sõnale mitu lemmat, valiti nende hulgast esimene. Võrdluseks kohandati samade koodide ning andmestikega ka EstBERT-BASE mudel. Eeltreenitud mudeli kaalud pole kohandamisel jäigalt fikseeritud, mistõttu muudetakse kohandamise protsessis kõiki mudeli kaale, mitte ainult lisatud mudeli pea kaale.

4.1 Keeleülesanded ning andmestikud

Mudelite hindamiseks kasutatakse nelja keeleülesannet: MLM ülesanne, rubriigi klassifitseerimine, lauseosade märgendamine ning nimeolemite märgendamine. MLM ülesanne oli kasutusel ka mudeli eeltreenimisel ning selle abil võrreldakse täiendatud mudeli vaheetappe st viit täiendatud mudelit, mille eeltreenimsel on tehtud $i \times 100000$ sammu, kus

$i \in \{1, 2, 3, 4, 5\}$. Kolme ülejäänud keeleülesande kohandamisel järgitakse ESTBERT mudeli kohandamise protsessi ning parameetreid - kõikide ülesannete puhul on mudeli treenimisel epohhide arv 3 ning õpisamm 5e-5. Rubriigi klassifitseerimisel ning MLM ülesandel on ploki suurus 16, lauseosade ning nimeolemite märgendamisel 8.

Rubriigi klassifitseerimiseks kasutatakse Eesti Keele Instituudi valentsikorpust (Pajupuu, Altrov ja Pajupuu, 2016). Korpus sisaldab endas 3848 ajalehe Postimees teksti, kus iga tekst kuulub ühte üheksast rubriigist. Andmestik jagatakse osakaaludega 70/10/20 treening-, valideerimis ning testandmestikuks, mille käigus säilitatakse rubriikide jaotumise osakaal alamandmestikes. Täpsem rubriikide jaotus korpuses on esitatud tabelis 4. Mudeli kohandamisel õpib mudel ennustama sisendteksti rubriiki. Sama andmestikku kasutatakse ka mudeli MLM ülesandele kohandamiseks, selle erinevusega, et rubriikide jaotumise osakaal alamandmestikes pole fikseeritud.

Tabel 4. Rubriikide ehk klasside jaotus treening-, valideerimis- ning testandmestikus.

Rubriik	**Treen**	**Val**	**Test**	**Kokku**
Arvamus	680	98	194	972
Kommentaar-Eesti	341	49	97	487
Kommentaar-Elu	336	48	96	480
Eesti	289	41	83	413
Sport	268	38	77	383
Elu	230	33	66	329
Välismaa	219	31	63	313
Kultuur	184	26	52	262
Krimi	146	21	42	209
Kokku	2693	385	770	3848

Lauseosade märgendamiseks kasutatakse Eesti keele EDT korpust *Universal dependencies (UD)* versioon 2.5 kogust (Zeman, 2019). Andmestik sisaldab lauseid, kus iga sõna kohta on muuhulgas esitatud lauseosa märgend (*UPOS*)[3]. Kokku on erinevaid võimalikke lauseosa märgendeid 16 ning nende jaotumine andmestikes on esitatud tabelis 5. Andmestikud on CoNLL-U formaadis ning on juba korpuses jaotatud treening-, valideerimis- ning testandmeteks. Mudeli kohandamisel ennustab mudel iga sisendtekstis esineva sõna lauseosa.

[3]Lauseosa märgendite tähendused on esitatud lehel `https://universaldependencies.org/u/pos/`

Tabel 5. Lauseosa märgendite jaotus treening-, valideerimis- ning testandmestikus.

Lauseosa märgend	**Treen**	**Val**	**Test**
ADJ	28713	3992	4070
ADP	7453	887	914
ADV	33003	4194	4861
AUX	17432	2247	2575
CCONJ	12422	1681	2031
DET	5398	675	812
INTJ	251	20	51
NOUN	91185	11655	12616
NUM	7174	1080	848
PRON	18025	2469	2524
PROPN	20927	2539	3061
PUNCT	56377	7499	7674
SCONJ	6752	859	1116
SYM	540	119	26
VERB	38394	4832	5254
X	857	61	100
Kokku	344903	44809	48533

Nimeolemite märgendamiseks kasutatakse Eesti nimeolemite korpust (Tkachenko, Petmanson ja Laur, 2013), mis sisaldab endas 572 Postimehe ja Delfi uudise artiklite tekste. Kokku sisaldab andmestik 217253 märgendatud sõne. Andmestik jaotatakse osakaaludega 70/10/20 treening-, valideerimis ning testandmestikuks. Nimeolemite märgendite jaotumine andmestike vahel on esitatud tabelis 6.

Tabel 6. Nimeolemite märgendite jaotus treening-, valideerimis- ning testandmestikus.

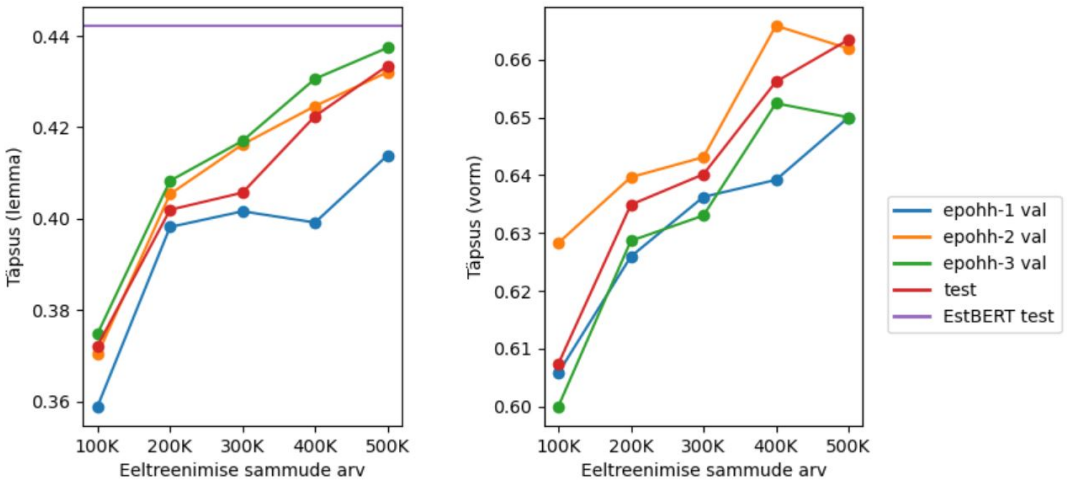
Nimeolemi märgend	**Treen**	**Val**	**Test**	**Kokku**
LOC - asukoht	4480	653	1178	6311
ORG - organisatsioon	4416	575	1154	6145
PER - isikunimi	5925	836	1729	8490
Kokku	14821	2064	4061	20946

4.2 Tulemused ja analüüs

Käesolevas alapeatükis esitatakse mudeli kohandamise tulemused joonistel. Detailsemad kohandamise tulemused on esitatud töö lisas I.

Eeltreenimisel peab mudel lahendama MLM ülesannet, seega teoreetiliselt võiks mudel ilma täiendava kohandamiseta seda juba üsna hästi osata. Samas on täiendav kohandamine asjakohane, kuna testitimiseks kasutatav andmestik sisaldab endas vaid ajakirjandustekste, kus keelekasutus erineb oluliselt näiteks netifoorumite tekstidest. MLM ülesande tulemuste jooniselt 12 on näha, et nii lemmade kui ka vormide ennustamise täpsus suureneb iga täiendava 100 000 eeltreenimise sammu tegemisel. Sealjuures ei näi mudel veel koondunud olevat, mis annab alust arvata, et mudelit tuleks edasi eeltreenida.

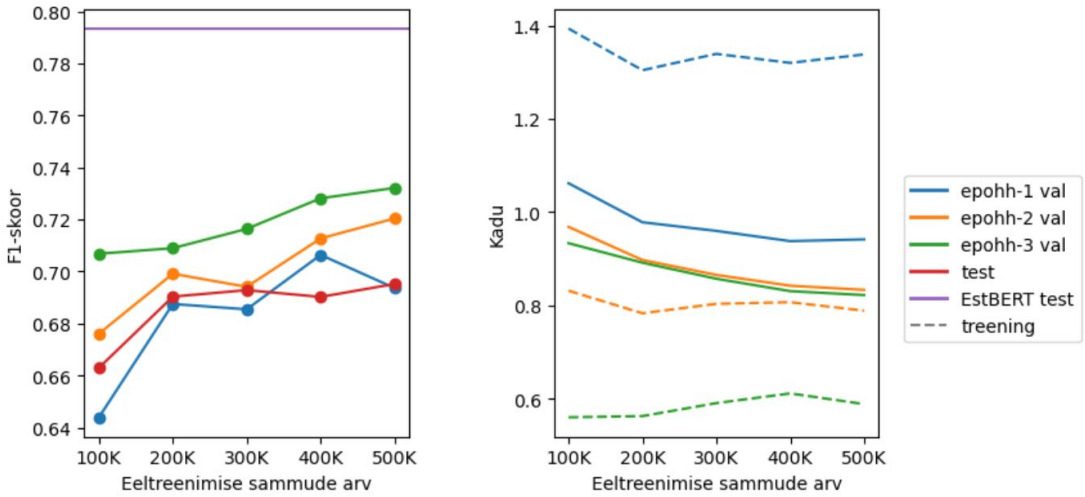
Parima mudeli ehk 500 000 sammu eeltreenitud mudeli korral on MLM ülesande täpsus testandmestikul lemmade ennustamisel 0,433 ning vormide ennustamisel 0,633. Vormide ennustamise täpsus on ootuspäraselt kõrgem lemmade ennustamise täpsusest, sest vorme on võrreldes lemmadega mudeli sõnastikes oluliselt vähem. Lemmade ennustamise täpsus on väga lähedal EstBERT mudeli MLM ülesande ennustustäpsusele, mis on testandmestiku puhul 0,442. Sealjuures tuleb meeles pidada, et EstBERT mudel ennustab MLM ülesandes puuduva lemma ja vormi asemel puuduvat sõne.



Joonis 12. MLM ülesande tulemused. EstBERT testandmestiku täpsus on orginaalse BERT mudeli MLM ülesande jaoks kohandatud, kus ennustama peab lüngas olevat sõne, mitte lemmat ja vormi.

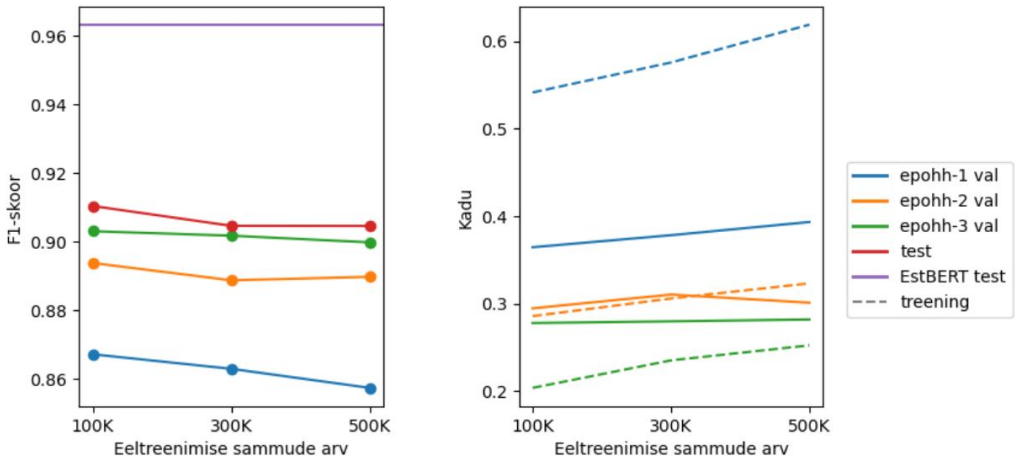
Rubriigi klassifitseerimise ülesande juures on seevastu märgata, et täiendav treenimine oluliselt mudeli tulemusi ei paranda. Jooniselt 13 on näha, et testandmestikul annavad

kõik mudelid, mida on eeltreenitud vähemalt 200 000 sammu, sarnase f1-skoori. Sealjuures jääb täiendatud mudelite f1-skoor testandmestikul vahemikku 0,69 - 0,70, mis on oluliselt madalam EstBERT mudeli f1-skoorist 0,79. Samas näitavad parempoolse joonise valideerimis- ning treenimiskaod, et antud andmestikule on kõige optimaalsel kohandada mudelit kaks epohhi. Kolme epohhi juures on mudeli kadu valideerimisandmetel märgatavalt suurem treenimisandmete kaost, mis viitab mudeli ülesobitamisele.



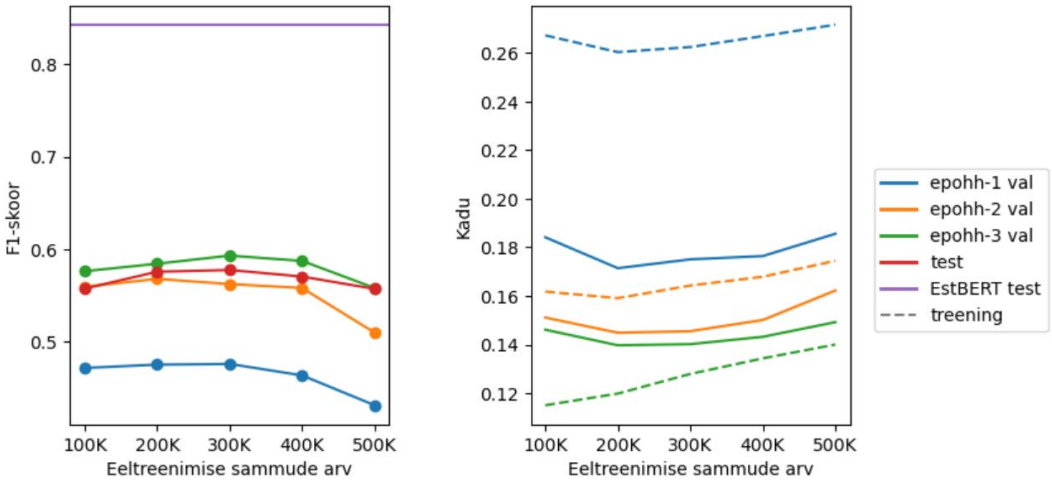
Joonis 13. Rubriigi klassifitseerimise tulemused.

Lauseosa märgendamise tulemustest jooniselt 14 on samuti ilmne, et täiendav treenimine ei anna võitu ülesande lahendamise tulemustes. Erinevus EstBERT mudeliga ei ole küll suur, ligi 4%, kuid see võib olla tingitud üleüldisest kõrgest f1-skoorist ülesandel. Mudeli kohandamiseks paistab sobivat kaks või kolm epohhi, kus valideerimise kadu ning kohandamise kadu on sarnased. Huvitaval kombel on testandmestikul kõrgeim f1-skoor 100 000 sammu eeltreenitud mudeli korral ning täiendav eeltreenimine tulemust ei suurenda.



Joonis 14. Lauseosa märgendamise tulemused.

Nimeolemite märgendamise ülesande puhul on joonise 15 põhjal valitud andmestiku jaoks kolmel epohhil treenimine sobiv, kuna kolmandal epohhil paiknevad valideerimiskadu ning treenimiskadu veel üsna lähestikku ning f1-skoor on kõrgem võrreldes kahel või ühel epohhil treenimsisega. Samas tuleb ka selle ülesande puhul välja, et täiendav eeltreenimine ei paranda täiendatud mudeli tulemust. Erinevus testandmetel on võrreldes EstBERT mudeliga väga suur - kui täiendava mudeliga tuleb parim f1-skoor 0,57, siis EstBERT mudeli f1-skoor testandmetel on 0,84.



Joonis 15. Nimeolemite märgendamise tulemused.

Kohandamise ülesannete tulemustest saab järeldada, et täiendav eeltreenimine annaks vaadeldud neljas ülesandes paremaid tulemusi vaid MLM ülesande lahendamisel - kolme

ülejäänud kohandatud ülesande puhul ei andnud täiendavalt eeltreenitud mudelid enam testandmestikul paremaid tulemusi. Mudeli võimekus on ESTBERT mudeliga võrreldav samuti vaid MLM ülesande puhul, teiste ülesannete korral on täiendatud mudeli tulemused oluliselt madalamad. See annab alust arvata, et täiendatud mudeli suutlikkust erinevatel keeleülesannetel saab ligikaudu hinnata juba peale 100 000 sammu eeltreenimist. See on oluline teadmine mudeli edasisel arendamisel, kuna mudeli täiendav eeltreenimine võtab märgatavalt rohkem aega. Samas viitavad tulemused, et täiendatud mudeli puhul on murekoht keeleülesannete lahendamisel muus kui väheses eeltreenimises.

Üheks ilmseks probleemiks täiendatud mudelile on tokeniseerijale tundmatute sõnede suur osakaal - rubriigi klassifitseerimise andmestiku puhul on see 6,5% kõikidest sõnedest, lauseosa märgendamise andmestikus 5,6% ning nimeolemite märgendamise andmestikus 7,1%. Tundamatule sõnele on mudelil raske anda täpset ennustust, kuna ainsaks sisuliseks infoks on mudelile sõne ümbritseva sisendi kontekst. Eriti mängib see rolli nimeüksuste tuvastamisel, sest enamik nimesid esinevad keeles harva ning seega ei leia uus tokeniseerija nimedele sobivat lemmat mudeli lemmade sõnastikust. Seega tuleks kaaluda tokeniseerimise loogika muutmist või täiendamist.

Täiendatud mudeli tulemusi mõjutab kindlasti ka asjaolu, et EstNLTK teek ei leia alati sõnale õiget lemmat ja vormi. Tehtud vead morfoloogilise info leidmisel kanduvad läbi tokeniseerija edasi mudelisse ning seeläbi ka mudeli tulemustesse. Samas ei saa esitatud analüüsi tulemusi võtta aboluutsetena, kuna tulemused sõltuvad ka kohandamiseks kasutatavate andmestikkude suurusest ning kvaliteedist.

5 Diskussioon ja tuleviku perspektiivid

Tokeniseerijale tundmatute sõnade vähendamiseks on mitmeid variante. Üks võimalus on toimetada sõnadega, mille lemmat mudeli sõnastikus ei esine sarnaselt *Wordpiece* tokeniseerimisele - jaotada tundmatu sõna üksikuteks sümboliteks või sõnaosadeks. Sellisel juhul peaks ka mudeli lemmade sõnastik sisaldama kõiki korpuses esinevaid üksikuid sümboleid ja sagedasemaid sõnaosasid. Sama loogikat võiks rakendada ka näiteks arvude tokeniseerimisel, mis sõnastikku ei mahu. See tähendaks, et harvaesinevad arvud ei saaks vastena sõnastikust erisõne [UNK] nagu praegu, vaid arv tehakse eraldi numbriteks, mille kõik indeksid lisatakse eraldi tokeniseerija väljundisse. Informatiivsema mudeli sõnastiku saamiseks saaks sõnu ka rohkem reeglipõhiselt grupeerida. Näiteks sisaldab lemmade sõnastik 46 '.ee' lõpuga lemmat, mille võiks kõik liigitada üheks konkreetseks sõneks sõnastikus, mis tähistab veebilehte. Erinevate tokeniseerijatega mudelite väljatreenimine ning ennustuste võrdlemine annaks selguse optimaalse tokeniseerimisstrateegia osas.

Võimalikke arengukohti on ka mudeli analüüsi osas. Vormi esituse pikkuse valimisel toetuti reeglile, et valitud peakomponentide arv peaks seletama ära vähemalt 80% esituste hajuvusest. Tegelikult oleks mõistlikum lahendada vormi esituse pikkuse küsimus läbi katsetuste - valida erinevad vormi esituse pikkused, eeltreenida mudelit teatud arv samme ning võrrelda mudelite tulemusi MLM ülesandel. Selline katse annaks täpsema vastuse küsimusele vormi esituse optimaalsest pikkusest. Samuti oleks asjakohane optimiseerida kohandamisülesannete parameetreid nagu õpisamm ja ploki suurus. Mõlemad esitatud täiendused oleksid samas küllaltki ressursimahukad.

Lisaks tuleks täiendatud mudeli ennustamist detailsemalt uurida, et näha, millistel juhtudel mudel ennustamisega hakkama ei saa. Oluline oleks ka täiendatud mudeli ja EstBERT mudeli vektoresituste ning mudeli kihtide omavaheline võrdlus. Tuleks uurida, kas ja kuidas saaks ära kasutada juba eeltreenitud EstBERT mudelit täiendatud eestikeelsete BERT mudelite treenimiseks.

Morfoloogilise analüüsi kasutamist BERT-tüüpi mudelites on siiamaani implementeeritud ning uuritud vaid üksikutes töödes (Klemen, Krsnik ja Robnik-Šikonja, 2020), (Song *et al.*, 2020). Käesolevas töös esitatud tulemused ja kood annavad aluse põhjalikumateks eksperimentideks ning analüüsideks morfoloogilise info kasutamisest keelemudelites ning konkreetselt eestikeelsete keeleülesannete lahendamisest.

6 Kokkuvõte

Töö esimeses osas esitati ülevaade BERT mudeli olemusest ja toimimisest. Sealjuures kirjeldati ära mudeli jaoks olulised osad - mudeli sõnastiku loomine, tokeniseerija töötamine, vektoresituste tekitamine ning mudeli eeltreenimine ja kohandamine.

Töö teises osas implementeeriti ning seletati lahti BERT mudelile tehtud täiendused. Põhilised muudatused puudutasid mudeli tokeniseerimist ning vektoresituste leidmist, mistõttu tuli vastavusse viia ka kõik ülejäänud mudeli osad. Muudatuste idee seisnes morfoloogilise info lisamises tokeniseerija väljundisse sõnade lemmade ehk algvormide ning sõnade vormide ehk käänete või pöörate näol. Samuti loodi mudelile täiendava keelelise info edasikandmiseks binaarsed kanalid, mida implementatsioonis kasutatakse liitsõnade juurte märkimiseks.

Töö kolmandas osas eeltreeniti täiendatud mudel välja ning analüüsiti selle tulmusi neljal kohandatud ülesandel. Sealjuures selgus, et mudeli täiendav eeltreenimine parandab vaadeldud ülesannetest vaid MLM ülesande tulemusi. Võrreldes EstBERT mudeliga on täiendatud mudeli tulemused oluliselt madalamad, mis on ilmselt põhjustatud tokeniseerijale tundmatute sõnade käsitlusest ning liiga vähesest mudeli eeltreenimisest.

Põhitulemusena näidati töös, et morfoloogilise analüüsi kaasamine BERT mudelisse on tehniliselt võimalik. Samuti treeniti välja mudelid, mille tulemusi saab võrrelda teiste eestikeelsetele ülesannetele kohandatud mudelitega. Kirjutatud koodi saab kasutada alusena mudeli edasi arendamiseks. Kogu töö kood ning tulemused on avalikult kättesaadavad GitHubi repositooriumis `https://github.com/raulniit/transformers`.

Kasutatud allikad

Beltagy, Iz, Matthew E. Peters ja Arman Cohan (2020). *Longformer: The Long-Document Transformer*. arXiv: 2004.05150 [cs.CL].

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever ja Dario Amodei (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].

Collobert, Ronan, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu ja Pavel Kuksa (2011). *Natural Language Processing (almost) from Scratch*. arXiv: 1103.0398 [cs.LG].

Devlin, Jacob (2018). *Multilingual BERT github readme document*. URL: https://github.com/google-research/bert/blob/master/multilingual.md.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee ja Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". *arXiv e-prints*, arXiv:1810.04805.

Ek, Adam, Jean-Philippe Bernardy ja Stergios Chatzikyriakidis (2020). "How does Punctuation Affect Neural Models in Natural Language Inference". Teoses: *Passive and Active Network Measurement Conference*.

He, Pengcheng, Xiaodong Liu, Jianfeng Gao ja Weizhu Chen (2021). *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. arXiv: 2006.03654 [cs.CL].

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals ja Laurent Sifre (2022). *Training Compute-Optimal Large Language Models*. arXiv: 2203.15556 [cs.CL].

Huffman, David A. (1952). "A Method for the Construction of Minimum-Redundancy Codes". *Proceedings of the IRE* 40.9, lk. 1098–1101. DOI: 10.1109/JRPROC.1952.273898.

Hugging Face (2023). URL: \url{https://huggingface.co/models}.

Kittask, Claudia, Kirill Milintsevich ja Kairit Sirts (2020). *Evaluating Multilingual BERT for Estonian*. arXiv: 2010.00454 [cs.CL].

Klemen, Matej, Luka Krsnik ja Marko Robnik-Šikonja (september 2020). "Enhancing deep neural networks with morphological information". *Natural Language Engineering* 29.2, lk. 360–385. DOI: 10.1017/s1351324922000080. URL: https://doi.org/10.1017%2Fs1351324922000080.

Koppel, Kristina ja Jelena Kallas (2018). *Eesti keele ühendkorpus 2017*. DOI: https://doi.org/10.15155/3-00-0000-0000-0000-071E7L.

Lai, Guokun, Qizhe Xie, Hanxiao Liu, Yiming Yang ja Eduard Hovy (2017). *RACE: Large-scale ReAding Comprehension Dataset From Examinations*. arXiv: 1704.04683 [cs.CL].

Laur, Sven, Siim Orasmaa, Dage Särg ja Paul Tammo (mai 2020). "EstNLTK 1.6: Remastered Estonian NLP Pipeline". Teoses: *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, lk. 7154–7162. URL: https://www.aclweb.org/anthology/2020.lrec-1.884.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer ja Veselin Stoyanov (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

Mikolov, Tomas, Kai Chen, Greg Corrado ja Jeffrey Dean (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv: 1301.3781 [cs.CL].

Pajupuu, Hille, Rene Altrov ja Jaan Pajupuu (juuni 2016). "Identifying Polarity in Different Text Types". *Folklore: Electronic Journal of Folklore* 64, lk. 125–142. DOI: 10.7592/FEJF2016.64.polarity.

Rajpurkar, Pranav, Robin Jia ja Percy Liang (2018). *Know What You Don't Know: Unanswerable Questions for SQuAD*. arXiv: 1806.03822 [cs.CL].

Ruder, Sebastian (2018). *A Review of the Neural History of Natural Language Processing*. http://ruder.io/a-review-of-the-recent-history-of-nlp/.

Song, Yuncheng, Shuaifei Song, Juncheng Ge, Menghan Zhang ja Wei Yang (aprill 2020). "Incorporating Morphological Compostions with Transformer to Improve BERT". *Journal of Physics: Conference Series* 1486, lk. 072071. DOI: 10.1088/1742-6596/1486/7/072071.

Sutskever, Ilya, Oriol Vinyals ja Quoc V. Le (2014). *Sequence to Sequence Learning with Neural Networks*. arXiv: 1409.3215 [cs.CL].

Tanvir, Hasan, Claudia Kittask, Sandra Eiche ja Kairit Sirts (2020). "EstBERT: A Pretrained Language-Specific BERT for Estonian". *arXiv e-prints*, arXiv:2011.04784.

Taylor, Wilson L. (1953). “Cloze Procedure: A New Tool for Measuring Readability”. *Journalism Quarterly* 30.4, lk. 415–433. DOI: 10.1177/107769905303000401. eprint: https://doi.org/10.1177/107769905303000401. URL: https://doi.org/10.1177/107769905303000401.

The Hugging Face Course (2022). URL: https://github.com/huggingface/course.

Tjong Kim Sang, Erik F. ja Fien De Meulder (2003). “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. Teoses: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, lk. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

Tkachenko, Alexander, Timo Petmanson ja Sven Laur (august 2013). “Named Entity Recognition in Estonian”. Teoses: *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*. Sofia, Bulgaria: Association for Computational Linguistics, lk. 78–83. URL: https://aclanthology.org/W13-2412.

Transformers (2023). URL: https://github.com/huggingface/transformers.

University of Tartu (2018). *UT Rocket*. DOI: 10.23673/PH6N-0144.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser ja Illia Polosukhin (2017). *Attention Is All You Need*. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

Virtanen, Antti, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter ja Sampo Pyysalo (2019). *Multilingual is not enough: BERT for Finnish*. arXiv: 1912.07076 [cs.CL].

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy ja Samuel R. Bowman (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. arXiv: 1804.07461 [cs.CL].

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest ja Alexander M. Rush (2020). *Transformers: State-of-the-Art Natural Language Processing*. URL: https://github.com/huggingface/transformers.

Wu, Shijie ja Mark Dredze (2020). *Are All Languages Created Equal in Multilingual BERT?* DOI: 10.18653/v1/2020.repl4nlp-1.16.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato,

Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes ja Jeffrey Dean (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv: 1609.08144 [cs.CL].

Zellers, Rowan, Yonatan Bisk, Roy Schwartz ja Yejin Choi (2018). *SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference*. arXiv: 1808.05326 [cs.CL].

Zeman, Daniel et al. (2019). *Universal Dependencies 2.5*. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. URL: http://hdl.handle.net/11234/1-3105.

Zhang, Aston, Zachary C. Lipton, Mu Li ja Alexander J. Smola (2021). "Dive into Deep Learning". *arXiv preprint arXiv:2106.11342*.

Znotiņš, Artūrs ja Guntis Barzdins (september 2020). "LVBERT: Transformer-Based Model for Latvian Language Understanding". Teoses: ISBN: 9781643681160. DOI: 10.3233/FAIA200610.

Lisad

I. Treenimise tulemused

Tabelites 7, 8, 9 ning 10 on esitatud alapeatüki 4.2 detailsed kohandamise tulemused. Tulbad 'Treen kadu' ning 'Val kadu' tähistavad vastavalt mudeli kadu treenimis- ning valideerimisandmestikul.

Tabel 7. MLM ülesande tulemused. *EstBERT mudeli täpsus on saadud orginaalse BERT mudeli MLM ülesande lahendamisel, kus ennustada tuleb maskeeritud sõne. Täiendatud mudelis ennustatakse maskeeritud sõne lemmat ja vormi.

Mudel	Epohh	Treen kadu	Val kadu	Täpsus (lemma)	Täpsus (vorm)
100K	1	4.916	4.925	0.359	0.606
	2	4.702	4.793	0.37	0.628
	3	4.734	4.741	0.375	0.6
	test			0.372	0.607
200K	1	4.603	4.583	0.398	0.626
	2	4.487	4.38	0.405	0.64
	3	4.429	4.431	0.408	0.629
	test			0.402	0.635
300K	1	4.534	4.45	0.402	0.636
	2	4.408	4.247	0.416	0.643
	3	4.36	4.305	0.417	0.633
	test			0.406	0.64
400K	1	4.48	4.392	0.399	0.639
	2	4.312	4.149	0.425	0.666
	3	4.279	4.223	0.431	0.652
	test			0.422	0.656
500K	1	4.323	4.251	0.414	0.65
	2	4.186	4.059	0.432	0.662
	3	4.163	4.099	0.437	0.65
	test			0.433	0.663
EstBERT*	1	3.947	3.642	0.415	
	2	3.498	3.406	0.415	
	3	3.423	3.601	0.423	
	test			0.442	

Tabel 8. Rubriigi klassifitseerimise tulemused

Mudel	Epohh	Treen kadu	Val kadu	Täpsus	Saagis	F1-skoor
100K	1	1.393	1.062	0.663	0.642	0.644
	2	0.832	0.969	0.684	0.678	0.676
	3	0.561	0.934	0.708	0.706	0.707
	test			0.672	0.658	0.663
200K	1	1.304	0.978	0.715	0.681	0.687
	2	0.784	0.898	0.703	0.701	0.699
	3	0.563	0.892	0.715	0.706	0.709
	test			0.699	0.686	0.69
300K	1	1.339	0.96	0.72	0.678	0.685
	2	0.804	0.866	0.703	0.694	0.694
	3	0.591	0.857	0.728	0.712	0.716
	test			0.7	0.688	0.693
400K	1	1.32	0.938	0.736	0.701	0.706
	2	0.807	0.843	0.721	0.712	0.713
	3	0.612	0.831	0.747	0.722	0.728
	test			0.705	0.683	0.69
500K	1	1.338	0.942	0.711	0.691	0.694
	2	0.789	0.834	0.728	0.719	0.72
	3	0.589	0.822	0.738	0.73	0.732
	test			0.706	0.69	0.695
ESTBERT	1	1.139	0.741	0.792	0.766	0.77
	2	0.453	0.647	0.794	0.784	0.786
	3	0.187	0.686	0.81	0.803	0.804
	test			0.794	0.794	0.793

Tabel 9. Lauseosade märgendamise tulemused

Mudel	Epohh	Treen kadu	Val kadu	Täpsus	Saagis	F1-skoor
100K	1	0.541	0.364	0.87	0.867	0.867
	2	0.286	0.294	0.895	0.894	0.894
	3	0.203	0.278	0.904	0.903	0.903
	test			0.911	0.91	0.91
300K	1	0.576	0.378	0.866	0.863	0.863
	2	0.306	0.31	0.89	0.889	0.889
	3	0.235	0.28	0.903	0.901	0.902
	test			0.905	0.904	0.905
500K	1	0.619	0.393	0.86	0.857	0.857
	2	0.323	0.301	0.891	0.89	0.89
	3	0.252	0.282	0.9	0.9	0.9
	test			0.905	0.905	0.905
ESTBERT	1	0.195	0.143	0.96	0.959	0.959
	2	0.072	0.124	0.967	0.967	0.967
	3	0.035	0.129	0.97	0.97	0.97
	test			0.963	0.963	0.963

Tabel 10. Nimeolemite märgendamise tulemused

Mudel	Epohh	Treen kadu	Val kadu	Täpsus	Saagis	F1-skoor
100K	1	0.267	0.184	0.56	0.407	0.471
	2	0.162	0.151	0.622	0.508	0.559
	3	0.115	0.146	0.608	0.547	0.576
	test			0.584	0.533	0.557
200K	1	0.26	0.171	0.559	0.413	0.475
	2	0.159	0.145	0.627	0.519	0.568
	3	0.12	0.14	0.61	0.561	0.584
	test			0.601	0.552	0.576
300K	1	0.263	0.175	0.57	0.408	0.476
	2	0.164	0.145	0.628	0.509	0.562
	3	0.128	0.14	0.627	0.562	0.593
	test			0.609	0.549	0.577
400K	1	0.267	0.176	0.555	0.397	0.463
	2	0.168	0.15	0.615	0.51	0.558
	3	0.134	0.143	0.622	0.556	0.587
	test			0.596	0.546	0.57
500K	1	0.272	0.186	0.529	0.364	0.431
	2	0.175	0.162	0.574	0.458	0.51
	3	0.14	0.149	0.581	0.537	0.558
	test			0.585	0.532	0.557
EstBERT	1	0.124	0.073	0.793	0.834	0.813
	2	0.045	0.07	0.837	0.858	0.848
	3	0.019	0.074	0.834	0.864	0.849
	test			0.838	0.848	0.843

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Raul Niit**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   BERT mudeli kohandamine eesti keelele,

   mille juhendaja(d) on Sven Laur ja Hendrik Šuvalov,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Raul Niit
09.05.2023