TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Informaatika õppekava

Jakob Univer

Grammatiliste vigade parandamine sageduspõhise sünteetilise andmestikuga

Bakalaureusetöö (9 EAP)

Juhendajad: Agnes Luhtaru, BSc
Mark Fišel, PhD

Tartu 2022

Grammatical error correction with frequency-based synthentic corpus

Abstract: In this thesis we introduce a grammatical error correction method with a neural network trained only on synthetic data. The method is useful for languages without big corpora for training a grammatical error correction model, like Estonian. From a smaller human corrected corpus, we found the probabilities of word deletion, addition, substitution and changing word order mistakes in the text. With the help of these probabilities we created a bigger synthetic corpus and we trained a neural network for grammatical error correction on the synthetic data. The author found that the probabilities of mistakes do not have to be very precise and the trained neural network can correct spelling mistakes as well as grammar mistakes.

Keywords: Grammatcal Error Correction, neural network, synthetic data

CERCS: P176 Artificial intelligence

Grammatiliste vigade parandamine sageduspõhise sünteetilise andmestikuga

Lühikokkuvõte: Selles töös tutvustatakse grammatiliste vigade parandamise meetodit tehisnärvivõrguga, mida treenitakse vaid sünteetilise andmestiku peal. Meetod on kasulik keeltele, kus ei leidu suuri korpuseid grammatiliste vigade parandamise mudeli treenimiseks, nagu eesti keel. Väiksemast, inimeste parandatud korpusest leitakse sõnade kustutamise, lisamise, vahetamise ja liigutamise vigade osakaalud ning nende osakaaludega loodi palju suurem sünteetiline korpus. Sünteetilisel andmestikul treeniti tehisnärvivõrk, mis parandas eesti keele grammatikavigu. Autor leidis, et vigade osakaal ei pea olema väga täpne ning treenitud mudel saab hakkama lisaks lause ülesehituse vigadele ka õigekirja vigade parandamisega.

Võtmesõnad: Grammatiliste vigade parandus, tehisnärvivõrk, sünteetiline andmestik

CERCS: P176 Tehisintellekt

Sisukord

1 Sissejuhatus **4**

2 Metoodika **5**
2.1 Mõisted 5
2.2 Varasemad tulemused teistes keeltes 5
2.3 Vigade genereerimine 7
2.4 Tehisnärvivõrgu kirjeldus 9
2.5 Hindamise metoodika 9

3 Tegevused **10**
3.1 Korpuse analüüs 10
3.2 Sünteetiliste korpuste genereerimine 11
3.3 Tehisnärvivõrkude loomine 13
3.4 Mudelite hindamine 14

4 Analüüs **16**
4.1 Kvantitatiivne tulemuste võrdlus 16
4.2 Kvalitatiivne tulemuste võrdlus 17

5 Kokkuvõte **20**

Viidatud kirjandus **21**

Lisad **24**
I. Sõnastik 24
II. Litsents 25

1 Sissejuhatus

Grammatiliste vigade parandamine (ingl *Grammatical Error Correction, GEC*) on ülesanne, mille eesmärk on grammatiliste vigadega tekst teisendada korrektseks tekstiks. Vigadeks võivad olla näiteks kirjavahemärkide ja sõnade valik ja paigutus ning õigekiri. Selline grammatikaparandaja oleks suureks abiks eesti keele õppijatele, sest suudab parandada suure hulga lauseid palju kiiremini kui inimene. See ei oleks tõenäoliselt küll sama täpne, kuid vähendaks siiski suurel määral inimeste tööd.

Tavaliselt tehakse parandusi lausetega ehk süsteemile antakse ette vigane lause ning loodetakse vastu saada parandatud lause. Edukates mudelites teistes keeltes on grammatikavigu parandatud tehisnärvivõrkude abiga, kuid levinud lahendused treenivad võrku päris andmetega korpusel, mille on koostanud heade keeleliste teadmistega inimesed. Selliseid veaparanduskorpuseid ei ole eesti keeles väga palju, samuti ei ole need väga suured, sest inimeste tööaeg on värtuslik ning arvutiga võrreldes ei suuda inimesed grammatika vigu kiiresti parandada. Tehisnärvivõrgud töötavad aga kõige paremini, kui neil on treenimiseks väga palju ja erinevaid andmeid.

Seega on hea ja lihtne lahendus luua ise juurde korrektsest tekstist vigadega tekst ning treenida tehisnärvivõrku sünteetilise korpusega ning peenhäälestada päris tekstidest koosneva korpusega. Sünteetilise korpuse ja juhendatud õppega tehisnärvivõrkudega grammatika vigade parandust on tehtud teiste seas näiteks inglise [7] ja saksa keeles [17], kus on palju suuri korpuseid, millega närvivõrku treenida. Siiski tulevad paremini antud meetodi eelised esile keeltes, kus suuri korpuseid ei ole, nagu näiteks eesti keel. Andmete puudumisel saab need ise luua, saavutades suurte korpustega treenimisega võrreldavad tulemused.

Antud lõputöös leiab autor, milliste sagedustega esineb eestikeelsetes tekstides erinevaid grammatikavigu, loob esimese ainult sünteetilisel korpusel treenitud tehisnärvivõrgu eesti keele jaoks, mis on võimeline parandama eesti keele grammatikavigu. Autor võrdleb erinevate sagedustega vigadega loodud korpustel treenitud mudeleid, et selgitada täpse vigade jaotuse tähtsus sünteetilise korpuse loomisel. Selline süsteem on heaks näiteks ka teistele väikestele keeltele, kuidas suuri korpuseid omamata luua kvaliteetne grammatiliste vigade parandaja.

Metoodika peatükk kirjeldab varasemaid töid, mis on tehtud sarnase meetodi kasutamisest teiste keeltega, kuidas genereeritakse vigu nendes töödes, mis muudatused peaks tegema eesti keele jaoks ning kuidas peaks looma ja treenima tehisnärvivõrku, et saada parim võimalik tulemus. Samuti selgitatakse seal töös kasutusel olevaid mõisteid. Tegevuste peatükis kirjeldatakse detailselt töö käiku ning raskusi, mis esinesid olemasolevas korpuses erinevate vigade leidmisega, sünteetilise korpuse loomisega, tehisnärvivõrgu loomise ja treenimise ning tulemuste analüüsimisega. Analüüsi peatükis saab ülevaate, kui hästi parandavad testkorpusel vigu erinevate sagedustega vigadega korpustel treenitud mudelid, kirjeldatakse, milliseid puudujääke oli erinevatel mudelitel seoses erinevate veatüüpidega ning milliste vigade parandamisega said mudelid paremini hakkama.

2 Metoodika

Selles peatükis selgitatakse töös läbivalt kasutatavaid mõisteid ja kirjeldatakse varasemaid töid, kus kasutati grammatiliste vigade parandamiseks sünteetilisel korpusel treenitud tehisnärvivõrke ning meetodeid, millega antud töös vigu genereeritakse, tehisnärvivõrku treenitakse ja hinnatakse.

2.1 Mõisted

Tehisnärvivõrk on arvustuslik arhitektuur, mis jäljendab bioloogilise aju tegevust. Tehisnärvivõrk koosneb mitmest kihist, neis olevatest tehisneuronitest ehk sõlmedest ja nende omavahelistest ühendustest. Tehisnärvivõrgule ei pea kirjeldama tehtava ülesande raamistikku reeglipõhiselt, vaid see õpib ülesannet näidete põhjal lahendama.

Eeltreenimine on tehisnärvivõrgu treenimise moodus, kus ei treenita tehisnärvivõrku lõpuni ühe andmestiku peal mõnda ülesannet lahendama, vaid katkestatakse poole peal ära. Kui hiljem jätkatakse uuel andmestikul treenimist samast kohast, kus enne katkestati, siis on tegemist eeltreenitud tehisnärvivõrguga.

Tegevust, mis järgneb eeltreenimisele, nimetatakse peenhäälestamiseks (ingl *fine-tuning*). Ehk eeltreenitud mudelit treenitakse ühte kindlat ülesannet lahendama nii kvaliteetselt kui võimalik teistsuguse andmestiku peal, kui see, millega treenimist alustati.

Transformer on mudeli arhitektuur, mis väldib rekurrentsust ja toetub tähelepanu mehhanismile, et sisendi ja väljundi vahel sõltuvusi luua [25]. Transformer mudelitega on masintõlkes saavutatud paremaid tulemusi kui teiste kooder-dekooder arhitektuuridega. Samuti treenib transformer mudel kiiremini kui rekkurentsed või konvolutsioonilised närvivõrgud.

2.2 Varasemad tulemused teistes keeltes

Tavaliselt kasutatakse grammatiliste vigade parandamise mudeli treenimiseks mõnda valmis korpust, kus on inimeste kirjutatud vigadega tekstid ning keeleteaduse professionaalide poolt parandatud korrektsed tekstid. Sellega võivad tekkida probleemid. Näiteks on treenimise jaoks andmeid liiga vähe ja tehisnärvivõrk ei suuda väheste näidetega vajalikku ülesannet ära õppida või on antud korpus liiga palju mõne veatüübi poole kallutatud. Kui keelekorpus on kallutatud, siis suudab tehisnärvivõrk parandada sama tüüpi vigadega tekste tõenäolisemalt, kuid teistsuguste vigadega ei ole tehisnärvivõrgu parandused nii asjakohased. Autori tehtava tehisnärvivõrgu puhul võiks neid probleeme olla lihtsam vältida, sest ise sünteetilist andmestikku luues saab valida vajaliku suuruse ning vigade tüüpide sagedused. Kui selgub, et valitud korpus on mõne vea poole kallutatud, saab teistsuguste sagedustega uuesti proovida.

Sarnaseid grammatiliste vigade parandamise mudeleid on teiste keeltega varemgi tehtud. Need on piiratud koguse päris tekstide olemasolul saavutanud paremaid tulemusi,

kui varasemad tehisnärvivõrgud, mis sellist meetodit ei kasutanud. R. Grundkiewicz jt [7] on kirjutanud, et inglise keele puhul saavutasid nad sünteetilise andmestikuga tehisnärvivõrku õpetades aastal 2019 olemasolevate tipptasemel grammatiliste vigade parandajatega võrreldavaid tulemusi. Samuti saavutasid nad BEA19 jagatud ülesande raames 21 esitatud süsteemi seas piiratud ja vähese ressursiga kategooriates parimad tulemused. Töö autorid kasutasid Aspell [16] õigekirja kontrollijat, et leida valitud sõnale hulk sõnu, millega seda sagedasti segamini võidakse ajada ja nende abiga tekitasid korrektsetesse lausetesse vigasid. Lauses olevaid sõnu eemaldati, asendati sarnastega, vahetati nende järjekorda ja/või lisati juurde. Sarnaseid muudatusi tehti ka tähtede tasemel sõnade sees. Kõigile muudatustele oli määratud konstant, millise tõenäosusega vastav viga tekstis esineda võib. Saadud andmestikuga eeltreeniti tehisnärvivõrku, kuna seal oli palju rohkem näiteid, kui märgendatud inimeste kirjutatud tekstidega korpuses. Hilisem peenhäälestus toimus vastavas kategoorias kasutusel olnud korpustega, sest sealsed veaparanduskorpuse vigade näited suutsid parandada tehisnärvivõrgu täpsust inimeste kirjutatud tekstide grammatiliste vigade parandamisel.

Eesti keelega sama meetodiga grammatiliste vigade parandamise mudeleid tehtud ei ole, kuid on kasutatud sünteetilisi korpuseid, et mudelite vigade parandamise täpsust suurendada [15]. Samuti on teiste meetoditega katsetatud grammatiliste vigade parandust sama korpusega [23], millel katsetatakse mudeleid antud lõputöös [11]. Nende töödega saab võrrelda autori treenitud mudeleid ning hinnata kui hästi see meetod eesti keelega toimib.

Kirjeldatud süsteem oli tehtud inglise keele jaoks, milles leidub väga hea suuruse ja märgendusega korpuseid, et tehisnärvivõrke treenida, kuid teistes keeltes sama häid korpuseid leida on palju keerulisem. Seda illustreerib hästi ka J. Náplava jt [17] kirjutatud töö, kus kasutati sarnaseid võtteid, et luua sünteetilisi andmestikke inglise, saksa, vene ja tšehhi keeles ning nende abil treeniti tehisnärvivõrke, mis suudaksid vastavates keeltes grammatika vigu parandada. Olemasolevatel keele korpustel oli kõige suurem vigade osakaal tšehhi keele korpusel, kus umbes iga viies sõna oli vigane ja kõige väiksem vene keele korpusel, kus vigade protsent oli 6,4%. Samuti olid tšehhi, saksa ja vene keele korpused mitu korda väiksemad kui kõik inglise keele korpused kokku.

Sarnaselt Grundkiewicz jt tööle, valisid J. Náplava jt normaaljaotusest osakaalu igale vea tüübile nii sõnade kui ka tähtede jaoks ning lõid sellega korrektsetest tekstidest inimeste kirjutatud tekstidele sarnanevate vigadega sünteetilise andmestiku iga keele jaoks. Nende töös tuli ka välja keelte erinev käsitlemine tehisnärvivõrguga ennustamisel parema tulemuse saamiseks. Näiteks tekitati tšehhi keele andmestikus lisa viga suur- ja väiketähtede muutmiseks ning kuna vene keele korpus oli palju väiksem teiste keelte korpustest, siis tehisnärvivõrgu peenhäälestamisel kasutati 10 korda vähem vene keele päris näiteid kui teiste keelte puhul. Inglise keele puhul saavutati tipptasemel olemasolevate grammatika vigade paranduse süsteemidega võrreldav tulemus. Saksa keele puhul saavutati üle 25 punkti võrra parema F0,5 skooriga tulemus kui eelneva parima

süsteemiga. Tšehhi keele puhul oli F0,5 skoori vahe üle 20 punkti. Vene keele puhul saavutati küll parem tulemus kui eelnev parim süsteem, kuid üldiselt oli tulemus ikkagi halb - parim F0,5 skoor vaid 50,2 - sest vene keeles oli kõikidest keeltest kõige väiksem korpus, millega tehisnärvivõrku treenida.

Grammatiliste vigade parandamine tehisnärvivõrkudel saab toimuda sarnaselt masintõlkega tehisnärvivõrkudel [22]. Mõlema ülesande puhul on nii sisendiks kui ka väljundiks tekst ning sõnastikud luuakse sõnade omavaheliseks vahetamiseks. Seega saab grammatiliste vigade parandamist käsitleda ka kui masintõlget vigadega tekstist vigadeta tekstiks.

M. Junczys-Dowmunt jt [8] põhjal oli enne nende esitatud tööd maailma tipptasemel grammatiliste vigade paranduse süsteemid fraasipõhised statistilised mudelid mitte tehisnärvivõrgud. Oma töös esitasid nad mitmeid meetodeid, mida varasemalt parimatest masintõlke tehisnärvivõrkudest grammatiliste vigade paranduse tehisnärvivõrkudele üle kanda, et parandada nende vigade parandamise võimekust.

Olulisemad meetodid on väljajätu (ingl *dropout*) suurendamine, treenimise näidete arvu suurendamine nende kordamise abil, erinevate korpuste vigade osakaalu ühtlustamine, sisendi ja väljundi vektoresituste (ingl *embedding*)-ute omavaheline sidumine ja parameetrite muudatustele kaalude määramine. Nende töö näitel on väheste andmetega mudelite treenimine tulemuslikum, kui eeltreenimine tehakse suuremal andmestikul, peenhäälestamine täpsemal ja väiksemal andmestikul ning kasutatakse varem tehtud ükskeelsete keelemudelite abi, kuna need on juba treenimata kasutamiseks sobilikud. Veel aitab tulemust parandada mitme treenitud mudeli omavahel kombineerimine ning mudelite sügavuse suurendamine. Nende ja veel mõne meetodiga saavutati varem parimast tehisnärvivõrgu süsteemist 5,9% parem tulemus. Kuna ka üldiselt parimast fraasipõhisest statistilisest süsteemist saadi 2% parem tulemus, siis saavutati uus tipptasemel süsteem grammatika vigade parandamiseks.

2.3 Vigade genereerimine

Vigade genereerimiseks kasutas autor meetodit, mis on kirjeldatud R. Grundkiewicz jt kirjutatud töös [7]. Mõlemas töös kasutatakse õigekirjakontrollijat Aspell [16], mis annab parameetrina etteantavale sõnale vasteid, millega seda sõna tihti segamini aetakse või kuidas seda sõna tihti valesti kirjutatakse.

Iga lause jaoks saadakse tõenäosus, mis näitab, mitu viga normaaljaotusest lausesse lisada. Vajaliku normaaljaotuse keskväärtus on vigade protsent kogu tekstist jagatud sajaga ja standardhälve on juhuslikult valitud 0,2. Saadud tõenäosus korrutatakse lauses esinevate sõnade arvuga ja ümardatakse lähima täisarvuni. Edasi valitakse lausest ühtlase tõenäosusega nii mitu erinevat sõna.

Iga valitud sõnaga tehakse üks järgnevatest grammatikavigadest: sõna kustutamine, sõna lisamine, sõna liigutamine lauses ja sõna vahetamine sarnase sõnaga. Sõna kustutamisel võetakse valitud sõna lausest ära, sõna lisamisel lisatakse üks juhuslik sõna

lauses valitud sõna ette, sõna liigutamisel vahetatakse valitud sõna asukoht tema ees või taga oleva sõnaga ja sõna vahetamise puhul leitakse Aspelli [16] abiga järjend sõnadest, millega valitud sõna tihti segamini aetakse ning asendatakse valitud sõna ühe juhusliku sõnaga saadud järjendist.

R. Grundkiewicz jt töös anti sõnade vahetamisele tõenäosus 0,7 ja ülejäänud vigadele tõenäosused 0,1, aga kuna antud töö autoril on kättesaadav korpus vigadega tekstidest ja parandatud tekstidest, siis leidis töö autor iga vea jaoks täpse tõenäosuse, mida eesti keelega kasutada. R. Grundkiewicz jt töös tehti vigu ka sisse 10 % juhuslikult valitud sõnadesse, tehes tähtedega sõnas samasuguseid vigu nagu eelmainitud neli veatüüpi tegid sõnadega lauses. See aitab tehisnärvivõrgule ka paremini õigekirja vigu õpetada. Antud lõputöö autor seda ei kasuta, sest Aspelliga [16] tehtud sarnaste sõnade vahetused juba aitavad õigekirja vigade õpetamisele kaasa ning lisa vigade genereerimine sõnade tasemel nõuaks palju rohkem aega ja ressursse.

R. Grundkiewicz jt töös on ka kirjutatud, et kuigi selline vigade genereerimise meetod ei anna konteksti teadlikke vigu, siis esialgsetes katsetes andis see meetod paremaid tulemusi, kui Levenshteini kauguse või sõnade vektoresituste (ingl *embedding*) sarnasuste kasutamine. Samuti selgitatakse, et antud meetod sobib kasutamiseks igale keelele, kus on olemas õigekirja kontrollija.

Vigade genereerimisel on tähtis ka teksti eeltöötlus truecaser-i [4] ning SentencePiece-iga [12], et kasutatavaid tekste ühtlustada ning muuta tehisnärvivõrgule õigete muudatuste leidmine lihtsamaks. Truecasinguks nimetatakse suur- ja väiketähestuse taastamist halvasti või mitte üldse suur- ja väiketähestatud tekstis [14]. Nii truecase kui ka SentencePiece programmiga saab treenida mudelid, mis näiteteksti pealt õppides rakendavad tegevusi ette antud tekstile. Antud lõputöös on see oluline sünteetilise korpuse loomisel, kus korrektne tekst saadakse korpusest, mille tekstid on võetud eestikeelsest Vikipeediast ja uudiste lehekülgedelt, millest tulenevalt ei pruugi nende suur- ja väiketähestus olla ühtlane. Truecase mudeli treenimisel sünteetilisel korpusel saab ühtlustada kogu lõputöös kasutatavaid tekste ning seeläbi vähendada tekstis esinevate suur- ja väiketähe vigade hulka. Antud töös kasutatud truecaser [4], kasutab unigrammide tõenäosulikku lähenemist teksti suur- ja väiketähestuse uurimiseks ning muutmiseks.

SentencePiece on keelest sõltumatu teksti alamsõnedeks sõnestaja(ingl *tokenizer*) ja pöördsõnestaja(ingl *detokenizer*), mida kasutatakse peamiselt tehisnärvivõrkude teksti eeltöötlusel [12]. SentencePiece'i sõnestaja jagab teksti pseudomorfeemideks (ingl *subword*)ja erinevate pseudomorfeemide hulka. SentencePiece'il saab ka määrata parameetriga sõnastiku suurust. Kuna SentencePiece töötab ka sõnestamata teksti peal ning on keelest sõltumatu, siis saab seda hästi eesti keele peal kasutada. Samuti võiks teksti alamsõnedeks tükeldamine aidata tehisnärvivõrgul õigekirjavigu parandada.

2.4 Tehisnärvivõrgu kirjeldus

Tehisnärvivõrgu puhul on samuti aluseks suuresti R. Grundkiewicz jt kirjutatud töö [7]. Seega on süsteemi aluseks Transformer mudelid [25], mida on muudetud GEC-ile spetsiifilisemaks muudatustega, mis on kirjeldatud M. Junczys-Dowmunt jt [8] töös. Ülesobitamise vältimiseks kasutatakse ulatuslikku regulariseerimist. Selleks kasutatakse väljajätumeetodit lähteandmete vektoresitusel (ingl *embedding*) ning tähelepanu ja pärilevivõrkude transformerite kihtidel. Samuti asendati parima nelja mudeli kontrollpunktide keskmistamine eksponentsiaalse silumisega (ingl *exponential smoothing*) ning suurendati miniplokkide suurust.

Aluseks võetud töös katsetatakse mitmete erinevate mudelitega, kuid käesoleva lõputöö autor valib neist kõige algelisema, niinimetatud *base* versiooni, sest see sobis hästi kokku lõputöö vajadustega ning on kõige lihtsamini loodav. Selline struktuur valiti, sest aluseks võetud töös katsetati erinevaid struktuure ning see andis väheste algandmetega alamülesande puhul algse testimisega häid tulemusi. Seega annab see struktuur potentsiaalselt väga head tulemused antud lõputöö puhul, kus on vähe algandmeid.

2.5 Hindamise metoodika

Erinevate grammatiliste vigade parandajate mudelite hindamiseks kasutatakse üldiselt kindlaid korpuseid, et saaks erinevaid mudeleid omavahel võrrelda. Eesti keele jaoks on autoril võimalik selleks kasutada Tartu Ülikoolilt saadud mitte avalikku eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpust [23]. Samuti kasutatakse üldiselt grammatiliste vigade parandaja skoori saamiseks MaxMatch ehk Mˆ2 [2] või ERRANT [3, 1] hindajat. Need mõlemad hindajad vajavad, et tekst oleks M2 formaadis ning kindla märgendusega, kus vigade asukohad on välja toodud. Vajalikku märgendust eesti keele jaoks pole. Järgmine laialdaselt kasutatav hindaja on GLEU [18]. GLEU hindaja jaoks on vaja faile tavatekstina ja vaid sõnadeks tokeniseeritult. Seda on EstNLTK [13] võimeline tegema, nii et GLEU hindajat saab eesti keele puhul kasutada.

GLEU hindaja põhineb masintõlke hindajal BLEU [21] ehk lihtsustatult vaatab see sarnasusi mudeli ja inimeste poolt tõlgitud tekstide n-grammides, karistades kohtade eest, kus mudel pole muudatusi teinud, kuid inimeste pakutud parandustes on muudatused. Arvatakse, et n-grammide hindamise kaudu arvestab GLEU [18] lisaks vähimatele muutustele ka lause ladusust.

3 Tegevused

Selles peatükis kirjeldab autor sünteetilise korpuse aluseks võetud korpust, selle analüüsi. Samuti kirjeldatakse selles lõigus sünteetilise andmestiku loomise, tehisnärvivõrgu treenimise ning hindamise protsesse.

Katsete käigus tehti esmalt inimkorpuse analüüs, et leida erinevate vaetüüpide osakaal eestikeelses tekstis. Seejärel genereeriti leitud sageduse ning võrdluseks võetud sageduste abil sünteetilised andmestikud. Genereeritud andmestikega treeniti 3 mudelit, mis olid eesti keele vigade jaotusega andmestiku peal treenitud ehk eesti mudel, inglise keele vigade jaotusega andmestiku peal treenitud ehk inglise ja ühtlase vigade jaotusega andmestiku peal treenitud ehk ühtlane mudel. Sellised jaotused valiti, et kontrollida, kas täpsete eesti keele vigade jaotusega andmestiku peal treenitud mudel parandab eesti keele grammatikavigu palju paremini kui mõne muu jaotusega andmestikul treenitud mudel. Kõikide mudelite grammatiliste vigade parandamise võimet hinnati GLEU hindajaga, sest see ei vajanud erilist märgendust ega formaati.

3.1 Korpuse analüüs

Sünteetiliste vigade genereerimise jaoks peab teadma eesti keeles kirjutades esinevate vigade sagedusi. Selleks kasutas autor Tartu Ülikoolis eesti keelt võõrkeelena õppivate üliõpilaste kirjutatud lausetest ja nende võimalikest parandustest koosnevat korpust [23]. Selles korpuses on kokku 8921 üliõpilaste kirjutatud lause, mis on jagatud treenimis-, arendus- ja testhulgaks vastavalt suurustega 7121, 1000 ja 800 lauset. Iga hulga vigadega lausete failile on antud kolm faili, kus on kirjas parandatud laused, vastavalt failinimede järelliidetega ref1, ref2 ja ref3. Kui lausele on korpusesse lisatud mitu erinevat parandust, siis on igas kolmes failis ka need erinevad parandused kirjas. Kui korpusesse on lisatud ainult üks või kaks erinevat parandatud lauset, siis on seda sama parandatud lauset dubleeritud järgnevatesse failidesse. Kõiki kolme faili parandatud lause dubleerimine on vajalik GLEU [18] skoori arvutamiseks.

Töö autor leidis tekstist nelja erinevat tüüpi viga: üleliigse sõna lisamist, vajaliku sõna puudumist, sõna asukoha liigutamist lauses ning sõna vahetamist teise sõnaga. Kogu lõputöös kirjutatud kood on kas Pythonis või bash-i skriptidena. Vigadega ja korrektsete lausete erinevuste leidmiseks tokeniseeriti kõigepealt lause sõnadeks EstNLTK [13] abiga. Edasi analüüsiti saadud kahte järjendit moodulis difflib[1] leiduva SequenceMatcher klassi objektiga, mis on loodud järjendite omavaheliste erinevuste leidmiseks. Get_opcodes funktsiooniga saadi järjend tegevustega, mis muudaksid vigadega lause tokeniseeritud järjendi vigadeta lause tokeniseeritud järjendiks.

Kuna SequenceMatcher leiab lisatud, eemaldatud, vahetatud ja samaks jäänud järjendi elemendid ja nende asukohad, siis tuli neljas veatüüp - sõna lauses liigutamine - leida

[1] difflib mooduli dokumentatsioon: `https://docs.python.org/3/library/difflib.html`

eelnevalt leitud muudatusi kombineerides. Sõna ühes kohas eemaldamine ning teises kohas lisamine sai üheks sõna liigutamise veaks. Ühes kohas eemaldamine ning teises kohas lausesse sisse vahetamine sai üheks sõna liigutamise ning üheks sõna lisamise veaks. Sõna ühes kohas lisamine ning teises kohas välja vahetamine sai üheks sõna liigutamise ning üheks sõna kustutamise veaks. Leitud vigade arv salvestati muutujatesse ning tsüklite abiga leiti selliste vigade arv iga lause kohta korpuses. Antud meetodit korrati kõigi kolme võimalike paranduste failiga ning nii treenimis-, arendus- kui ka testimishulgal. Saadud vigade arvud liideti kokku ning sellest kujunesid ka lõputöö vigade genereerimise osas erinevate vigade osakaalud.

SequenceMatcher andis järjendite muutmiste nimekirja indeksite vahemikena järjendis. Seega pidi arvestama ka igas muutuses kasutatud sõnade arvu, et leida vigade osakaal kogu tekstist. Kuna SequenceMatcher andis tulemustes teada ka kõikide muudatuste indeksid lauses, siis sai salvestada ka kõikide sõna liigutamiste kaugused lauses ehk mitme sõna võrra sõna lauses ette või taha poole liigutati. Sellega selgus, et umbes kaks kolmandikku kõikidest liigutamistest on tehtud ühe sõna võrra lauses taha poole ning kuna ka inglise keele peal tehtud töös kasutati ainult ühe sõna võrra sõna asukoha liigutamist, siis otsustas töö autor samuti kasutada vaid ühe sõna võrra liigutamist.

Tabel 1. Vigade arvud ja osakaalud kogu tekstist päris teksti korpuses

Hulk	**Kustutamisi**	**Lisamisi**	**Vahetusi**	**Liigutamisi**
Treening	4869 (4,1%)	4960 (4,2%)	15389 (12,9%)	1386 (1,2%)
Arendus	638 (3,8%)	680 (4%)	2091 (12,3%)	186 (1,1%)
Test	524 (4%)	539 (4,1%)	1681 (12,9%)	144 (1,1%)

Tabelis 1 on kirjas vigade arvud, mis on saadud vigadega faili ja parandatud faili ref1 võrdlemisel. Ülejäänud paranduste failidega võrdlusel olid leitud vigade arvud vaid mõne vea võrra erinevad ning protsendid olid kõikide failide puhul samad, mis tabelis kirjas.

Vigade osakaalud kõikidest tekstis esinevatest vigadest tulid sõnade kustutamisele, sõnade lisamisele, sõnade vahetamisele ja sõnade liigutamisele vastavalt 18%, 19%, 58% ja 5% kogu korpuses esinenud vigadest. Sellise osakaaluga tekitati ka sünteetilisse korpusesse sarnaseid vigu. Iga lause puhul salvestati ka sõnade arv lauses ning selle ja vigade arvuga korpuses, leiti, et umbes 22% sõnadest kogu tekstis on vigadega. Selle tulemuse abiga sai vigade genereerimisel leida vigade arvu, mis korrektsesse lausesse tekitada.

3.2 Sünteetiliste korpuste genereerimine

Sünteetilise korpuse tegemine käis peaaegu täielikult R. Grundkiewicz jt kirjutatud töö [7] järgi. Muudeti vaid erinevate vigade tekstis esinemise tõenäosusi ning algand-

meid, sest lõputöö kasutab eesti keelt ja aluseks võetud töö inglise keelt ning keelest tulenevat eeltöötlust. Puhas tekst saadi Eesti keele ühendkorpusest 2019 [10] Eesti Keeleresursside Keskusest. Saadud korpus oli märgendamata ning vaid lausestatud. Esimese sammuna töötles lõputöö autor korpusest 10 miljonit lauset HTML vormingust tavateksti ning tokeniseeris sõnadeks EstNLTK [13] abiga, eemaldades seejuures laused, mis olid lühemad kui 30 tähemärki ning laused, mis koosnesid ainult numbritest.

Valiti 10 miljonit lauset, sest Naplava jt töös [17] on näidatud, et 10 miljoni lausega saab kontrollida, kas mudel leiab korpusest üles õiged seosed ja hakkab vigu parandama. Mida rohkem lauseid, seda parema korpuse ja seega ka parema mudeli saaks, kuid suuremate mudelite treenimine on raha- ja ajakulukam ning pole antud lõputöö raames mõistlik. Seejärel treenis autor saadud lausete peal truecaser-i [4] mudeli ning rakendas antud mudelit sama teksti peal ning treenis ka SentencePiece'i [12] mudeli, mida samuti rakendas sama teksti peal. Sellega saadi valmis 10 miljonit eeltöödeldud lauset ja sai hakata korrektsesse teksti vigu genereerima.

Vigade genereerimise programm käis korrektse teksti faili läbi lause kaupa ning iga lausega tegi samad tegevused. Lause dekodeeriti SentencePiece [12] mudeliga, et leida sõnade arv lauses ning et oleks lihtsam luua vigu sõnade mitte alamsõnade tasemel. Seejärel leidis programm sõnade arvu lauses ning valis normaaljaotusest keskväärtusega 0,22 ja standarhälbega 0,2 juhusliku tõenäosuse lauses vigade esinemiseks. Keskväärtus 0,22 valiti, sest Tartu Ülikoolist saadud korpuses olid vead umbes 22%-l sõnadel ning standardhälve 0,2 valiti programmi aluseks võetud R. Grundkiewicz jt kirjutatud töö [7] järgi, kus valiti antud parameeter juhuslikult. Normaaljaotusest saadud tõenäosus ja sõnade arv lauses korrutati, et saada lausesse loodavate vigade arv.

Seejärel valiti loodavate vigade arvu jagu lisamise, kustutamise, vahetamise ja liigutamise vigu nii, et iga vealiigi esinemise tõenäosus oli vastavalt 0,19, 0,18, 0,58 ja 0,05, sest päris korpuse analüüsis selgus, et eestikeelsetes tekstides esinevad need vead umbes sellises vahekorras. Saadud vead salvestati järjendisse ning edasi leiti vigade arvu võrra kordumatuid indekseid, mis vastavad sõnadele lauses, millega vigu luua. Leitud kaks järjendit, üks indeksitega ja teine vealiikidega käiakse tsükliga järjest läbi, et viia lauses valitud muudatused sisse. Kui vealiik on sõna lisamine, siis lisati valitud indeksile lauses üks 10 miljonist eeltöödeldud lausest juhuslikult valitud sõna. Kui vealiik oli sõna kustutamine, siis võeti valitud indeksil olev sõna lausest välja. Kui vealiik oli sõna liigutamine ja valitud indeks oli 0, siis vahetati esimene ja teine sõna lauses ära. Kui vealiik oli sõna liigutamine ja valitud indeks ei olnud 0, vahetati indeksil olev sõna ära lauses tema ees oleva sõnaga. Kui vealiik oli sõna vahetamine otsis programm Aspell [16] õigekirja kontrollijaga sõnale järjendi sõnadest, millega lauses olevat sõna tihti segi aetakse. Saadud järjendist valiti juhuslik vaste ning lauses valitud indeksil olev sõna asendati valitud vastega.

Kui valitud indeksil oli lauses kirjavahemärk, asendati see ühe juhuslikult valitud, erineva kirjavahemärgiga. Mõne sõna puhul ei osanud Aspell [16] õigekirja kontrollija

vasteid anda. Sel juhul asendati lauses olev sõna 10 miljoni eeltöödeldud lause hulgast valitud juhusliku sõnaga. Õigekirja kontrollijalt saadud vaste eelised on inimestele loomulikumana tunduva vea loomine lausesse ning hiljem võiks selle meetodiga sõnadesse tähevigade genereerimine aidata tehisnärvivõrgul ka õigekirja vigu parandada.

Kuna ükski vigade loomise meetod, kaasaarvatud Aspelli [16] kasutamine, ei muutnud sõnade suur- ja väiketähelisust, siis vigadega tekstile uuesti truecaser [4] mudelit rakendama ei pidanud. Küll aga pidi enne vigadega teksti faili salvestamist seda SentencePiece [12] mudeliga kodeerima, et hiljem tehisnärvivõrgule ette antavad ja oodatavad tulemused oleks samas vormingus ja sama töötlusega.

Selleks et veenduda, et tehtav eesti keelele omaste vigade osakaaludega genereeritud korpusel treenitud mudel töötab paremini, kui suvaliste vigadega genereeritud korpusel treenitud mudel ning kontrollida, kui palju mõjutavad mudeli täpsust genereeritavate vigade osakaalud, koostas autor veel kaks korpust. Erinevate korpuste vigade osakaalud on näha tabelis 2. Inglise korpuse vigade osakaalud on võetud R.Grundciewicz jt tööst [7], kus inglise keele jaoks määrati sõnade vahetamise tõenäosuseks 0,7 ning ülejäänud vigade tõenäosusteks 0,1. ühtlase korpuse vigade osakaalud on määratud ühtlase jaotusega, et saaks võrrelda, kas vigade osakaalude määramine keelele spetsiifiliseks muudab mudeli tulemust väga palju.

Mudelite treenimise jaoks genereeriti lisaks 10 miljonile lausele iga vigade osakaaluga ka 2000-lauselised arendushulgad, mida töödeldi samamoodi ja samade SentencePiece [12] ning truecaser [4] mudelitega nagu 10 miljoni lauselist treenimishulka.

Tabel 2. Erinevate veatüüpide osakaalud kõikidest vigadest sünteetilistes korpustes

Korpus	**Kustutamisi**	**Lisamisi**	**Vahetusi**	**Liigutamisi**
eesti	18%	19%	58%	5%
inglise	10%	10%	70%	10%
ühtlane	25%	25%	25%	25%

3.3 Tehisnärvivõrkude loomine

Kogu mudeli treenimise ja hindamise skriptide jooksutamine käis Tartu Ülikooli HPC klastris [24]. Mudeli treenimiseks kasutas lõputöö autor Fairseq [20] ning Cuda [19] riistakomplekte. Mudelit treeniti HPC Rocket klastril , 50Gb mäluga ja ühe videokaardi peal.

Kuna mudel käsitleb vigade parandamist kui tõlget vigadega tekstist korrektsesse teksti, siis esimese asjana loodi Fairseq eeltöötlusega sünteetilise ning korrektse tekstifaili jaoks sõnastikud. Eeltöötlusel kasutati peale käitamiseks vajalike parameetrite vaid *joined-dictionary* parameetrit, sest kuigi tegemist on masintõlke süsteemiga, siis

tõlgib see praeguse töö puhul eesti keelest eesti keelde. Seega on lähte- ja tulemuskeele sõnastikud samad.

Fairseq-i mudeli treenimise käsus võttis töö autor parameetrid R. Grundkiewicz jt tööst [7] nii täpselt kui võimalik oli. Arhitektuuriks valiti *transformer*, sest see implementeerib *tranformer base* arhitektuuri, mida kirjeldati alapeatükis 2.4. Õpisamm seati 0,0003 koos uueduste soojendusega esimese 16000 kirje puhul ning tagurpidi ruutjuur planeerijaga. Optimiseerimisfunktsiooniks valiti Adam [9] ning kriteeriumiks (ingl *criterion*) märgendi silumisega ristentroopia ja margendi silumise parameetriks 0,1. Uuendamiste sageduseks seati 12, kuna aluseks võetud töös treeniti mudelit 4 videokaardi peal ning uuendusi tehti iga 3 iteratsiooni järel, kuid lõputöö autoril oli palju lihtsam panna programm jooksma ühe videokaardi peal ja uuendada iga 12 iteratsiooni järel. *Patience* patrameeter ehk mitme validatsiooni skoori mitte kasvamise järel treenimine enneaegselt lõpetatakse seati 10 peale ja maksimaalne epohhide arv 100-ks. Õppimise plokkide suurus piirati maksimaalse märkide arvuga 15000 ning valideerimiste vahe määrati 5000 uuendusele. Üldine väljajätt seati 0,3 peale ja tähelepanu väljajätt ning aktveerimise väljajätt 0,1 peale.

Nii eeltöötluse kui ka treenimise käsud kirjutati vastavate parameetritega bash skripti ning pandi HPC klastris jooksma. Kogu selle skripti jooksutamine võttis aega umbes kuus ja pool ööpäeva. Mudelit treeniti 45 epohhi ning iga epohhi järel loodi mudelist kontrollpunkt, mida sai katsetada ning hinnata eelnevalt vigade statistika jaoks kasutatud päris teksti korpuse peal.

Kõigi kolme sünteetilise korpuse jaoks oli mudeli treenimise skript samasuguste parameetritega. Esimesena treeniti lõpuni eesti keele vigade jaotusega korpusel treeninud mudel ning kui selle mudeli kontrollpunktide GLEU [18] hindamise tulemused olid Tartu Ülikoolist saadud korpusel head, pandi treenima ka ülejäänud kaks mudelit.

3.4 Mudelite hindamine

Nagu juba mainitud, kasutati mudeli hindamiseks Tartu Ülikooli eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpust [23]. Vigadega teksti parandamise õigsuse hindamiseks kasutas töö autor GLEU hindajat [18], sest see oli ainuke, mis oli eesti keele jaoks võimalik. Sarnaselt treenimisele toimus ka osa hindamise protsessist Fairseq ja CUDA riistakomplektide abiga. Päris tekstide korpusest kasutas töö autor mudelite hindamiseks arendushulka ja testhulka, sest need olid piisavalt suured, et hõlmata piisavalt erinevaid lauseid ja vigu, kuid piisavalt väiksed, et nendega hindamine oleks tehtav mõistliku ajaga.

Hindamine toimus sarnaselt treenimisele bash skriptiga, kus esimeseks sammuks oli Fairseq-i vahenditega parandada salvestatud mudeliga teksti vead ning salvestada parandatud laused faili. Kuna mudelile ette antud treeningandmed olid tokeniseeritud SentencePiece [12] mudeliga ning suur- ja väiketähestatud truecaser [4] mudeliga, siis ka hindamiseks ette antud fail oli samade mudelitega töödeldud. Tulemuse fail oli samuti

sarnases vormingus. GLEU [18] hindamine toimib aga kõige paremini ainult sõnadeks tokeniseeritud tavateksti peal. Seega tuli järgmise sammuna hindamise skriptis tulemuse fail dekodeerida SentencePiece [12] mudeli abil tavatekstiks. Hindamise skripti viimaseks sammuks oli GLEU programmi tööle panemine, andes programmile parameetriteks päris tekstide korpuses kolm võimalike parandustega faili, vigadega teksti fail ning mudeli parandatud ja järeltöödeldud fail.

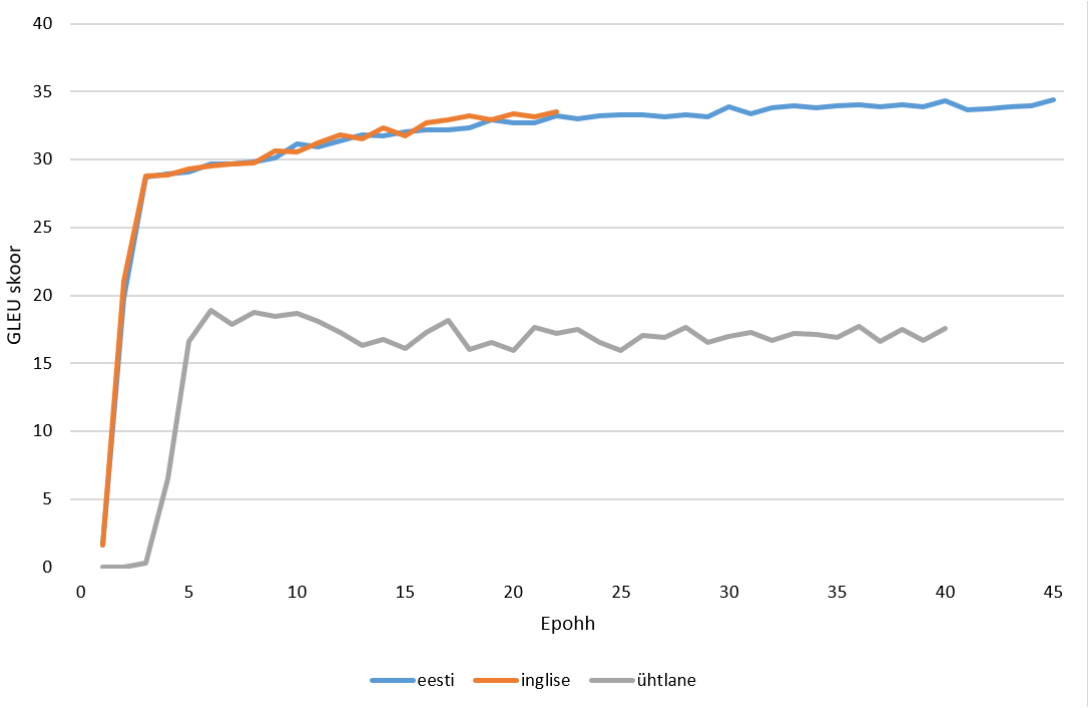
Kirjeldatud hindamise skripti jooksutas autor tsükli abiga iga epohhi järel salvestatud mudeli kontrollpunktiga, et näha mudeli arengut iga epohhi järel. Parimat epohhi igast mudelist hinnati ka päris tekstide korpuse testhulgal. Hindamise tulemused on kirjeldatud järgmises peatükis.

4 Analüüs

Selles peatükis kirjeldatakse erinevatel sünteetilistel korpustel treeninud mudeleid nii GLEU [18] hinnanguga kui ka mudeli parandatud tekstide ja korpuses esindatud parandustе võrdlemisega autori äranägemise järgi.

4.1 Kvantitatiivne tulemuste võrdlus

Mudeleid hindas autor vaid GLEU [18] skooriga, sest see oli ainuke eestikeelse grammatiliste vigade parandaja jaoks kohandatud hindamisprogramm. GLEU skoor näitab, kui sarnane oli mudeli parandatud tekst päris teksti korpuses olevate inimeste parandatud tekstidega skaalal nullist üheni. Parema loetavuse nimel on kõik esitatavad GLEU skoorid korrutatud sajaga.



Joonis 1. Erinevate vigade suhetega korpustega treenitud mudelite võrdlus

Joonisel 1 on näha kõigi kolme mudeli GLEU [18] skoori muutumine iga treenitud epohhi järel. Eesti keelele omaste vigade suhtega ehk korpusega, kus 19% vigadest olid sõna lisamise vead, 18% sõna kustutamise vead, 58% sõnade vahetamise vead ja 5% sõnade liigutamise vead, treenitud mudel ja inglise keelel kasutatud vigade suhtega ehk

korpusega, kus 10% vigadest olid sõna lisamise vead, 10% sõna kustutamise vead, 70% sõnade vahetamise vead ja 10% sõnade liigutamise vead, treenitud mudel saavutasid peaaegu samasuguseid tulemusi. Kuna nende kahe mudeli ainuke erinevus oli vigade osakaal sünteetilises korpuses, saame järeldada, et vigade osakaalu väga täpne seadmine ei ole mudeli algse täpsuse kasvatamisel väga oluline. Kuna ühtlase jaotusega vigadega korpusel treeninud mudel saab iga epohhiga palju halvemaid tulemusi kui ülejäänud kaks mudelit, siis peaks vigade osakaal olema eesti keele puhul parima tulemuse saamiseks ülejäänud kahele mudelile lähedane.

Eesti keele jaotusega mudel saavutas treenimise lõpuks Tartu Ülikooli korpuse arendushulgal GLEU [18] skoori 34,4 ning testimishulgal 32,1. Inglise keele jaotusega mudel saavutas GLEU skoorid 33,5 arendushulgal ning 31,5 testhulgal. Kui seda võrrelda inglise keele peal GLEU-ga parimaid tulemusi saanud töödega [5, 6], siis on skoorid palju madalamad, kuid need mudelid on treenitud või peenhäälestatud ekspertide poolt parandatud korpuste peal, mitte ainult sünteetiliste andmete peal ning kasutavad ka muid mudeli täpsuse parandamise meetodeid peale sünteetilise korpuse peal treenimise. Samuti võib inglise keele peal kasutatud JFLEG [18] korpuses erineda vigade arv ja osakaal antud lõputöös analüüsitud korpusest.

Kui võrrelda tulemusi eesti keele peal tehtud sarnaste töödega [15, 11], siis on näha, et GLEU skoorid on täiesti võrreldavad. Seega võiks lõputöös kirjeldatud meetod olla hea algus parema eestikeelse grammatiliste vigade parandaja mudeli loomiseks.

Kuna parandusteta failiga GLEU [18] hindamise tulemus on 28,7, siis saame öelda, et eesti ja inglise vigade osakaaludega korpustel treenitud mudelid muutsid teksti korrektsemaks, ühtlase vigade osakaaluga korpusel treenitud mudel, aga tekitab teksti vigu juurde.

4.2 Kvalitatiivne tulemuste võrdlus

Mudeli tehtud paranduste paremaks arusaamiseks võrdles autor ka ise Tartu Ülikooli korpuse arendushulgas olevat vigadega faili, mudeli parandatud faili ning inimeste parandatud faile. Nii oli kõige lihtsam leida tüüpvigu, mille parandamisega mudelid hakkama said või millega esines probleeme. Samuti leidis autor ka muutusi, mida mudel vigadega lausetega läbi viis, sama skriptiga, millega ta leidis eesti keele vigade jaotuse alapeatükis 3.1.

Mudelite parandatud lauseid läbi vaadates selgus, et mõne lause puhul oli mudel teinud paranduse hoopis venekeelsete sõnadega. Tihti toimus selline muudatus lausega, kus kasutati jutumärke. Näiteks vigadega lause „mina olen juba lugenud Eduardi Vilde näidendi „ Pisuhänd ” .” paranduseks pakkus mudel lauset „mina olen juba lugenud Eduardi Vilde näidendit Pisuhänd .”. Sünteetilise korpuse lausete täpsemal ülevaatamisel selgus, et päris mitu lauset olid poolenisti vene keeles ning sellega õppis ka mudel eestikeelsesse teksti vahel venekeelseid sõnu panema. Seda oleks saanud ennetada

sünteetilise korpuse moodustamisel karmima kontrolliga lausete valikul, kuid autor ei osanud oodata Eesti Keeleressursside Keskusest saadud korpusest venekeelset teksti.

Ühtlase vigade jaotusega korpusel treeninud mudeli tehtud parandusi ülejäänud kahe mudeli omadega võrreldes on selge, et see mudel üritab teha liiga palju parandusi. Kui eesti ja inglise vigade jaotusega korpustel treeninud mudelid teevad muutusi 300-400 lauses 1000-st, siis ühtlane mudel teeb muudatusi 950 lauses. Samuti teeb see mudel igas lauses umbes 25% sõnadega parandusi. Seetõttu on ka ühtlase mudeli parima epohhi GLEU [18] skoor 18,9, mis on madalam kui GLEU skoor muutmata vigadega tekstil. Probleemi põhjuseks on tõenäoliselt sünteetilises korpuses vähene Aspelliga [16] tehtud vigade hulk ning suur kustutamiste ja lisamiste arv, mis õpetab mudelit tegema liiga palju parandusi sõnade lisamise, liigutamise ja äravõtmisega lauses ning liiga vahe sõnavormide ning käändelõppude muutmisega sõnades.

Ülejäänud kaks mudelit ehk eesti ja inglise vigade jaotustega korpustel treeninud mudelid jätsid enamuse lausetest parandamata ning lausetel, mida need muutsid, tehti parandusi üldiselt 1-3 sõnaga lauses. Seda leiti sama programmiga, millega leiti Tartu Ülikooli õppijakeele korpuse muudatusi alapeatükis 3.1. Sellega vältisid mudelid lausetesse paljude uute vigade tegemist nagu oli probleemiks ühtlase jaotusega vigadega treenitud mudelil. Mudelite parandatud lauseid kontrollides selgus, et mõlemad mudelid said üldiselt hästi hakkama lühemate lausetega, kus oli õigekirja vigu ainult ühes sõnas. Pikemate lausetega, eriti lausete kokku panemise või lahku ajamise ning kirjavahemärkidega ei saanud mudelid väga hästi hakkama.

Tabelis 3 on töödud näitelaused koos eesti keele vigade jaotusega sünteetilise korpusega treenitud mudeli soovitatud parandatud lausetega. Esimene lause näitab, et mudel oskab lühikestes parandada õigekirja vigu tihti kasutatavatel sõnadel. Lausetest 3, 6 ja 8 näeme, et mudel oskab lisada lausesse sõnapaaride puuduvaid osi: „arvutit kasutada", „jäi ilma", „tõstab immuunsüsteem". Samuti näeme nendest näidetest, et mõnikord muudab mudel lause küll grammatiliselt korrektsemaks, kuid võivad sisse tekkida loogikavead. Lauses 7 näeme, et kui mudelile anda ette mitu lauset korraga, üritab ta neid kokku üheks panna. Lausetes 4 ja 5 näeme, et mudel saab hakkam lihtsamate kirjavahemärkide vigadega ning oskab neile ka sobivaid sidesõnu juurde panna. Teine lause on hea näide sellest, kui sõna on lauses liigutatud rohkem kui ühe sõna kaugusele, ei oska mudel seda parandada, kuna sünteetilises korpuses on sõnade liigutamise vead maksimaalselt vaid ühe sõna kaugusel.

Tabel 3. Näitelaused Tartu Ülikooli korpusest koos eesti mudeli parandustega

Parandaja	**Lause**
	1.
Vigadega	inimesed tegevad liiga vähe , et loodust puhtana hoida ja säilitada .
Mudel	inimesed teevad liiga vähe , et loodust puhtana hoida ja säilitada .
Inimene	inimesed teevad liiga vähe , et loodust puhtana hoida ja säilitada .
	2.
Vigadega	võib-olla see arvamus kellelegi imelikuks paistab .
Mudel	võib-olla see arvamus kellelegi imelikuks paistab .
Inimene	võib-olla paistab see arvamus kellelegi imelikuna .
	3.
Vigadega	näiteks , ühes toas on võimalik lauamänge mängida ja teises arvutit .
Mudel	näiteks , ühes toas on võimalik lauamänge mängida ja teises arvutit kasutada .
Inimene	näiteks on ühes toas võimalik mängida laua- ja teises arvutimänge .
	4.
Vigadega	aga perekonna õnn on mitte ainult selles , tal on teine külg ka .
Mudel	aga perekonna õnn on mitte ainult selles , et tal on teine külg ka .
Inimene	aga perekonna õnn pole mitte ainult selles , tal on teine külg ka .
	5.
Vigadega	minu meelest , mööduvad siin parimad aastad üliõpilaste elus .
Mudel	minu meelest mööduvad siin parimad aastad üliõpilaste elus .
Inimene	minu meelest mööduvad siin parimad aastad üliõpilaste elus .
	6.
Vigadega	minu hea sõbranna jäi pedagoogikakoolist .
Mudel	minu hea sõbranna jäi pedagoogikakoolist ilma .
Inimene	sain endale hea sõbranna pedagoogikakoolist .
	7.
Vigadega	niisugune ootamine võib jätkuda aastaid . Tundmatus on jube asi .
Mudel	niisugune ootamine võib jätkuda aastaid , Tundmatus on jube asi .
Inimene	niisugune ootamine võib jätkuda aastaid . Teadmatus on jube asi .
	8.
Vigadega	kuna immuunsüsteem on nõrgestatud , tõstab vastuvõtlikkust haigustele .
Mudel	kuna immuunsüsteem on nõrgestatud , tõstab immuunsüsteem vastuvõtlikkust haigustele .
Inimene	kuna immuunsüsteem on nõrgestatud , tõuseb vastuvõtlikkus haiguste suhtes .

5 Kokkuvõte

Käesolevas lõputöös analüüsis autor TÜ eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpuses esinevaid grammatika vigasid, et selgitada välja eesti keele puhul sõnade kustutamise, lisamise, vahetamise ja liigutamise vigade osakaal tekstis. Leitud osakaaludega ning võrdluseks inglise keele peal proovitud ja ühtlase vigade osakaaluga loodi 10 miljoni lause suurused sünteetilised korpused. Selleks võeti korrektne tekst ning loodi sinna vigu vastavalt vigade osakaalule. Sõnade vahetamisel kasutati õigekirja kontrollijat Aspell [16], et vahetada lauses sõnu teiste sarnaste sõnadega. Saadud korpustega loodi esimesed ainult sünteetilise andmestiku peal treenitud eesti keele grammatika parandamise mudelid. Igal sünteetilisel korpusel treeniti Tartu Ülikooli HPC klastris tehisnärvivõrk, mis parandaks eesti keele grammatika vigu. Treenitud mudelite iga epohhi järel salvestatud kontrollpunkte hinnati GLEU [18] skooriga TÜ eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpuse arendushulgal ning iga mudeli parimat kontrollpunkti ka testhulgal.

Eesti keele ning inglise keele vigade jaotustega korpustel treeninud mudelid saavutasid parima epohhiga päris tekstide korpusel GLEU [18] skoorid vastavalt 32,1 ja 31,5, mis on paremad kui muutmata lausete GLEU skoor 28,7. Seega parandasid mudelid ette antud teksti grammatikat. Ühtlase vigade jaotusega sünteetilisel korpusel treeninud mudel lõi ette antud teksti vigu juurde. Kvalitatiivne analüüs mudelite parandatud lausetest näitas ka, et sünteetilisel korpusel treenimine õpetas mudeleid lisaks lause ülesehituse vigadele parandama ka sõnade õigekirja vigu. Ükski treenitud mudel ei oska hästi parandada kirjavahemärkide vigu. Samuti mudelitel parandamisega raskusi, kui sõna on lauses liigutatud kaugemale kui ühe sõna kaugusele oma algsest asukohast.

Loodud mudelite tulemused näitavad, et sünteetilise korpusega eeltreenitud mudel annab hea alguse eesti keele grammatiliste vigade parandamise mudeli loomiseks. Seega omamata suuri inimeste poolt parandatud ja märgendatud korpuseid saab luua hea kvaliteediga grammatiliste vigade parandamise mudeli. Samuti sai töö tulemustest teada, et väga täpne vigade osakaal sünteetilise korpuse loomisel, ei ole tähtis parima täpsuse saavutamiseks tehisnärvivõrguga grammatika vigade parandamisel.

Viidatud kirjandus

[1] Christopher Bryant, Mariano Felice ja Ted Briscoe. "Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction". Teoses: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, juuli 2017, lk. 793–805. DOI: 10.18653/v1/P17-1074. URL: https://aclanthology.org/P17-1074.

[2] Daniel Dahlmeier ja Hwee Tou Ng. "Better Evaluation for Grammatical Error Correction". Teoses: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, juuni 2012, lk. 568–572. URL: https://aclanthology.org/N12-1067.

[3] Mariano Felice, Christopher Bryant ja Ted Briscoe. "Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments". Teoses: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, detsember 2016, lk. 825–835. URL: https://aclanthology.org/C16-1079.

[4] Mark Fišel, Lisa Korotkova ja Liisa Rätsep. *TartuNLP/truecaser*. https://github.com/TartuNLP/truecaser. (05.05.2022). 2018.

[5] Tao Ge, Furu Wei ja Ming Zhou. "Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study". *CoRR* abs/1807.01270 (2018). arXiv: 1807.01270. URL: http://arxiv.org/abs/1807.01270.

[6] Roman Grundkiewicz ja Marcin Junczys-Dowmunt. "Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation". Teoses: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, juuni 2018, lk. 284–290. DOI: 10.18653/v1/N18-2046. URL: https://aclanthology.org/N18-2046.

[7] Roman Grundkiewicz, Marcin Junczys-Dowmunt ja Kenneth Heafield. "Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data". Teoses: *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Florence, Italy: Association for Computational Linguistics, august 2019, lk. 252–263. DOI: 10.18653/v1/W19-4427. URL: https://aclanthology.org/W19-4427.

[8] Marcin Junczys-Dowmunt *et al.* “Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task”. Teoses: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, juuni 2018, lk. 595–606. DOI: 10.18653/v1/N18-1055. URL: https://aclanthology.org/N18-1055.

[9] Diederik P. Kingma ja Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[10] Kristina Koppel ja Jelena Kallas. “Eesti keele ühendkorpus 2019.” Teoses: DOI: 10.15155/3-00-0000-0000-0000-08565L.

[11] Elizaveta Korotkova *et al. Grammatical Error Correction and Style Transfer via Zero-shot Monolingual Translation*. 2019. DOI: 10.48550/ARXIV.1903.11283. URL: https://arxiv.org/abs/1903.11283.

[12] Taku Kudo ja John Richardson. “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing”. Teoses: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, november 2018, lk. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012.

[13] Sven Laur *et al.* “EstNLTK 1.6: Remastered Estonian NLP Pipeline”. Teoses: *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, mai 2020, lk. 7154–7162. URL: https://www.aclweb.org/anthology/2020.lrec-1.884.

[14] Lucian Vlad Lita *et al.* “tRuEcasIng”. Teoses: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, juuli 2003, lk. 152–159. DOI: 10.3115/1075096.1075116. URL: https://aclanthology.org/P03-1020.

[15] Agnes Luhtaru. *Grammatiliste vigade parandamine mitmekeelse neuromasintõlkega*. 2020. URL: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69750&year=2020.

[16] Wojciech Muła. *aspell-python - Python bindings for GNU aspell*. https://github.com/WojciechMula/aspell-python. (17.04.2022). 2020.

[17] Jakub Náplava ja Milan Straka. “Grammatical Error Correction in Low-Resource Scenarios”. Teoses: *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*. Hong Kong, China: Association for Computational Linguistics, november 2019, lk. 346–356. DOI: 10.18653/v1/D19-5545. URL: https://aclanthology.org/D19-5545.

[18] Courtney Napoles *et al.* “Ground Truth for Grammatical Error Correction Metrics”. Teoses: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, juuli 2015, lk. 588–593. DOI: 10.3115/v1/P15-2097. URL: https://aclanthology.org/P15-2097.

[19] NVIDIA, Péter Vingelmann ja Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[20] Myle Ott *et al.* “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. Teoses: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.

[21] Kishore Papineni *et al.* “Bleu: a Method for Automatic Evaluation of Machine Translation”. Teoses: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, juuli 2002, lk. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[22] Alla Rozovskaya ja Dan Roth. “Grammatical Error Correction: Machine Translation and Classifiers”. Teoses: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, august 2016, lk. 2205–2215. DOI: 10.18653/v1/P16-1208. URL: https://aclanthology.org/P16-1208.

[23] Ingrid Rummo ja Kristiina Praakli. “TÜ eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpus [The language learner’s corpus of the department of Estonian language of the University of Tartu]”. Teoses: *EAAL 2017: 16th annual conference Language as an ecosystem, 20-21 April 2017, Tallinn, Estonia: abstracts, 2017, p. 12-13*. 2017.

[24] University of Tartu. *UT Rocket*. https://share.neic.no/#/marketplace-public-offering/c8107e145e0d41f7a016b72825072287/. 2018. DOI: 10.23673/PH6N-0144.

[25] Ashish Vaswani *et al.* “Attention is All you Need”. Teoses: *Advances in Neural Information Processing Systems*. Köide 30. 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Lisad

I. Sõnastik

BEA19 ingl *Building Educational Applications 2019 Shared Task*. Üritus aastal 2019, mille käigus võistlesid erinevad töögrupid, et luua kõige paremat grammatika vigade parandamise mudelit. 6

EstNLTK 2016. aastal Tartu Ülikoolis loodud teekide kogu, mis võimaldab eesti keelset teksti tokeniseerida, teha morfoloogilist analüüsi ning palju muud. 9, 10, 12

F0,5 Mõõdik, millega hinnata mudeli õigsust (ingl *accuracy*). See on harmooniline keskmine täpsusest (ingl *precision*) ja saagisest (ingl *recall*). F0,5 skooris on täpsusel kaks korda suurem kaal kui saagisel. 6, 7

GEC Grammatiliste vigade parandamine (ingl *Grammmatical Error Correction*). 9

HPC *High Performance Computing*. Tartu Ülikooli Linuxi põhine arvutusklaster Slurm järjekorra süsteemiga. 13, 20

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Jakob Univer**,
*(*autori nimi)

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

   Grammatiliste vigade parandamine sageduspõhise sünteetilise andmestikuga,
   *(*lõputöö pealkiri)

   mille juhendajad on Agnes Luhtaru ja Mark Fišel,
   *(*juhendaja nimi)

   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.

Jakob Univer
10.05.2022