TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Arvutiteaduse instituut
Andmeteaduse õppekava

Hendrik Šuvalov

Raamistik närvivõrgupõhiste infoeraldustöövoogude loomiseks

Magistritöö (15 EAP)

Juhendajad: Dage Särg, MA
Raivo Kolde, PhD
Sven Laur, PhD

Tartu 2022

Raamistik närvivõrgupõhiste infoeraldustöövoogude loomiseks

Lühikokkuvõte:

Meditsiinilised tekstid, nagu näiteks diagnoosid ja epikriisid, esinevad enamjaolt struktureerimata kujul, tihti vabateksti näol. Nendest tekstidest väärtusliku info (nimeolemid ja nendevahelised semantilised seosed) kättesaamiseks kasutatakse üldiselt reegli- ja mustripõhiseid lähenemisi, sh. regulaaravaldisi. Enamikel juhtudel on see kõige kiirem ja efektiivsem lähenemine, kuid eelkõige antud domeenis võib see olla keeruline, kui tekstis esineb palju kirjavigu või kui me ei tea täpselt, mis mustreid otsida. Sellisel juhul sooritaksid närvivõrgud edukamalt tööd kui reeglipõhised lähenemised, kuna nad oskavad ära õppida sõnade tähendused vastavalt kontekstile, milles need esinevad. Käesoleva töö tulemus on töövoog, mis lubab kasutajal luua infoeraldustöövooge meditsiinilistel tekstidel kasutades EstMedBERT keelemudelit, mis on spetsiifiliselt eel-treenitud eestikeelsetel meditsiinitekstidel ja mida saab peenhäälestada klassifitseerima sõnesid. Kui mudel on õppinud esialgsete andmete pealt ülesande ära, saab seda kasutada järgnevate tekstide märgendamiseks, mida kasutaja kontrollib ning järjest rohkemate andmete peal iteratiivselt treenib. Sellist tüüpi treenimist nimetatakse inimsekkumisega õppeks (*human-in-the-loop*) ning see on osa aktiivõppest. Selline lähenemine võib olla kasulikum teatud tüüpi infoeraldusülesanneteks ning uute nimeolemite leidmiseks töövoogude loomine võib antud lähenemise puhul kasutaja jaoks kergem olla, kuna see ei nõua temalt tehnilisi oskusi. Lisaks valminud tööle kasutasime ka enda arendatud töövoogu, et arendada enda EstMedBERT mudelit kasutav märgendaja, rakendasime seda tekstidele ning analüüsisime nii meie lähenemist kui ka tulemusi.

Võtmesõnad:
Närvivõrgud, BERT, medBERT, infoeraldustöövood, loomuliku keele töötlus, meditsiinitekstid, töövood, nimeolemite märgendamine

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

Framework for neural network based fact extraction workflows

Abstract:

Medical texts, such as diagnoses and epicrises, are due to their nature often unstructured, sometimes in the form of free text. The common practice for extracting useful information from them, such as named entities (e.g. drug or disease names) and their semantic relations, is by using rule or pattern-based extraction, namely regular expressions. In most cases, this is the fastest and most effective approach, however, in certain circumstances this can be difficult, for example, if the text contains misspellings of words or in cases where we do not know the patterns to look for in the first place but could detect them once we saw them. This is a task for which neural network language models could prove to be useful, as they are capable of understanding the meaning of words based on the context they appear in. The main result of this thesis is a pipeline for implementing fact extraction tasks on medical texts. It uses EstMedBERT, a Bidirectional Encoder Representations from Transformers (BERT) model specifically pre-trained on Estonian medical texts, which can be fine-tuned to classify tokens using labelled data given by the user implementing the task. Having initially learned the task, the model will continue labelling new data under the supervision of the user, who will correct any mistakes and, using active learning, retrain the model. This is considered a human-in-the-loop approach for training neural networks. This approach could be a more effective solution to some fact extraction tasks in the medical field and implementing new tasks using this pipeline is easier to the user on a technical level, making it more accessible to people in medical domains. Moreover, in addition to providing the pipeline, as a result of this thesis, an example task has also been implemented using this approach and both the process and results have been analyzed.

Keywords:
Neural networks, BERT, EstMedBERT, fact extraction, natural language processing, medical texts, pipeline, named entity recognition

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

Sisukord

1 Sissejuhatus

Eesti Tervise infosüsteem, ligipääsetav läbi Patsiendiportaali (`www.digilugu.ee`), on patsientide terviseandmete keskhoidla, mis sisaldab juba üle 15 miljoni meditsiinidokumendi ligikaudu 1,3 miljonist inimesest [1]. Need andmed sisaldavad olulist meditsiinilist informatsiooni patsientide kohta, mida annab kasutada statistilisteks analüüsideks, et pakkuda patsientidele lisaväärtust. Näiteks kasutab personaalmeditsiini valdkond patsientide geneetilist informatsiooni kombinatsioonis nende meditsiiniandmetega, et pakkuda patsientidele efektiivsemat abi lähtudes nende isiklikust ja unikaalsest meditsiinilisest taustast [2]. Meditsiiniandmeteks on näiteks epikriisid, saatekirjade vastused, e-tõendid jms [1]. Suur osa nendest andmetest on aga struktureerimata kujul, näiteks vabatekstina, mida ei saa otseselt statistiliseks analüüsiks kasutada. Selle jaoks, et struktureeritud kujul andmeid nendest tekstidest kätte saada, saab kasutada loomuliku keele töötlust. Sellist tüüpi ülesandeid, mille käigus ekstraheeritakse strukrureeritud informatsiooni struktureerimata tekstidest, et neid statistliseks analüüsiks kasutada, kutsutakse infoeralduse ülesanneteks [3]. Toortekstidest nimeolendite leidmist (nt. nimed, asukohad, meditsiinilised koodid) ja neid kategooriatesse jaotamist nimetatakse nimeolendite märgendamiseks (NER ehk *named entity recognition*) [4]. Mida efektiivsemad need töövood on, seda rohkem väärtust me andmetest saame ning seda rohkem lisaväärtust saab tulemusena patsient.

Nimeolemite märgendamiseks saab kasutada närvivõrkudel põhinevaid keelemudeleid, mida kõigepealt eel-treenitakse (*pre-training*) suure korpuse peal, et mudel õpiks ära keele ja vastava domeeni omapärad, ning seejärel peenhäälestatakse (*fine-tuning*) konkreetse ülesande jaoks vastavalt sellele, milliseid nimeolemeid tekstist otsitakse [5].

Eel-treenitud mudeli peenhäälestamise jaoks konkreetse ülesande täitmiseks on vaja märgendatud andmeid, kus on sõned (*token*) ning nende vastavad märgendused (üldjuhul 0 või otsitav olem). Märgendused võib genereerida automaatselt (nt. reeglipõhiselt) või manuaalselt, kus kasutaja ise märgendab iga sõne kohta, kas tegemist on otsitava olemiga või mitte. Genereeritud märgenduste puhul on võimalik andmeid saada kiiremini ja mugavamalt, kuid need on vähem täpsed kui kasutaja poolt märgendatud andmed.

Inimsekkumisega (*human-in-the-loop*) mudeli treenimine kombineerib juhendatud masinõppe ja aktiivõppe, nii et inimene märgendab mudeli jaoks andmeid, mille abil mudel treenitakse, ning seejärel kontrollib ja parandab mudeli poolt genereeritud ennustusi [6].

Närvivõrkude kasutamine nimeolemite märgendamisel on viimase ajaga võrreldes reeglipõhiste lähenemistega populaarsust kogunud [7]. Tüüpiline juhendatud õppega lähenemine kujutab endast seda, et närvivõrgule antakse ette märgendatud tekstikorpus, mille pealt ta õpib erinevad märgendamisklassid. S. Zhang et al. kombineerisid reeglipõhise

lähenemise närvivõrkudega [8]. Nad kasutasid olemasolevat regulaaravaldisel põhinevat märgendajat, et genereerida treeningandmed närvivõrgu jaoks. Nad demonstreerisid, et närvivõrkude põhine märgendaja klassifitseeris õigesti juhtumeid, mida esialgne regulaaravaldis kätte ei saanud ning lisas märgendamisele täpsust juurde.

Y. Zhao ja J. Liu kasutasid inimsekkumisega õpet, et treenida nimeolemite märgendaja [9]. Nad kasutasid hiina- ja ingliskeelseid Vikipeedia [1] tekstikorpuseid, et eel-treenida enda sügav närvivõrk (*Deep Neural Network ehk DNN*) ning peenhäälestasid inimese kontrollimisega mudelit edaspidiste märgendustega. Nad leidsid, et see saavutab võrreldavaid tulemusi muude tüüpiliste lahendustega, lahendades probleemi, et treenimiseks on palju andmeid vaja ja vähendades aega, mis kulus inimestel märgendamiseks.

Sarnaseid lähenemisi on ka meditsiinidomeenis kasutatud. C. Wen et al. implementeerisid nimeolemite märgendaja, mis kasutas erinevaid keelemudeleid, et märgendada meditsiinilistes tekstides erinevaid nimeolemeid (nt. raviminimed, haigused jms.) [10]. Nad leidsid, et antud lähenemine annab märkimisväärselt häid tulemusi olemite märgendamises ja võimaldab kasu saada suurtes kogustes märgendamata meditsiinilistest korpustest, et mudel saaks keeledomeenist aru.

Närvivõrke on ka kasutatud meditsiinilises kontekstides dokumentide klassifitseerimisel (ehk kogu tekstil, mitte sõnedel nimeolemite märgendaja näol). J. Yang et al. kasutasid sügavat närvivõrku, et vabatekstilistel haiglaaruannetel klassifitseerida allergiliste reaktsioonide esinemist [11]. Nad leidsid, et mudeli implementeerimine vähendas inimeste poolt ülevaatamist 63.8% võrra ning ja tuvastas 24.2% rohkem juhtumeid.

Käesolevas töös kasutasime Meelis Perli poolt Digiloost pärinevate kliiniliste dokumentide abil eel-treenitud BERT (*Bidirectional Encoder Representations from Transformers*) mudelit EstMedBERT [12], et arendada Label Studio liides, mille abil saab kasutaja rakendada inimsekkumisega õppe lähenemist, et jooksvalt märgendada kliiniliste dokumentide pealt otsitavaid nimeolemeid, peenhäälestada märgenduste abil EstMedBERT mudelit, parandades treenitud mudeli ennustusi, seeläbi iteratiivselt arendades EstMedBERT mudelit. Samuti on valminud peenhäälestatud mudelit kasutav nimeolemite märgendaja, mis on ühilduv EstNLTK[2] märgendajatega ning mida saab edaspidi töövoogudes kasutada, et tekstidel otsitavaid nimeolemeid märgendada. Töö käigus valminud koodirepositoorium on kättesaadav Tartu Ülikooli GitLab repositooriumis - `https://gitlab.cs.ut.ee/shuva/labelstudio-backend-with-taggers`.

[1] `https://wikipedia.org`
[2] `https://github.com/estnltk/estnltk`

2 Tehniline taust

Kättesaadava meditsiinilise informatsiooni hulk järjest tõuseb, kuid laialdaselt on seda informatsiooni vabatekstilisel kujul, mis teeb nende andmete analüüsimise keeruliseks. Kuna arstid keskenduvalt peamiselt sellele, et kogu oluline informatsioon patsiendi staatuse kohta saaks võimalikult kiiresti kirjutatud, on tihti tekst umbmäärane ja seal esineb kirjavigu. Et antud andmete peal statistilist analüüsi läbi viia, on meil vaja struktureerimata tekstist ekstraheerida meie jaoks oluline informatsioon, olgu see siis näiteks ravimite nimed, patsientide sümptomid vms. Nimeolemite märgendamine on selle protsessi esimene ja vaieldavalt kõige olulisem osa [13].

2.1 Nimeolemite märgendamine

Nimeolemite märgendamine on toortekstist nimeolemite tuvastamine ning nende kategooriatesse jaotamine [4]. Oluline on tähele panna, et kuigi populaarseteks nimeolemite kategooriateks loetakse nimesi, kuupäevi, organisatsioone [4] jms, siis käesolevas valdkonnas võib olla kategooriaks ka näiteks meditsiinilised koodid, raviminimed jms.

Kui varasemalt kasutatud nimeolemite märgendajad on laialdaselt reeglipõhised ja heuristilised, siis tänapäeval on hakatud laialdasemalt kasutusele võtma aina enam masinõppe- ja tehisnärvivõrkudel põhinevaid märgendajaid [7].

Antud töö käigus arendatud koodirepositoorium lubab kasutajal treenida BERT mudelil põhinevat nimeolemite märgendajat, kuid võimaldab kasutada lisafunktsioonina ka reeglipõhiseid EstNTLK märgendajaid, et Label Studios eel-märgendada tekstis otsitavad sõnad ning need mudeli treenimiseks muuta unikaalseteks sõnedeks, et tõsta märgendaja täpsust (vt. Sektsioon 2.2).

2.1.1 Reeglipõhised märgendajad

Reeglipõhiste märgendajate eeliseks on see, et nende loomiseks pole vaja suurt treeningkorpust ning nende arendamine käib mugavalt läbi iteratiivsete sammude, kus järjest reegleid lisades või muutes üritatakse iga iteratsiooni järgselt kätte saada rohkem õigeid ning vähem valesi vasteid. Seetõttu on need laialdaselt kasutusel ka tänapäeval [7]. Enamjaolt kasutavad reeglipõhised nimeolendite märgendajad meditsiinidomeenis regulaaravaldisi [14] või sõnastikke, et tekstist nimeolemeid leida. STACC [3] on kliiniliste

[3] https://stacc.ee/ai-solutions/online-media/text-analytics/

uuringute läbiviimiseks loonud mitmeid töövooge, mis kasutavad enamjaolt regulaaravaldisepõhiseid märgendajaid, et leida tekstist olemeid nagu näiteks haiguste nimesid, sümptomeid, ravimite nimesi jms.

Regulaaravaldisi kasutavate märgendajate arendamine algab üldiselt esmase naiivse regulaaravaldise loomisega. Näiteks kui me üritaks kliinilistest dokumentidest kätte saada diagnoose, kus esineb pahaloomuliste kasvajate levikut kirjeldavaid TNM-koode [15], mis enamasti esinevad kujul *T<näitaja>N<näitaja>M<näitaja>*, kus näitajaks on kas number või täht, alustaksime me ilmselt otsimist regulaaravaldismustriga *r"T.N.M."*. Seejärel otsiks me andmebaasist selle regulaaravaldisega vasteid ning uuriks, kas kättesaadud vastete seas on õiged tulemused ning kas kättesaamata vastete seas leidub tegelikult otsitavaid vasteid. Antud regulaaravaldisega saaksime me kätte kõige tüüpilisemad vasted (vt. Joonis 1).

```
Carcinoides intestinei ilei pT4N1M0 operata.
Carcinoides intestinei ilei pT1aN2M0 operata
Carcinoma neuroendocrinum  hepatis<T3NxMo st. IIIA
Eesnäärme pahaloomuline kasvaja cT3NxMx; Gleason 4+4=8p
Carcinoma ventriculi cT3N3M1(hep), HER2 neg (FISH) st IV
Carcinoma cutis nasi T1N0M0 st I
Ca ampullae recti pT3NoMo st. II
```

Joonis 1. Naiivse regulaaravaldise tulemused

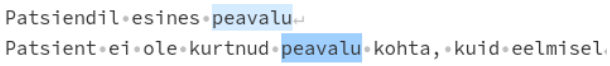
Nagu testandmetest on näha, jääksid meil mõned prefiksid ja sufiksid kättesaamata ning ka üks keerulisem TNM-kood. Seetõttu uuendaksime me regulaaravaldist, et ta täpsus tõuseks. Laiendaksime regulaaravaldist ja arvestaksime rohkemate võimalike sümbolitega - *r".?T..?N..?M."*. Nüüd oleks meil ka vastetes tähtede prefiksid ja sufiksid (vt. Joonis 2).

```
Carcinoides intestinei ilei pT4N1M0 operata.
Carcinoides intestinei ilei pT1aN2M0 operata
Carcinoma neuroendocrinum  hepatis<T3NxMo st. IIIA
Eesnäärme pahaloomuline kasvaja cT3NxMx; Gleason 4+4=8p
Carcinoma ventriculi cT3N3M1(hep), HER2 neg (FISH) st IV
Carcinoma cutis nasi T1N0M0 st I
Ca ampullae recti pT3NoMo st. II
```

Joonis 2. Keerulisema regulaaravaldise tulemused

Kuid nagu tulemustest näha, jääks meil uue regulaaravaldisega *T* koodi prefiksi puudumisel sisse ka vahel irrelevantne sümbol enne klassifikatsiooni (nt. tühik või <). Reaalsete andmete peal tuleks ka välja, et arstid asendavad tihti numbrilist 0 sümbolit tähega O ja muud sarnased domeenispetsiifikad. Erinevaid nüansse arvestades võib regulaaravaldise

arendamine olla ajakulukas protsess ning mustri süntaks nõuab ka tehnilist oskust. Antud näide on ka väga naiivne selle poolest, et klassifikatsioon on kindla struktuuriga, kuid tihti sõltub otsitavate olemite tähendus kontekstist. Näiteks sümptomite puhul võib neid tekstis esineda kontekstis, kus need ei ole tegelikult patsiendi terviseseisundit kirjeldavad (vt. Joonis 3).



Joonis 3. Peavalu esinemine tekstis

Sellisel juhul kui me otsiks näiteks sõnastikupõhise lähenemisega mustreid, ei pruugi need kindlalt patsiendi seisundit kirjeldada, vaid esineda üldisemas kontekstis. See on reeglipõhise lähenemise üks suurimaid puudujääke.

2.1.2 Masinõppe- ja tehisnärvivõrgupõhised märgendajad

Masinõppe- ja tehisnärvivõrgupõhiste märgendajate kasutumise populaarsus on tänapäeval tõusnud [7]. Nadeau ja Sekine jaotavad enda töös masinõppepõhised märgendajad kolme kategooriasse:

1. Juhendatud õppe põhised (nt. otsustuspuud, Markovi peitmudelid jms.)
2. Osaliselt juhendatud õppe põhised (nt. alustatakse inimese poolt antud näidetest ning seejärel laiendatakse leksikoni)
3. Juhendamata õppe põhised (tüüpiliselt klasterdamine)

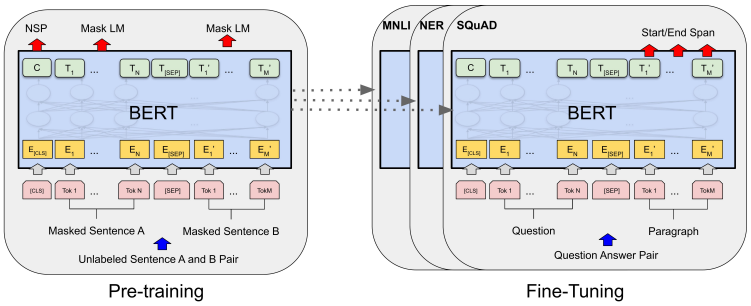
Masinõppepõhiste märgendajate puhul tuleb andmetunnused manuaalselt ette anda, kuid tehisnärvivõrgupõhistel märgendajatel on treeningprotsess, mille käigus nad õpivad tunnused ära ise [16]. Tüüpiliselt kasutavad tehisnärvivõrkude põhised märgendajad kahesuunalisi pika lühiajalise mälu kihte (*bidirectional long short-term memory* ehk bi-LSTM) [3]. Antud töös kasutame nimeolemite märgendamiseks BERT mudelit kasutades inimsekkumisega õpet, mis on juhendatud õppe alamtüüp.

2.2 BERT

BERT on loomuliku keele töötluse keelemudel, mis eel-treenib kahesuunalisi sõne esitusi tekstidest, võttes arvesse sõnest mõlemal pool esinevaid teisi sõnesid, seeläbi õppides ära ka konteksti, milles sõne esineb [17]. Selle tulemusena saab eel-treenitud BERT mudelit peenhäälestada konkreetse ülesande jaoks (nt. antud kontekstis nimeolemite märgendamine) lisades olemasolevale närvivõrgule juurde lineaarkihi (*linear layer*, vt. Joonis 4) [17].

BERT'i eel-treenimise käigus tükeldatakse sõnad tokeniseerija abil sõnedeks ning igale sõnele omistatakse arvuline väärtus. Seejärel õpib mudel keelt mõistma läbi kahe protsessi - maskeeritud keele mudelleerimise (*Masked-Language Modeling*) ja järgneva lause ennustamise (*Next Sentence Prediction*) [17]. Esimesel juhul võetakse tekstis järjestikused sõned, asendatakse neist osa ära (muudetakse sõneks <*MASK*>) ning treenitakse mudelit ennustama, mis sõne sinna kuulub. Järgneva lause ennustamise korral võetakse tekstidest lõike, mis esinevad üksteise järel ja lõike, mis ei esine ning seeläbi treenitakse mudelit õppima, mis lõigud on omavahel seotud.

Oluline tähelepanek on, et treenimise käigus asenduvad sõned arvulise väärtusega ning me saame tükeldamise käigus manuaalselt ise sõnesid sõnavarasse juurde panna. Näiteks võime ise muuta märgendaja abil kõik kuupäevad tekstis sõnedeks <*KUUPV*> ja lisada antud sõne ka mudeli sõnavarasse. Sel juhul õpiks keeleline mudel kohtlema igat kuupäeva sama väärtusega, mis aitaks mudelil keelest kiiremini aru saada, kuid mitte nii täpselt, sest ta ei eristaks erinevaid kuupäevi.



Joonis 4. BERT'i treenimine [17].

2.2.1 BERT'i peenhäälestamine

BERT'i peenhäälestamise mõte on võtta olemasolev domeenispetsiifilisest keelest aru saav mudel ning adapteerida see uut ülesannet lahendama. Selle käigus lisatakse eel-treenitud BERT mudeli viimasele kihile üks väljundkiht juurde ning treenitakse kogu närvivõrku ainult mõne epohhiga [17]. Uue lisatud kihi algväärtused luuakse suvaliselt, kuna need pole veel treenitud ning uue kihiga mudel vajab seeläbi edasist treenimist konkreetse ülesande peal [12]. Kuna peenhäälestamise käigus treenitakse ainult mudeli pealmisi kihte, võtab protsess palju vähem aega, mis lubab ka antud töö kontekstis inimsekkumisega mudelit treenida. Kui eel-treenimine kasutab juhendamata õpet, ehk saab toorel kujul teksti, siis peenhäälestamine toimub juhendatud õppega ehk mudelile antud tekstikorpused on märgendatud.

BERT mudeleid saab peenhäälestada erinevat tüüpi ülesanneteks ning uus kiht, mis mudelile lisatakse, sõltub ülesandeks, mille jaoks mudelit peenhäälestatakse. Transformers[4] pakett pakub 6 erinevat mudeli pead (*model head*) erinevateks ülesanneteks, mille hulgas on näiteks küsimustele vastamine (*QuestionAnswering*), kus mudel eraldab kasutaja poolt esitatud küsimusele vastuse tekstist, järjendite klassifitseerimine (*SequenceClassification*), mis klassifitseerib antud järjendeid, sõne klassifitseerimine (*TokenClassification*) jms. Antud töös on keskendutud sõne klassifitseerimise jaoks peenhäälestamisele, kuna nimeolendi märgendaja kasutab seda peenhäälestatud mudelit, et olemite tüüpi sõnedele märgendada.

Huggingface'i dokumentatsioonide[5] järgi kasutatakse enamasti sõne klassifitseerimisele peenhäälestatud mudeleid kolmeks ülesandeks:

- Nimeolemite märgendamine
- Sõnaliikide määramine (*Part-of-speech tagging*) - Iga sõna puhul märgendatakse antud sõnaliik (tegusõna, nimisõna jms.)
- Tükeldamine - Sõnede leidmine, mis kuulub samasse entiteeti. Töötab sarnaselt eelmisele kahele ülesandele, kuid kasutab IOB märgendamist, ehk igale kategooriale vastab kaks märgendit: B ehk algus (*begin*) ja I ehk sisemine (*inside*) ja sõnad, mis ei kuulu ühegi märgendi alla, saavad märgenduse O ehk väljas (*outside*) [4].

Põhimõtteliselt saab ka informatsiooni eraldamiseks kasutada tükeldamise lähenemist, kombineerides reeglipõhise lähenemise närvivõrkude põhise lähenemisega, kuid antud

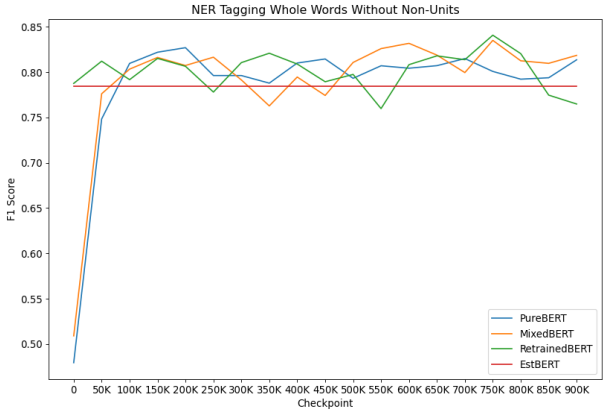
[4]Transformers - `https://huggingface.co/transformers/`
[5]Huggingface token classification - `https://huggingface.co/course/chapter7/2`

töös keskendusime taristu arendamisele, mis võimaldab nimeolemite märgendamise ülesannet.

2.2.2 EstMedBERT

Antud töös on valitud mudeliks, mida märgendaja kasutab, Meelis Perli poolt treenitud EstMedBERT mudeli arhitektuur [12], kuna need on eel-treenitud spetsiifiliselt eestikeelsete meditsiiniandmete peal ning näitavad paremaid tulemusi tekstist arusaamisel ning peenhäälestatud mudelite korral spetsiifiliste ülesannete täitmisel.

M. Perli kasutas BERT mudelite eel-treenimiseks kahte andmestikku - epikriise elektroonilistest terviseandmetest[6] ja Eesti keele üldkorpust[7] [18] [12]. Ta võrdles enda töös erinevate parameetritega eel-treenitud mudeleid, võttes alusmudeliks olemasoleva Eesti keele korpusel eel-treenitud EstBERT mudeli [19] (vt. Joonis 5). Iga eel-treenitud mudel oli peenhäälestatud märgendama ühikuid ja mõõtmisi (kokku 8 klassi, sh. nt. kaal, pikkus, pulss jms). Ta analüüsis erinevate eel-treenitud mudelite peenhäälestamisjärgset F1 skoori vastavate nimeolemite klassifitseerimisel ja leidis, et F1 skoorid, mis kirjeldavad täpsuse (*precision*) ja saagise (*recall*) harmoonilist keskmist, on meditsiiniliste andmetega eel-treenitud mudelitel kõrgemad (umbes 0.82 vs 0.785). Kuna meie töö keskendub nimeolemite märgendaja loomisele just eestikeelsetel meditsiiniandmetel, kasutasime selleks M. Perli mudeleid.

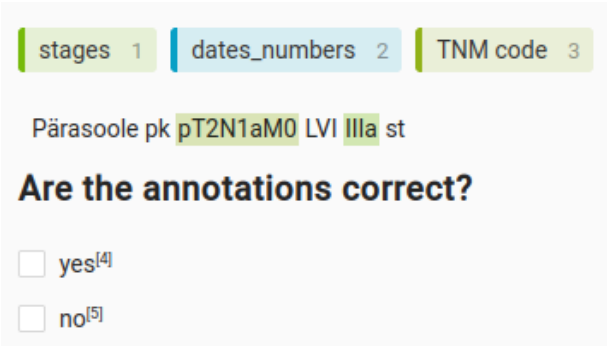


Joonis 5. Erinevate mudelite võrdlus NER ülesandel [12].

[6] https://e-estonia.com/solutions/healthcare/e-health-records/
[7] https://doi.org/10.15155/3-00-0000-0000-0000-071E7L

2.3 Label Studio

Label Studio[8] on vaba tarkvara programm, mis võimaldab mugavalt kasutajal andmeid importida, olgu nad tekstid või pildid, ning seejärel enda poolt kohandatud kujul neid märgendada. Kasutaja saab projekti luues ise valida, mis kujul ta märgendusi tahab, tekstide puhul näiteks kas kogu teksti klassifitseerida, seda kokku võtta, sõnesid märgendada jms. Antud näites (vt. Joonis 6) on Label Studio visuaalses liideses kombineeritud sõnetasemel märgendamine ja teksti klassifitseerimine. Label Studio võimaldab kasutajal kasutada klaviatuuri, et valida sõne märgend ning seejärel tõmmata vastav ala, seeläbi märgendades selle. Programm jookseb masina peal pordil (tüüpiliselt 8080) ning graafiline kasutajaliides suhtleb tagaliidesega läbi päringute.



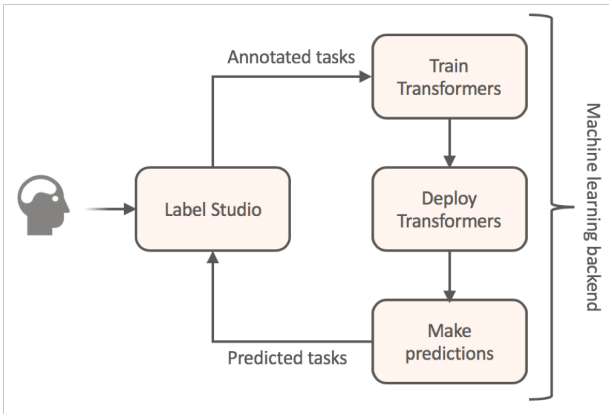
Joonis 6. Label Studio näidisülesanne

Kui kasutaja on ülesande jaoks piisavalt tekste märgendanud, saab ta need enda valitud kujul eksportida (nt. *JSON*, *CSV*, *TSV* formaadis). *JSON* kujul eksporditud andmed sisaldavad seejärel informatsiooni märgendatud tekstide kohta, kus on muuhulgas olemas algkujul tekst ja iga kasutaja poolt loodud märgendus ning selle märgenduse piir (*span*). Näiteks, kui tekstis "Patsidendil esines eesnäärme kasvaja T2N1M3"märgendada ära TNM-kood, oleks *JSON* failis meile kätte saadav kogu algtekst, vastav TNM-kood ning selle piir ehk vahemik (37-42). Niimoodi saab kasutada Label Studiot, et mugavalt tekste märgendada ja pärast eksporditud *JSON* failist vajalikud märgendused parsida.

[8] `https://labelstud.io/`

2.3.1 Label Studio masinõppe tagaliides

Label Studio loojad, Heartex[9], on ka arendanud programmile tagaliidese Label Studio ML backend[10] (masinõppe tagaliides), mis lubab kasutajal integreerida enda mudeleid Label Studioga, et nende abil andmetes märgendusi ennustada või märgendatud teksti abil mudeleid treenida (vt. Joonis 7).



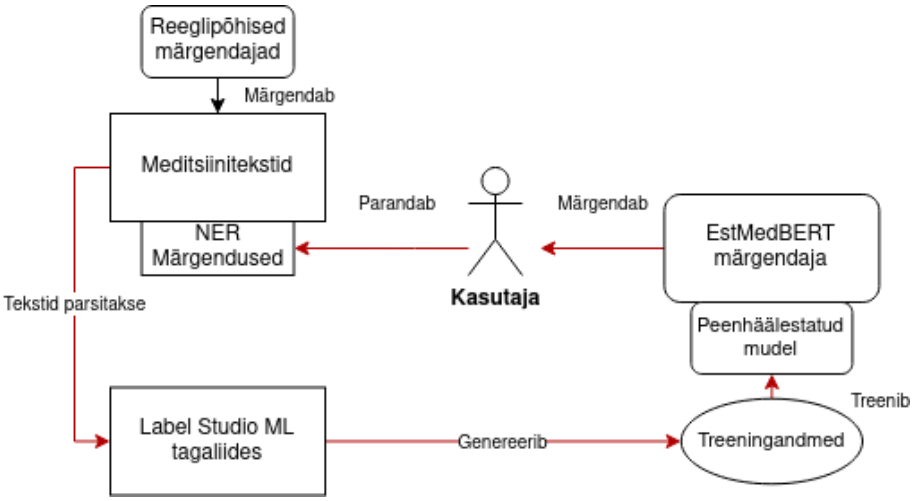
Joonis 7. Label Studio tagaliidese kasutusskeem [20]

Tagaliides on loodud Pythonis, kus kasutajal on võimalus luua enda programm, kasutades olemasolevat klassi *LabelStudioMLBase*, kirjutades üle klassi funktsioonid (peamiselt *__init__()*, *predict()* ja *fit()*) enda loodud funktsioonidega, mis võimaldab kasutajal enda arhitektuuriga mudeleid Label Studios kasutada. Olles enda klassi teinud, on võimalik parsida valminud märgendusi ning viia andmed kujule, mille abil saaks *fit()* funktsioonis mudelit treenida ja *predict()* funktsioonis oma mudelit kasutada, et Label Studios automaatselt selle abil tekstidel märgendusi ennustada. Label Studio ML backend jookseb samuti masina pordil (tüüpiliselt 9090) ning suhtleb Label Studioga läbi päringute.

[9] https://heartex.com/
[10] https://github.com/heartexlabs/label-studio-ml-backend

3 Valminud lahendus

Käesoleva magistritöö raames lõime enda Label Studio ML backendil põhineva liidese, mis kasutab EstMedBERT'i mudelit. Valminud lahendus lubab kasutajal Label Studios tekste märgendades jooksvalt peenhäälestada mudelit, mis ennustab tekstidel märgendusi, neid märgendusi kontrollida ning seejärel mudeli treenimist jätkata uute ja parandatud andmetega (vt. Joonis 8, punased nooled indikeerivad treeningtsüklit ehk *loop*'i *human-in-the-loop*'is).



Joonis 8. Töövoo diagramm

3.1 Kasutamine

Kui kasutaja tahab märgendajat treenida, peab ta esmalt looma klassi, mis pärib meie poolt loodud klassi *RetrainableSpanExtractor*. Järgnevalt peab kasutaja defineerima mis eel-treenitud mudelit meie poolt loodud *BertNerTagger* klass kasutab, mis märgendusi ta tahab, et mudel treeniks ning mis regulaaravaldisepõhiseid EstNTLK märgendajaid kasutada, et muuta nende vasted spetsiaalseteks sõnedeks. Seejärel saab ta tagaliidese püsti panna ja Label Studio sättetes sellega ühendada. Täpsemad tehnilised juhised on olemas ka koodirepositooriumis.

Kui kasutaja just ei alusta juba peenhäälestatud mudelist, peab ta esmalt alustama ise märgendamist, võttes järjest ülesandeid ette ning märgendades mis osad tekstist on otsitavad nimeolemid, mida ta soovib, et *BertNerTagger* märgendaja õpiks leidma. Olles

märgendanud ära enda jaoks sobival hulgal tekste, saab ta sättete alt alustada mudeli treenimist, mis üldiselt võtab alla minuti aega (Label Studio lubab ka peale igat uut märgendust mudelit treenida, kuid see võtaks liiga palju aega). Seejärel saab ta kasutada olemasolevat peenhäälestatud mudelit, et genereerida soovitud kogustes seni märgendamata tekstidele märgendused. See tähendab, et kasutaja saab järjest sirvida tekste üle ning vajadusel parandada märgendusi, genereerides juurde uusi treeningandmeid. Mida täpsem on mudel, mida märgendaja kasutab, seda vähem parandusi peab kasutaja tegema ning seda kiirem on ka treeningandmete genereerimine. Peale igat treenimist salvestatakse peenhäälestatud mudel.

Olles piisavalt palju tekste läbi käinud ning veendunud, et esinevate tekstide puhul ennustab märgendaja piisavalt korrektselt nimeolemeid, saab kasutaja lõpetada treenimise ning võtta kasutusele viimase peenhäälestatud mudeli. Kasutaja saab seejärel rakendada *BertNerTagger*'it, andes talle argumendiks soovitud treenimise käigus valminud peenhäälestatud mudeli, et edaspidi töövoogudes seda märgendamiseks kasutada.

3.2 Label Studio masinõppe tagaliides

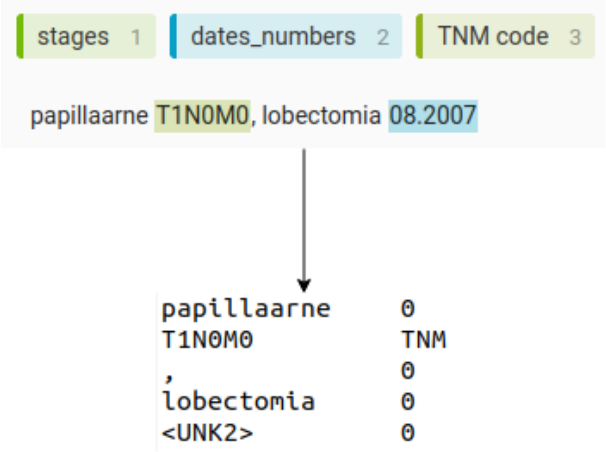
Label Studio ML tagaliidese dokumentatsiooni[11] järgi tuleks enda tagaliidese arenduseks pärida *LabelStudioMLBase* klass ning asendada funktsioonid ja seda me töös tegime.

- *__init__()* - funktsioon kutsutakse välja esialgsel Label Studio ja selle tagaliidese ühendamisel, peale mudeli treenimist ning vastavalt sätetele ka igal ennustamisel. Talle antakse kaasa eelmise töö lõpetamise (ehk mudeli treenimise) tulemused, mis sisaldab peenhäälestatud mudelit sisaldava kausta teed (*path*). Seejärel initsieeritakse nii regulaaravaldisepõhised EstNLTK märgendajad kui ka *BertNerTagger*, mida edaspidi kasutatakse märgenduste genereerimiseks.
- *predict()* - funktsioon kutsutakse välja siis, kui kasutaja on valinud enda soovitud koguse tekste ning soovib neile märgendused ennustada või uue teksti avamisel, kui sätetest on valitud interaktiivne ennustamine. Funktsiooni väljakutsumisel antakse kaasa ülesanded ehk tekstid, mida ta märgendama peab. Lõime funktsioonid, mis parsivad antud ülesandeid, teevad neist EstNTLK *Text* objektid, rakendavad nii regulaaravaldisepõhiseid märgendajaid kui ka meie loodud EstMedBERT mudeli põhist märgendajat nende peal ning seejärel vormistavad *JSON* kujul märgendatud tekstid, mis saadetakse tagasi Label Studiosse, kus neid näidatakse visuaalsete märgendustega (vt. Joonis 6).

[11] `https://github.com/heartexlabs/label-studio-ml-backend`

- *fit()* - funktsioon kutsutakse välja mudeli treenimisel. Talle antakse väljakutsumisel kaasa kõik seni lõpetatud ülesanded (kus kasutaja peab olema kinnitanud, et märgendused on õiged). Lõime funktsioonid, mis parsivad valmis ülesandeid, loovad andmetest treeningfaili ning peenhäälestavad mudelit, mida järgmisel initsieerimisel märgendaja kasutab.

Joonisel 9 on näidis Label Studio graafilises liideses märgendatud tekstist ning millisel kujul lõpuks treeningandmed välja näevad. Sõnad on tükeldatud sõnedeks vastavalt EstNLTK tokeniseerijale, kuid edaspidi tükeldavad ka M. Perli poolt löödud töövood need vastavalt EstMedBERT mudeli tokeniseerijale edasi. Reserveerisime enda kasutatud mudelites 100 spetsiaalset sõne initiseerimata väärtustega mudelile, et me saaksime regulaaravaldiste poolt märgendatud sõnad muuta eraldi sõnedeks. Joonisel muutub nt. kuupäev sõneks *<UNK2>*, sest mudeli täpsuse kontekstis pole väga vahet, mis kuupäevaga tegu on, seega pole mõistlik mudelil lasta igale kuupäevale eraldi väärtus treenida, vaid arvestada kõik kuupäevad omaette sõneks.



Joonis 9. Label Studiost moodustunud treeningandmed

Selle jaoks, et tagaliidese töövoos peenhäälesatud mudelit kasutada nimeolemite märgendamiseks, lõime me *BertNerTagger* klassi, mis pärib EstNLTK *Tagger* klassi. See tähendab, et kui töövoo kaudu on valminud piisavalt hea täpsusega mudel, saab kasutaja võtta soovitud peenhäälestatud EstMedBERT mudeli iteratsiooni ja muus töövoos initsieerida sama klassi *BertNerTagger* märgendaja, mis seda mudelit kasutab.

Samuti lõime töö käigus vajaliku koodi, et valmis oleva mudeli täpsust võrrelda olemasoleva regulaaravaldist kasutava EstNTLK märgendajaga, andes ette andmebaasist päritud positiivsed soovitud nimeolemit sisaldavad tekstid ja negatiivsed. Antud töös väljatoodud tulemused (vt. Lisad 1 ja Joonis 11, 12) on valminud sama koodi abil.

4 Töövoo rakendamine

Viisime märgendaja loomise töövoo simuleerimiseks läbi eksperimendi, mille käigus treenimise TNM-koodide märgendajat kasutades inimsekkumisega lähenemist meie loodud tarkvara peal. Selle jaoks kasutasime M. Perli poolt valminud EstMedBERT mudelit [12], mida me jooksvalt erinevates iteratsioonides peenhäälestasime, kasutasime tekstidel märgenduste ennustamiseks, kontrollisime märgendusi ning jätkasime mudeli peenhäälestamist uuemate andmetega.

4.1 Eksperimendi tingimused

Andmete saamiseks võtsime esmalt STACC'is loodud regulaaravaldist kasutava märgendaja, modifitseerisime seda (algselt korjas see ka muid klassifikatsioone, mida me antud eksperimendis ei kasuta) ning otsisime selle abil Eesti Geenivaramu[12] diagnoositekstidest positiivseid näiteid, kus esines TNM-kood ning negatiivseid, kus ei esinenud. Kasutatav regulaaravaldis nägi välja järgmine:

r"[p|c|r|y]{0,4}?T(x|X|o|O|([a-cA-C])?)).{0,2}[p|c|r|y]{0,4} ?N(x|X|o|O|([a-cA-C])?)) .{0,2}[p|c|r|y]{0,4} ?M(x|X|o|O|([a-cA-C])?))"

Moodustasime treeningandmeteks 1:10 suhtega 500 ülesannet, st. iga positiivse näite kohta, kus esines TNM-kood, oli 10 negatiivset. See suhe oli meie poolt valitud, et mingil määral säilitada reaalsetes andmetes TNM-koodide esinemise osakaalu ning et mudel ei saaks liiga palju positiivseid näiteid.

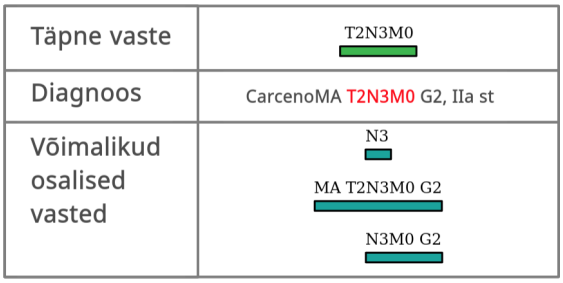
Seejärel seadsime üles Tartu Ülikooli HPC [21] (*High Performance Computing*) peal Label Studio ja meie poolt loodud tagaliidese, mis antud eksperimendis kasutas ainult ühte võimaliku NER märgendiklassi "TNM", importisime sisse andmed ning alustasime märgendamist. Peenhäälestasime iga 50 märgenduse pealt uue mudeli ning kasutasime valminud mudelit edasiste märgenduste ennustamiseks, et protsessi kiiremaks teha.

Võrdlesime valminud peenhäälestatud mudelite sooritust testandmestikul, milleks oli 10'000 erinevat diagnoositeksti, mida ükski mudel polnud veel näinud ja mis olid sama positiivsete ja negatiivsete näidete suhtega kui treeningandmedki (1:10). Kontrollisime treenitud märgendaja sooritust testülesannete peal, võttes alustõeks olemasoleva eelnevas sektsioonis väljatoodud regulaaravaldist kasutava märgendaja. Oluline tähelepanek siinkohal on see, et ka alustõde pole tegelikult täpne ning ei leia üles kõiki positiivseid näiteid korrektselt.

[12] `https://geenidoonor.ee/geenivaramu`

Kuna meie loodud ja treenitud märgendaja märgendab spetsiifiliselt enda tokeniseerija järgi tükeldatud sõnesid ja regulaaravaldis leiab tekstist vastava mustri alguse ja lõpu, ei pruugi need tihti üks-ühele ühilduda. Näiteks kui tekstis oleks "Kasvaja T2 N3 M0", võiks leida meie märgendaja iga tähe klassifikatsiooni eraldi ("T2", "N3", "M0"), kuid regulaaravaldis võtaks selle kui tervikuna. Seetõttu eeltöötlesime enne tulemuste kontrollimist meie märgendaja leitud sõnede piire ning kombineerisime ühese vahega sõnepiirid kokku üheks (st. arvestasime näiteks [0, 1], [2, 4] -> [0, 4]).

Seejärel lugesime kokku, palju oli täpseid vasteid, ehk kus peenhäälestatud EstMedBERT mudelit kasutava märgendaja piirid ühtisid üks-ühele regulaaravaldist kasutava märgendaja piiridega ja palju esines osalisi klappe, kus piirid kattusid, kuid mitte täpselt (vt. Joonis 10). Samuti loendasime ka kokku ka valepositiivsed vasted (*false positive*), mida arvestasime juhtudeks, kui piirid ei ühildunud üldse, ja valenegatiivsed vasted (*false negative*), kui ükski piir ei leidnud koodi üles. Lugesime tõesteks positiivseteks (*true positive*) nii täpseid vasteid kui ka osalisi sellepärast, et väga tihti tuli erinevus fundamentaalsest erinevusest tokeniseerija ja regulaaravaldise abil märgendamisest (nt. kui TNM-koodile järgnes punkt või muu klassifikatsiooni kood) ning reaalses töövoos peaks ilmselt paratamatult efektiivsuse pärast kasutama reeglipõhist filtreerimist ja kontrollima vaste piire, mida on väga lihtne teha, kui *BertNerTagger* on juba leidnud korrektse klassifikatsiooni asukoha üles.



Joonis 10. Erinevate vastete näidis

Arvutasime ka antud tulemuste põhjal täpsuse (*precision*), saagise (*recall*) ja F1 skoori.

$$Täpsus = \frac{TP}{TP + FP} \tag{1}$$

$$Saagis = \frac{TP}{TP + FN} \tag{2}$$

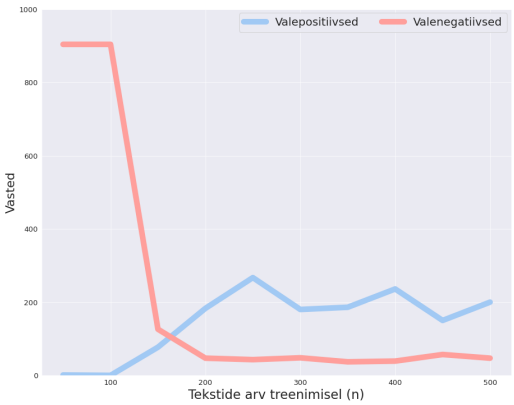
$$F1 = \frac{2 * Täpsus * Saagis}{Täpsus + Saagis} \tag{3}$$

4.2 Eksperimendi tulemused

Tulemustest loeme välja, et kuigi esimese 100 ülesande peal peenhäälestatud mudel ei julenud veel ennustusi anda, siis järgnevaid märgendatud treeningandmeid lisades õppis märgendaja väga kiiresti ülesande ära (vt. Joonis 11, kogu tulemuste tabel on lisades).

Joonis 11. Peenhäälestatud mudelite tulemused erinevates iteratsioonides

Nagu ka eelnevalt mainitud, tuleb tähele panna, et kuna oleme võtnud alustõeks regulaaravaldisel põhineva märgendaja, mis ei ole kaugeltki perfektne, siis tulemused ei ole 100% täpsed. Näiteks käisime läbi viimase mudeli (500 teksti peal peenhäälestatud versiooni) puhul testandmetes valepositiivsed juhtumid (mida oli 200) ning leidsime, et 38 neist olid tegelikult õiged. See tähendab, et need olid juhtumid, kus meie poolt



Joonis 12. Valepositiivsed ja valenegatiivsed vasted

treenitud *BertNerTagger* leidis TNM-koodi korrektselt üles, kuid regulaaravaldis ei saanud sellega hakkama. Üldiselt olid need juhtumid, kus keset klassifikatsiooni esines ka muid sümboleid või väga spetsiifilisi täpsustusi (nt. "pT1b pNo(SN) MOG1") või siis juhtumid, kus oli olemas ainult osa klassifikatsioonist (nt. "kartsinoid G1pT2.").

Üldpildis tundub, et *BertNerTagger*'il polnud antud ülesande jaoks üldsegi palju andmeid vaja, ilmselt sellepärast, et ülesanne on suhteliselt kerge - klassifitseerida on vaja väga spetsiifilisi sõnesid, st. peaaegu alati on olemas tähed T, N ja M ning seetõttu on ka neid sisaldavatele sõnedele omistatud kõrge olulisus ja neid ei esine ilmselt peaaegu kunagi muus kontekstis kõrvustikku. Vaadates nii F1 skoore (vt. Joonis 11) kui ka valepositiivsete ja valenegatiivsete vastete koguseid (vt. Joonis 12), näeme, et põhimõtteliselt piisas juba 200-300 tekstist, et märgendaja töötaks enamvähem usaldusväärselt. Edasiste märgenduste puhul läks täielike vastete osakaal koguvastetes suuremaks, ehk märgendaja õppis kitsamalt õigeid tulemusi leidma. Valenegatiivsed jäid läbi iteratsioonide peale 200 teksti peal peenhäälestamist sama madalaks, kuid valepositiivsete arv, mis meid väga ei häiri, kuna neid saab lihtsalt reeglipõhiselt lõplikult kontrollida, siiski mingil määral jäi kõikuma.

Kogu protsess, et saada kätte eksperimendi viimane peenhäälestatud mudel, ei võtnud kasutaja poolt märgatavalt aega. Peale 100 märgendusega mudeli peenhäälestamist oli märgendaja juba piisvalt täpne, et kasutaja sai mugavalt ülesanded üle käia ja ainult mõningaid parandusi teha. Probleemid esinesid tarkvaralisel küljel, st. Label Studio ML tagaliidese kood oli kohati optimiseerimata (nt. lõi uue mudeli iga ennustuse peale), ilmselt, kuna ei ole eeldatud, et väga mahukate mudelitega seda treenitaks. EstMedBERT

enda treenimine ei võtnud HPC peal väga oluliselt kaua aega ja sai igal juhul alla minutiga treenitud.

5 Diskussioon ja tuleviku perspektiivid

Kuigi eksperimendi eesmärk oli meid veenda, et antud lähenemine on võimalik, milles me ka veendusime, siis tegelikult olid selle tingimused võrdlemisi ebaefektiivsed ning reaalseks töövooks *BertNerTagger*'ite arendamiseks oleks vaja palju asju muuta. Esiteks, valisime sättetest aja kokkuhoiu mõttes võrdlemisi odavad treeningparameetrid (nt. kasutasime ainult ühte epohhit (*epoch* peenhäälestamisel). Reaalseks töövooks kasutamisel peaks ilmselt katsetama erinevaid lähenemisi ja hüpertuunima treeningparameetreid. Samuti märgendasime me üsna vähe tekste ning jooniselt 11 on näha, et viimasel iteratsioonil tegid täielikud vasted hüppe (422 -> 555), seega on põhjust arvata, et jätkamisel võiks mudelid veel täpsemaks minna. Muuhulgas valisime treenimiseks tekstid pigem naiivse lähenemisega. Näiteks ei pruugi meil olla raskemate nimeolemite märgendamise jaoks olemas sarnaselt eksperimendile esialgset regulaaravaldisepõhist märgendajat ning peaksime alustama naiivsemast märgendajast. Sellisel juhul ei oleks isegi niivõrd oluline, et me palju negatiivseid näiteid saame, kuid kindlasti oleks oluline, et meil mingeid väga erilisi erandjuhtumeid kaduma ei läheks, mistõttu võib mudelil olla neid raskem õppida, kui tal treeningandmetes neid pole (kuid mitte võimatu, kuna BERT võtab ka konteksti arvesse). Tulevikus peaks ilmselt sättima üles konkreetsemad alustingimused märgendaja treenimise alustamiseks, sh. täpsemad treeningparameetrid, kindel viis, kuidas esialgseid andmeid hankida jms.

Olles optimiseerinud treenimistingimused, peaks katsetama erinevate märgendajate arendamist ning võrdlema neid regulaaravaldist kasutavate märgendajatega. Üheks oluliseks mõjutajaks on ka aeg, mis sõltub olemasolevast jõudlusest - kui *BertNerTagger* on parema saagise ja täpsusega, ei pruugi see ikka olla väärt kasutamist, kui tekstide läbikäimine sellega võtaks olemasoleva jõudlusega üüratult kaua aega. Samuti peaks uurima, kuidas *BertNerTagger* saab hakkama andmetega, mille peal ta pole treeninud. Üheks suurimaks eeliseks närvivõrkude kasutamisel märgendamisel üle reeglipõhiste lähenemiste on kontekstiga arvestamine. Näiteks kui me otsiks raviminimesi mõlema märgendajaga, mis ei ole konstantne list (st. uusi ravimeid võib juurde tulla), siis reeglipõhise lähenemisega ei saaks me ilmselt uuemaid kätte, kuid *BertNerTagger*'iga võiksime saada.

Suur osa praktilise töö ajast kulus meil tarkvaratehnilisele aspektile, st. Label Studio ja selle tagaliidese tehniliste töövoogude arendamisele. Töö tegemise käigus tuli mõlemal välja uus versioon, mis muutis tagaliidese ja esiliidese vahelist suhtlust, kuid selle käigus muutusid meie jaoks olulised funktsionaalsused. Kasutame enda töös mõlema programmi vanemaid versioone, et tagada töökindlus. Tulevikus peaks kindlasti vähemalt kaaluma ka nullist enda taristu loomist märgendaja treenimisteks, kuna mõne funktsionaalsuse järgi ei tundu, et Label Studio on just ideaalne koht meie ülesandeks. Näiteks kui ükshaaval ennustusi genereerida, initsieeritakse mudel iga ennustuse jaoks uuesti, mis on

üsnagi jõudlust nõudev suurte mudelite treenimisel. Enda loodud keskkonna puhul saaks me ka meie jaoks sobilikke funktsionaalsuseid lisada, näiteks kaasa anda märgendatud testandmestiku ning jooksvalt iga mudeli puhul selle täpsust kontrollida, et paremini hinnata, kui hea iga mudeli iteratsioon on jms.

6 Kokkuvõte

Töö käigus valmis Label Studio tagaliides, mis lubab kasutajal inimsekkumisega peenhäälestada EstMedBERT mudelit, et treenida soovitud infoeraldustöövooks vastav nimeolemite märgendaja. Lisaks toetab meie arendatud tagaliides ka EstNLTK märgendajate kasutamist, muutes nende leitud märgendused unikaalseteks sõnedeks, mis lihtsustab mudeli treenimist. Meie tagaliite abil peenhäälestatud mudelit saab ka edaspidi integreerida valminud EstNLTK märgendaja näol infoeraldustöövoogudesse. Viisime ka läbi eksperimendi, et veenduda töövoo funktsionaalsustes ning analüüsisime tulemusi. Leidsime, et juba 200 märgendatud kirje puhul saavutas mudel 0.90 F1 skoori positiivse märgenduse suhtes ning leidis nimeolemeid, mida esialgne regulaaravaldis ise kätte ei saanud. Valminud lahendust saab täienduste järel kasutada meditsiinitekstidest informatsiooni eraldamiseks töövoogude puhul, kus on vaja teha üldistusi, mida reeglipõhiste lähenemistega ei ole võimalik teha. Töö käigus valminud koodirepositoorium on kättesaadav Tartu Ülikooli GitLab repositooriumis - `https://gitlab.cs.ut.ee/shuva/labelstudio-backend-with-taggers`.

Viidatud kirjandus

[1] e-Estonia. E-health - estonian digital solutions for europe. https://e-estonia.com/e-health-estonian-digital-solutions-for-europe/, 2016. (17.05.2022).

[2] Personalized Medicine Coalition. The Personalized Medicine Report. https://www.personalizedmedicinecoalition.org/Userfiles/PMC-Corporate/file/The-Personalized-Medicine-Report1.pdf, 2017. (17.05.2022).

[3] Rasmus Maide. Eesti keele nimeolemite märgendaja analüüs ja parandamine. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69982, 2020.

[4] James H. Martin Daniel Jurafsky. Speech and Language Processing. Chapter 2: Regular Expressions, Text Normalization, Edit Distance. https://web.stanford.edu/~jurafsky/slp3/2.pdf, 2021. (17.05.2022).

[5] Zihan Liu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. Ner-bert: A pre-trained model for low-resource entity tagging. *arXiv preprint arXiv:2112.00405*, 2021.

[6] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. A survey of human-in-the-loop for machine learning. *arXiv preprint arXiv:2108.00941*, 2021.

[7] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 08 2007.

[8] Shanshan Zhang, Lihong He, Slobodan Vucetic, and Eduard Dragut. Regular expression guided entity mention mining from noisy web data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1991–2000, 2018.

[9] Yunpeng Zhao and Ji Liu. Human-in-the-loop based named entity recognition. In *2021 International Conference on Big Data Engineering and Education (BDEE)*, pages 170–176, 2021.

[10] Chaojie Wen, Tao Chen, Xudong Jia, and Jiang Zhu. Medical Named Entity Recognition from Un-labelled Medical Records based on Pre-trained Language Models and Domain Dictionary. *Data Intelligence*, 3(3):402–417, 09 2021.

[11] Jie Yang, Liqin Wang, Neelam A Phadke, Paige G Wickner, Christian M Mancini, Kimberly G Blumenthal, and LiŽhou. Development and validation of a deep learning model for detection of allergic reactions using safety event reports across hospitals. *JAMA network open*, 3(11):e2022836–e2022836, 2020.

[12] Meelis Perli. Esindusõpe vabatekstilistel meditsiinilistel andmetel. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=72225, 2021.

[13] Aicha Ghoulam, Fatiha Barigou, and Ghalem Belalem. Information extraction in the medical domain. *Journal of Information Technology Research*, 8:1–15, 04 2015.

[14] Kristi Krebs, Jonas Bovijn, Neil Zheng, Maarja Lepamets, Jenny C Censin, Tuuli Jürgenson, Dage Särg, Erik Abner, Triin Laisk, Yang Luo, et al. Genome-wide study identifies association between hla-b 55: 01 and self-reported penicillin allergy. *The American Journal of Human Genetics*, 107(4):612–621, 2020. vt. Lisad.

[15] National Cancer Institute. Cancer Staging. https://www.cancer.gov/about-cancer/diagnosis-staging/staging, 2015. (17.05.2022).

[16] Chaojie Wen, Tao Chen, Xudong Jia, and Jiang Zhu. Medical named entity recognition from un-labelled medical records based on pre-trained language models and domain dictionary. *Data Intelligence*, 3(3):402–417, 2021.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Eesti keele ühendkorpus 2017. https://doi.org/10.15155/3-00-0000-0000-0000-071E7L, 2021. (17.05.2022).

[19] Hasan Tanvir, Claudia Kittask, Sandra Eiche, and Kairit Sirts. Estbert: A pretrained language-specific bert for estonian. *arXiv preprint arXiv:2011.04784*, 2020.

[20] Heartex. Label Studio for Hugging Face's Transformers. https://github.com/heartexlabs/label-studio-transformers. (17.05.2022).

[21] University of Tartu. Ut rocket, 2018.

Lisad

I. Eksperimendi tulemused

n	EXACT_M	PARTIAL_M	FP	FN	Precision	Recall	F1
50	0	0	1	904	0.000000	0.000000	NaN
100	0	0	0	904	NaN	0.000000	NaN
150	363	459	77	126	0.914349	0.867089	0.890092
200	370	581	183	47	0.838624	0.952906	0.892120
250	382	580	267	43	0.782750	0.957214	0.861235
300	398	536	180	48	0.838420	0.951120	0.891221
350	435	541	186	37	0.839931	0.963475	0.897471
400	466	495	236	39	0.802840	0.961000	0.874829
450	422	485	150	57	0.858089	0.940871	0.897575
500	555	354	200	47	0.819657	0.950837	0.880387

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Hendrik Šuvalov**,

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

   Raamistik närvivõrgupõhiste infoeraldustöövoogude loomiseks,

   mille juhendaja(d) on Dage Särg, Raivo Kolde ja Sven Laur,

   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.

Hendrik Šuvalov
17.05.2022