Tartu Ülikool
Loodus- ja täppisteaduste valdkond
Matemaatika ja statistika instituut

Mihkel Lepson

Epikriisi tekstide genereerimine GPT-2 mudeliga

Matemaatilise statistika eriala

Magistritöö (30 EAP)

Juhendaja: PhD. Raivo Kolde

Tartu 2023

EPIKRIISI TEKSTIDE GENEREERIMINE GPT-2 MUDELIGA

Magistritöö

Mihkel Lepson

Lühikokkuvõte

Vabatekstiliste terviseandmete analüüsimisel ja kasutamisel on palju piiranguid, sest täielikult anonümiseerida on neid võimatu. Õpetamiseks ja andmetöötlusmetoodikate väljatöötamiseks ei pea aga kasutama ilmtingimata päris andmeid, piisaks ka genereeritud sünteetilistest andmetest. Magistritöö eesmärk on treenida generatiivne tekstimudel, mis võimaldab genereerida epikriisi tekste vastavalt etteantud dokumendi osale, patsiendi demograafilistele andmetele ja diagnoosile. Töös treenitakse GPT-2 *small* mudel Tartu Ülikooli Eesti geenivaramuga liitunud patsientide epikriisi tekstidel. Saadud mudelil leitakse parim genereerimise algoritm, näidatakse, et genereeritud tekste on võimalik kasutada klassifitseerimismudeli treenimisel ning näidatakse, et on võimalik hinnata genereeritud tekstide originaalsust.

CERCS teaduseriala: P160 Statistika, operatsioonianalüüs, programmeerimine, finants- ja kindlustusmatemaatika.

Märksõnad: Loomuliku keele töötlus, tehisõpe, tehisnärvivõrgud, infoteooria.

EPICRISIS TEXT GENERATION WITH GPT-2

Master thesis

Mihkel Lepson

Abstract

There are a lot of limits when analyzing or using the unstructured health record data, because it is impossible to fully anonymize them. For teaching

and other scientific purposes there is actually no need to use real data as the synthetic data would be sufficent. The aim of the master thesis is to train a generative machine learning model which would allow to generate such texts. In this work a GPT-2 small model is trained on epicrisis texts. The data consists of all the patients who have joined with Estonian Biobank. On the trained model the best decoding method is found which is used to generate the dataset. On the dataset it is shown that the control mechanism to guide text generation works. Also, the ability to train a classification model is shown. Finally, it is shown that there is a way to assess each generated text's originality by comparing it with the training set.

CERCS research specialisation: P160 Statistics, operations research, programming, financial and actuarial mathematics.

Key Words: Natural language processing, machine learning, neural networks, information theory.

Sisukord

Sissejuhatus

Suur osa terviseandmetest on olemas vaid vabatekstina ning selle info kasutamine analüüsides, ennustusmudelites ja erinevates rakendustes nõuab põhjalikku tekstitöötlust. Tekstitöötlusmeetodite arendamine aga on komplitseeritud andmekaitse nõuete poolt, sest selliseid tekste on automaatsel võimatu lõpuni anonümiseerida. Andmekaitse piirangud aga tähendavad, et pole võimalik andmeid jagada uurimisgruppide vahel, luua võrldus andmebaase märgendatud tekstidest, kasutada avalikke tööriistu kuhu andmeid üles laadida ja kasutada neid andmeid õpetamisel. Üks lahendus on tekstide automaatne genereerimine, mis garanteerib, et tulemuseks saadavad tekstid ei sisalda isikustatavat informatsiooni. Oluline selle juures on, et tekstid oleks võimalikult realistlikud, kasutades õiget terminoloogiat, süntaksit ja struktuuri ning ideaalis oleks sisuliselt koherentsed. Kuna antud juhul pakuvad huvi eesti keelsed terviseandmed mida hoitakse epikriisides, siis edaspidi keskendutakse just sellele formaadile.

"Haigusloo epikriisi osa (edaspidi epikriis) vormistatakse iga haiglaravil viibiva patsiendi kohta. Epikriis on kokkuvõtlik väljavõte haigusloost, milles kajastub kõnealuse haigusjuhu dünaamika, lähtudes arsti käsutuses olevast sellekohasest teabest. (Sotsiaalministeerium, 2016)" Epikriisi vabatekstiline osa võib sisaldada järgnevat: diagnoosi põhjendus, haiguse kulu iseloomustus, kokkuvõte tehtud uuringutest/-protseduuridest, kokkuvõte patsiendi ravist, patsiendi terviseseisundi hinnang haiglast väljakirjutamisel, režiimi- ja ravialased soovitused edasiseks raviks, soovitused taastusraviks, terviseseisundist tingitud töökorralduse või töökeskkonna ajutise või alalise muutmise vajadusel selle põhjus, sisu, kestus (Sotsiaalministeerium, 2016). Nagu loetelust näha, siis epikriisi vabatekstiline osa sisaldab lisainformatsiooni, mida on võimalik kasutada näiteks ennustusmudelite loomisel.

Tekstide genereerimiseks on vaja luua keelemudel [1]. Teksti genereerimiseks on kõi-

[1]Funktsiooni mis sisseantud teksti põhjal väljastab selle tõenäosuse, et antud tekst pärineb kindlast keelest, nimetatakse keelemudeliks (Koehn, 2009).

ge populaarsemad ja tõhusamad *transformer* tüüpi mudelid (Liu *et al.*, 2018; Radford *et al.*, 2019). Töös treenitakse *transformer* tüüpi mudel *generative pre-trained transformer* 2 (*generative pre-trained transformer* edaspidi GPT). Mudeliks valitakse just GPT-2, kuna see on näidanud varasemalt häid tulemusi inglise keelsete meditsiinitekstide genereerimisel (Amin-Nejad, Ive ja Velupillai, 2020). Mudeli treenimisel kasutatakse kõigi Tartu Ülikooli Eesti geenivaramuga liitunud patsientide epikriisi anonümiseeritud tekste. Anonümiseeritud tekstides on arsti ja patsiendi nimed asendatud koodidega, mistõttu ei ole tekste võimalik pärisinimesega siduda.

Peale mudeli treenimist näidatakse, et saadud mudeliga on võimalik genereerida kvaliteetseid tekste, mis on sarnased päristekstidele ning mille abil on võimalik treenida päristekstide klassifitseerija. Lisaks sellele on genereeritud tekstide puhul vaja näidata, et genereeritud tekstid on originaalsed ning ei ole lihtsalt treeningandmestiku koopia. Genereeritud tekstide kohta on uuritud erinevaid omadusi nagu võimalust kasutada neid treeningandmetena (Amin-Nejad, Ive ja Velupillai, 2020), võimekust genereerida inimtekstiga sarnast teksti (Holtzman *et al.*, 2019), kuid mida ei ole töö autori teada uuritud on genereeritud tekstide originaalsus. Genereerides tekste tekib küsimus, kui originaalsed on genereeritud tekstid? Originaalsuse hindamisel tekib samuti küsimus, et mis hetkest lugeda genereeritud tekst originaalseks. Kui lause on 30 sõna pikk ning leidub päris tekst kus ainult 15. ja 16. sõna on erinevad, kas siis genereeritud tekst on originaalne või mitte? Samuti on küsimus selles, kui genereeritud tekst on ühesugune näiteks treeningandmestikus esineva lühikese ja mitte unikaalse lausega nagu "Patsient soovib uut retsepti", siis kas sellisel juhul lühikesi ja üldiseid tekste võiks jagada või mitte? Antud küsimustele vastuse andmine ei ole võimalik. See-eest näidatakse, et iga genereeritud teksti puhul on võimalik arvutada arvuline väärtus selle unikaalsuse kohta, mille põhjal saab teha järelduse iga genereeritud teksti originaalsuse osas.

1 Keelemudel ja tekstide genereerimine

Käesolevas peatükis antakse ülevaade, kuidas toimub teksti töötlus enne keelemudelit, milliseid eeldusi tehakse keele matemaatiliste omaduste kohta ning kuidas *transformer* tüüpi keelemudel töötab. Samuti antakse teoreetiline ülevaade meetoditest, mida kasutatakse keelemudeli hindamisel.

1.1 Tokeniseerija

Token tähistab üksikut tähte või tähtede kombinatsioon ehk sõna või alamsõna (Sennrich, Haddow ja Birch, 2015). Mudel nagu *transformer* saab vaadata limiteeritud koguses tokeneid ning tokenite kogumit, mida mudel kasutab, nimetatakse sõnastikuks (Vaswani *et al.*, 2017). Tokeniseerija ülesanne on viia tekst tokeniteks, mida saaks ette anda mudelile (Mielke *et al.*, 2021). Teksti tokeniseerimisel on kolm etappi: normaliseerimine, tokeniseerimismudel ja järeltöötlus (Hugging Face, 2023a).

1.1.1 Normaliseermine

Normaliseerimise eesmärk on tekstide ühtlustamine ning võimalikke kombinatsioonide vähendamine (Hugging Face, 2023a). Näiteks kui esinevad laused "Patsient on haige" ja "patsient on haige", siis mõlemal juhul on tähendus sama, kuid arvuti jaoks on sõnad "Patsient" ja "patsient" erinevad. Lausete ühtlustamise eesmärgil teostatakse tekstide normaliseerimine. Eeltoodud näites kui viia sõna "Patsient" kujule "patsient", ehk asendatakse suurtäht väiksega, siis sellisel juhul on tegu samade lausetega. Samuti läheb normaliseerimise alla liigsete tühikute ja kirjavahemärkide eemaldamine (Hugging Face, 2023a).

1.1.2 Tokeniseerimismudel

Byte-Pair Encoding (BPE) algoritm pärineb aastast 1994 ja loodi teksti kompresseerimise eesmärgil (Gage, 1994). Algoritmis leitakse kõige sagedasem baiti paar, mis esineb rohkem kui ühe korra, ja ühendatakse kokku. Kahe baiti ühendust tähistatakse sellisel juhul uue baitiga, mida sisendis ei esinenud. 2016. aastal kompresseerimise algoritmist inspireeritult loodi BPE tokeniseerija, mis töötab sarnasel põhimõttel (Sennrich, Haddow ja Birch, 2015). BPE tokeniseerija abil sõnastiku ehk tokenite kogumi loomine käib järgnevalt. Alguses luuakse sõnastik, mis sisaldab kõiki tähemärke ehk baastokeneid, mida on 256, ja eritokeneid (Hugging Face, 2023b). Eritokenid võivad olla näiteks lauselõpu ja lause algus token. Näiteks GPT-2 puhul on lause algus ja lauselõpu token samad ning on tähistatud sõnaga "<|endoftext|>" (Radford *et al.*, 2019). Seejärel treeningandmestikus sõnad eraldatakse lausetest ning sõnad tükeldatakse tähtedeks. Peale tükeldamist leitakse kõige sagedasem tähepaar. Oluline on märkida, et järjekord on siin oluline. Kui kõige sagedasem paar on leitud, siis kõikides sõnades, ühendatakse paar kokku ja sõnastikku lisatakse uus token. Kui kõige sagedasem paar tähemärkide kujul oli ("a","b"), siis sõnastikku lisatakse uus token mis tähistab "ab". Antud protseduuri tehakse, kuni saavutatakse soovitud sõnastiku suurus või kuni ühtegi ühendamist pole enam võimalik teha. Sõnastiku loomise täpsemat algoritmi on näha Lisa 1. BPE tokeniseerijaga sõnastiku loomise algoritm. Hiljem tokeniseerides kasutatakse õpitud ühendusi nagu täpsemalt on näha Lisa 2. Näide BPE tokeniseerijaga tokeniseerimisest. BPE tokeniseerija valik seisneb selles, et seda kasutasid GPT-2 loojad (Radford *et al.*, 2019).

1.1.3 Kärpimine ja *Padding*

Mudel nagu *transformer* nõuab fikseeritud pikkusega d_n sisendit (Vaswani *et al.*, 2017). See-eest iga tekst ei ole alati peale tokeniseerimist d_n tokenit pikk. Sellisel juhul kasutatakse sisendteksti kärpimist ja *padding*'ut (Hugging Face, 2023c). Kui

tekst on pikem kui d_n tokenit, siis teostatakse sisendteksti kärpimine, ehk üleliigsed tokenid eemaldatakse. Kui peale tokeniseerimist tekst on lühem kui d_n tokenit, siis teostatakse *padding*. Sellisel juhul lisatakse tekstile juurde *padding* token, kuni saavutatakse d_n pikkus. *Padding* tokenina saab kasutada näiteks lauselõpu tokenit (Radford *et al.*, 2019) või sõnastikku lisatakse eritoken mis tähistab *padding*'ut.

1.2 Keelemudel

Kui vaadata keelemudeleid (Radford *et al.*, 2018; Keskar *et al.*, 2019; Devlin *et al.*, 2018; Bahdanau, Cho ja Bengio, 2014), siis kõik põhinevad Bengio, Ducharme ja Vincent, 2000 välja pakutud ideel. Esiteks eeldatakse, et teksti statistilise mudeli saab esitada tinglike tõenäosuste korrutisena. Sellisel juhul saab kirjutada teksti tõenäosuse järgnevalt

$$P(x_{1:T}) = \prod_{t=1}^{T} P(x_t|x_{1:t-1}), \tag{1}$$

kus $x_{1:t-1} = (x_1, x_2, \ldots, x_{t-1})$ ning x_i tähistab lauses i'ndal positioonil olevat tokenit. Ülesande lihtsustamiseks tehakse eeldus, et tegu on d_n järku Markovi ahelaga ehk $P(x_t|x_{1:t-1}) \approx P(x_t|x_{t-d_n:t-1})$.

Teiseks, mudelile ei anta ühe tokeni kohta ette *one-hot* vektor, vaid õpitakse $\mathbb{R}^m$ tunnuste vektor, kirjanduses *embedding*. *One-hot* vektoris on element mille indeks on tokeni väärtus 1 ning mujal on 0. *Embedding*'u idee seisneb sellest, et kui sõnastiku suurus on 50257, siis tegu on liiga kõrge dimensioonilise ruumiga ning samuti on tokenite kaugus üksteisest alati sama kui võrrelda erinevaid tokeneid. Esitates tokenid ruumis $\mathbb{R}^m$, saab kasutada erinevaid kaugusi kahe tokeni võrdlemiseks. Üks võimalus kahte sõna võrrelda ruumis $\mathbb{R}^m$ on kasutada skalaarkorrutist (Vaswani *et al.*, 2017).

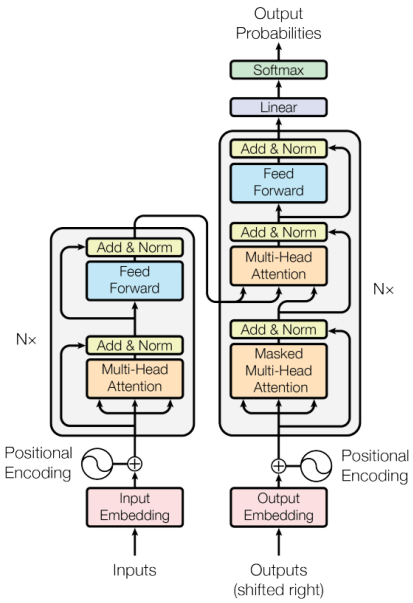
Viimaseks, transformatsiooni, mis viib tokenid tunnuste vektoriks, ja mudeli treenimine käib samaaegselt.

1.3 Transformer ja generative pre-trained transformer

Transformer arhitektuuri pakuti välja 2017. aastal artiklis "*Attention is all you need*" (Vaswani *et al.*, 2017) ja on peale seda edukalt lahendanud erinevaid loomuliku keele töötluse ülesandeid (Kalyan, Rajasekharan ja Sangeetha, 2021). Esialgne *transformer* loodi masintõlke eesmärgil ning koosnes kahest komponendist, *encoder* ja *decoder* osast, nagu on näha Joonisel 1. Sellisel juhul on *encoder*'i sisendiks tõlgitav tekst ning *decoder* osas on tõlgitud tekst. Vastavalt vajadusele on loodud eri tüüpe mudeleid, mis kasutavad ainult *encoder* osa (Devlin *et al.*, 2018) või ainult *decoder* osa (Liu *et al.*, 2018; Radford *et al.*, 2018; Keskar *et al.*, 2019). Kõigis *decoder* tüüpi mudelites on samuti vaadatud või keskendutud tekstide genereerimisele. Samuti on GPT-2 kasutatud inglise keelsete meditsiini tekstide genereerimisel, kus antud mudel töötas edukalt (Amin-Nejad, Ive ja Velupillai, 2020). Seetõttu töös vaadatakse täpselt kuidas ainult *decoder* osa sisaldav transformer mudel töötab ning kuidas selle mudeli treenimine käib. Täpsemalt keskendutakse sellele, kuidas GPT-2 loojad rakendasid ainult *decoder*'i sisaldavat mudeli.

1.3.1 Arhitektuur

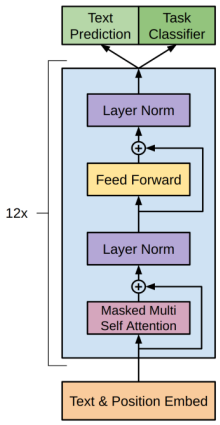
Alapeatükk põhineb kolmel artiklil: "*Attention is all you need*" (Vaswani *et al.*, 2017), "*Improving Language Understanding by Generative Pre-Training*" (Radford *et al.*, 2018) ja "*Language Models are Unsupervised Multitask Learners*" (Radford *et al.*, 2019), väljaarvatud juhul kui tekstidel on viide lisatud. Kui tekstis mainitakse, et midagi õpitakse treenimise käigus, siis selle all mõeldakse stohhastilist gradient laskumist, kus gradiendid on saadud *back-propagation*'i teel. Stohhastilises gradient laskumises arvutatakse mudeliga kadu treeningandmestikust väikse tükki ehk plokki peal. Saadud kao abil optimiseeritakse mudel, leides mudeli parameetrite gradiendid *back-propagation*'i teel ehk ketireegli abil (Goodfellow, Bengio ja Courville, 2016). Arhitektuuri näidet saab näha Lisa 3. GPT-2 osaline rakendamine, kus on tehtud läbi osaliselt mudeli rakendamine fiktiivsete arvudega.



Joonis 1: Transformer arhitektuur (*encoder* vasakul ja *decoder* paremal) (Vaswani *et al.*, 2017)

Transformer mudeli puhul fikseeritakse maksimaalne sisendpikkus d_n, ja *embedding*'u suurus d_{model}. Samuti on mudel seotud tokeniseerijaga ning d_{vocab} on sõnastikku suurus. Suurusest d_n saab mõelda kui Markovi ahela järgust. Mudel, mille sisend pikkus on d_n annab $d_n \times d_{vocab}$ suuruse väljundi, kus ühes reas on järgmise tokeni tõenäosused. Seega väljund maatriksis $d_n \times d_{vocab}$ on reas i iga sõnastikus oleva tokeni tõenäosus olla lauses positsioonil $i + 1$.

GPT arhitektuuri on näha Joonisel 2. Võrreldes esialgset *transformer*'i ainult *decoder*'i sisaldava mudeliga on näha, et GPT puhul on *decoder* osas ära jäetud keskelt *multi-head attention*, sest puudub *encoder* ja sealt informatsiooni ei tule. Vaadates Joonist 2 on näha, et mudeli sisend on *text* & *position embed*, mis on maatriks mõõtmetega $d_n \times d_{model}$, näidet saab näha Lisa 3.1 *Embedding* ja *positional encoding*. Olgu $X = (x_{t-d_n}, \ldots, x_{t-1})$ tokeniseeritud sisendvektori milles on d_n elementi. *Text* & *position embed* saamiseks viiakse kõigepealt vetkor X *one-hot* kujule, mis on maatriks U mõõtmetega $d_n \times d_{vocab}$. *One-hot* kujul olevas maatriksis on i'ndas reas



Joonis 2: GPT arhitektuur (Radford *et al.*, 2018)

positsioonil x_i väärtus 1 ja mujal 0 (Esimene positsioon on indeksiga 0). Saadud maatriks U korrutatakse paremalt maatriksiga $W_e \in \mathbb{R}^{d_{vocab} \times d_{model}}$. Nagu mainitud 1.2 Keelemudel, siis W_e maatriksi väärtused õpitakse treenimise käigus. Antud maatriksist saab mõelda kui funktsioonist, mis viib ühe tokeni ruumi $\mathbb{R}^{d_{model}}$. UW_e on sisendi *embedding* maatriks, kus reas on ühe tokeni *embedding*.

Embedding'ule liidetakse *positional encoding* maatriks W_p, mille eesmärk on anda mudelile infot antud tokeni asukoha kohta lauses. Erinevalt *transformer* autorite poolt pakutud siinuse ja koosinuse funktsioonil põhineval *positional encoding* maatriksist, siis GPT puhul *positional encoding* maatriks W_p õpitakse treenimise käigus. Näidet *embedding*'ust ja *positional encoding*'ust saab näha Lisa 3.1 *Embedding ja positional encoding*. Mida joonisel ei ole näha, kuid mida rakendatakse saadud maatriksile on väljajätumeetod. Väljajätumeetodi korral genereeritakse maatriks B mis on samade mõõtmetega kui maatriks millele seda rakendatakse ehk antud juhul $UW_e + W_p$. Maatriksis B on iga element Bernoulli jaotusest juhuslik suurus ja kui genereeritud väärtus on 0, siis seatakse maatriksis olev väärtus nulliks (Srivastava *et al.*, 2014). Rakendades väljajätumeetodi saadakse maatriks

$$A := (UW_e + W_p) \odot B,$$

kus $\odot$ tähistab Hadamardi korrutist. Bernoulli jaotuse korral antud väljajätumeetodis ning iga järgneva korral on nulli saamise tõenäosus 0,1. Väljajätumeetodi kasutatakse *transformer*'is, kuna see aitab vähendab mudeli ületreenimist.

Peale väljajätumeetodit teostakse kihi normaliseerimine. Seda ei ole küll joonisel näha, sest esialgse GPT puhul seda ei rakendatud sisendile, vaid alles GPT-2 puhul. Normaliseerimise puhul leitakse maatriksis A iga rea keskväärtus ja standardhälve (Ba, Kiros ja Hinton, 2016).

$$\mu^l = \frac{1}{d_{model}} \sum_{i=1}^{d_{model}} A_{l,i}$$

$$\sigma^l = \sqrt{\frac{1}{d_{model}} \sum_{i=1}^{d_{model}} (A_{l,i} - \mu^l)^2}$$

Saadud tulemustega normaliseeritakse rea elemendid järgnevalt

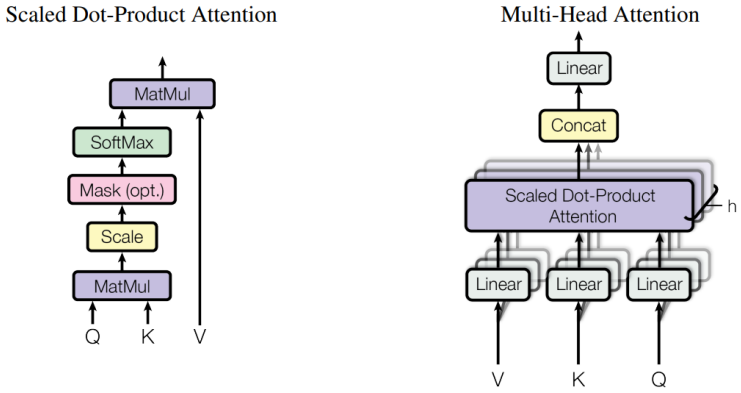
$$\frac{\vec{\mathbf{g}}'}{\sigma^l} \odot (A_{l,\cdot} - \mu^l \vec{\mathbb{1}}') + \vec{\mathbf{b}}'. \tag{2}$$

Valemis 2 $\vec{\mathbf{g}}$ ja $\vec{\mathbf{b}}$ on vektorid, mis õpitakse treenimise käigus ning on iga rea kohta samad (Ba, Kiros ja Hinton, 2016) ning $A_{l,\cdot}$ tähistab maatriksi A l'ndat reavektori. Normaliseerides maatriks A saadakse

$$\text{Norm}(A) = \text{Norm}\begin{pmatrix}\begin{pmatrix} A_{1,\cdot} \\ A_{2,\cdot} \\ \dots \\ A_{d_n-1,\cdot} \\ A_{d_n,\cdot} \end{pmatrix}\end{pmatrix} = \begin{pmatrix} \frac{\vec{\mathbf{g}}'}{\sigma^1} \odot (A_{1,\cdot} - \mu^1 \vec{\mathbb{1}}') + \vec{\mathbf{b}}' \\ \frac{\vec{\mathbf{g}}'}{\sigma^2} \odot (A_{2,\cdot} - \mu^2 \vec{\mathbb{1}}') + \vec{\mathbf{b}}' \\ \dots \\ \frac{\vec{\mathbf{g}}'}{\sigma^{d_n-1}} \odot (A_{d_n-1,\cdot} - \mu^{d_n-1} \vec{\mathbb{1}}') + \vec{\mathbf{b}}' \\ \frac{\vec{\mathbf{g}}'}{\sigma^{d_n}} \odot (A_{d_n,\cdot} - \mu^{d_n} \vec{\mathbb{1}}') + \vec{\mathbf{b}}' \end{pmatrix}$$

Saadud maatriks tähistatakse $H_0 := \text{Norm}(A)$. Väljajätumeetodi ja normaliseerimise näidet saab vaadata Lisa 3.2 Väljajätumeetod ja normaliseerimine.

Kui vaadata Joonist 2, siis on näha, et *decoder*'i esimene kiht on *Masked Multi Self Attention*. Enne kui seletatakse, kuidas *Masked Multi Self Attention* töötab,



Joonis 3: *Attention* (vasakul) ja *Multi-Head Attention* (paremal) (Vaswani *et al.*, 2017)

vaadatakse kuidas *masked attention* töötab. *Masked attention* avaldub valemiga

$$\text{Attention} = \text{softmax}\left(Mask\left(\frac{QK^T}{\sqrt{d_k}}\right)\right)V. \tag{3}$$

Valemist 3 ja Joonisel 3 on näha, et *masked attention* saab kolm sisendit: Q, K ja V, mis tähistab *Queries*, *Keys* ja *Values* maatrikse. Lihtsuse huvides vaadatakse, kuidas *self-attention* töötab. Sellisel juhul $Q = K = V$, mis on eelmise kihi väljund, ehk antud juhul oleks H_0.

Valemist 3 on näha, et esimese sammuna tehakse korrutis QK^T, ehk $H_0H_0^T$, mille tulemus on $d_n \times d_n$ maatriks. Saadud maatriksi elemendid on H_0 ridade skalaarkorrutised. Kuigi esialgse maatriksiga UW_e tehti erinevaid teisenduse, siis on ikkagi suurespildis seal ühes reas tokeni *embedding*. Seega saab korrutisest $H_0H_0^T$ mõelda, kui et igat tokeni sisendis võrreldakse teise tokeniga lauses. Seega on peadiagonaalil tokeni võrdlus iseendaga. Peadiagonaalist üleval on tokeni võrdlus tulevaste tokenitega ja peadiagonaalist allpool on tokeni võrdlus eelnenud tokenitega. Saadud skalaarkorrutiste maatriks skaleeritakse suurusega $\sqrt{d_k}$, kus d_k on maatriksi Q veergude arv ja maatriksi K ridade arv.

Peale skaleerimist rakendatakse *mask*'imist. *Mask*'imise tulemusel sõltub positsioo-

nil x_{i+1} olevad ennustused ainult tokenitest $x_i, x_{i-1}, \ldots, x_1$. Kuna tuleviku tokeneid ei ole näha, siis see tõttu sobib *decoder* osa sisaldav mudel autoregressivseks ülesandeks. Seega edaspidi, kui mainitakse *attention*'i mõeldakse alati *masked* versiooni. Saavutamaks olukorda, kus tuleviku tokenite osakaal on 0, seatakse maatriksis $\frac{1}{\sqrt{d_k}} H_0 H_0^T$ kõik peadiagonaalist ülevalpool olevad väärtused $-\infty$. Väärtus seatakse $-\infty$, kuna peale *mask*'imist rakendatakse *softmax* funktsiooni igale reale. *Softmax* maatriksi rea l elemendi i kohta avaldub järgneva valemiga.

$$\text{softmax}\left(\text{Mask}\left(\frac{1}{\sqrt{d_k}}(H_0 H_0^T)_{l,i}\right)\right) = \frac{\exp(\frac{1}{\sqrt{d_k}}(H_0 H_0^T)_{l,i})}{\sum_{j=1}^{d_{model}} \exp(\frac{1}{\sqrt{d_k}}(H_0 H_0^T)_{l,j})}. \tag{4}$$

Valemist 4 on näha, et väärtus $-\infty$ korral on lugejas eksponent 0, mistõttu jagatis on 0. Rakendades maatriksi igale elemendile *softmax* funktsiooni saadakse maatriks, kus iga rea elementide summa on 1.

Pärast *softmax* funktsioon, korrutatakse saadud maatriks paremalt maatriksiga H_0. Vaadates tulemust, siis vasakul on maatriks, kus rea summa on 1 ehk sellest saab mõelda kui osakaalude maatriksist. Paremal on maatriks, kus reas on tokeni *embedding*. Korrutades sisendmaatriksi vasakult osakaalude maatriksiga, on tulemuseks maatriks, kus reas on sisendmaatriksi ridade lineaarkombinatsioonid. Samuti kuna vasakul oli maatriks, mille peadiagonaalist ülevalpool olevad väärtused on 0, siis on näha, et tulemus maatriksis ei ole ühelegi reale liidetud temast allpool asuvat rida. Saadud tulemusest võibki mõelda kui tähelepanust ehk attention'ist. Asudes positsioonil i ning ennustades positsioonil $i+1$ olevat tokenit, siis kui palju tähelapanu peab pöörama tokenile $x_i, x_{i-1}, \ldots, x_1$.

Eelnev kirjeldas seda, kuidas *attention* töötab. Teades kuidas *attention* töötab, saab vaadata kuidas *Multi Self Attention* töötab. Originaal *transformer*'is ja GPT-2 korral kasutakase *Multi Self Attention*'i, kuna *transformer*'i arhitektuuri autorid leidsid, et parema tulemuse saab, kui *attention*'i rakendatakse sisend maatriksi

veergude lineaarkombinatsioonidele. *Multi Self Attention* on antud järgneva valemiga

$$\text{MultiHead}(H_0, H_0, H_0) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{5}$$

$$\text{head}_i = \text{Attention}(H_0W_i^Q, H_0W_i^K, H_0W_i^V) \tag{6}$$

On näha, et sellisel juhul *Multi Self Attention*'is kui rakendatakse üksikuid *attention* funktsioone, siis sisendid ei ole võrdsed, vaid vaadatakse sisendi erinevaid lineaarkombinatsioone $H_0W_i^Q \neq H_0W_i^K \neq H_0W_i^V$, kus $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_k}$. Kui mõelda, mis toimub lineaarkombinatsioonide võrdlemisel *attention*'is, siis erinevus on selles, et ei võrrelda mitte täispikku *embedding*'uid, vaid selle erinevaid projektsioone ruumis $\mathbb{R}^{d_k}$. Tehes üksik *attention* saadakse sellisel juhul maatriks mõõtmetega $d_n \times d_k$. Näha on valemist 5, et selliseid *attention*'e teostatakse h korda ning $d_k \cdot h = d_{model}$. Valemis 5 olev *Concat* on lühend sõnast *concatenate* ja tähendab, et saadud maatriksid ühendatakse veergu pidi kokku. Ühendades h maatriksit mille dimensioonid on $d_n \times d_k$, saab maatriksi dimensioonidega $d_n \times d_{model}$, mis on samade mõõmetega kui sisendmaatriks H_0. Peale *Concat* võetakse veel saadud maatriksi veergudest lineaarkombinatsioonid ehk korrutatakse maatriksiga $W_i^O \in \mathbb{R}^{d_{model} \times d_{model}}$ paremalt. Samuti peale viimast korrutist maatriksi mõõtmed ei muutu.

Vaadates Joonist 2 on näha, et *Masked Multi Self Attention*'ist paremalt läheb nool mööda. See viitab *residual connection*'ile, mis tähendab, et sisend H_0 mis läks *Masked Multi Self Attention*'isse liidetakse selle väljundile. *Residual connection*'i rakendatakse, kuna on tähele pandud, et see annab suurte kihiarvudega mudelite treenimisel parema tulemuse, kuna leides gradiendid ketireegli abil vähendab see olukordi, kus mudeli varasemates kihtides on gradiendid väiksed (He *et al.*, 2016). Samuti mõlemale liidetavale rakendatakse väljajätumeetodi. Peale liitmist teostatakse kihi normaliseerimine sarnaselt eelnevale. Võttes kaks eelnevat sammu kokku

saadakse tulemus

$$Vahekiht_0 = \text{Norm}(\text{MultiHead}(H_0, H_0, H_0) \odot B_{0|0} + H_0 \odot B_{0|1}),$$

kus $B_{0|0}$ ja $B_{0|1}$ on maatriksid kus elemendid on Bernoulli jaotusest juhuslikud suurused.

Peale normaliseerimist rakendatakse tulemusele ühe peidetud kihiga närvivõrku nagu on näha Joonisel 2. Peidetud kihi dimensioon on 3078 ja väljund kihi dimensioon on $d_{model} = 768$. Ühe peidetud kihiga närvivõrgu rakendamine ei ole muud, kui kaks korda maatriksi veergudest lineaarkombinatsioonide võtmine, kus peale esimest lineaarkombinatsioone rakendatakse sõlm-funtksiooni. Sõlm-funktsioonina kasutatakse GPT-2 puhul *Gaussian Error Linear Unit*'i ehk GELU. GELU avaldub valemiga

$$\text{GELU}(x) = xP(X < x),$$

kus $X \sim \mathcal{N}(0, 1)$ (Hendrycks ja Gimpel, 2016). Arvutamisel kasutatakse GELU hinnangut, sest normaaljaotuse tõenäosusfunktsiooni täpse väärtuse arvutamine on ajaliselt kulukam. Hinnang avaldub valemiga

$$\text{GELU}(x) \approx 0.5x \left(1 + \tanh\left((\frac{2}{\pi})(x + 0{,}044715x^3)\right)\right).$$

(Hendrycks ja Gimpel, 2016). Seega, saab närvivõrgu kirja panna järgnevalt

$$\text{GELU}(Vahekiht_0 W_{0|0} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0})W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1}, \tag{7}$$

kus $\text{GELU}(Vahekiht_0 W_{0|0} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0})$ tähendab, et igale elemendile maatriksis rakendatakse GELU funktsiooni, $W_{0|0}$ on maatriks mõõtmetega $d_{model} \times 3072$, $W_{0|1}$ on maatriks mõõtmetega $3072 \times d_{model}$, $\vec{\mathbf{b}}_{0|0}$ on vektor mõõtmetega 3072 ja $\vec{\mathbf{b}}_{0|1}$ on vektor mõõtmetega d_{model}. Näidet ühe peidetud kihiga närvivõrgust saab näha

Lisa 3.4 Ühe peidetud kihiga närvivõrk.

Peale närvivõrku rakendatakse jälle *residual connection*'i ja kihi normaliseerimist. Eelnevad sammud saab kirja panna järgnevalt

$$H_1 = \text{Norm}((\text{GELU}(Vahekiht_0 W_{0|0} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0})W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1})) \odot B_{0|2} + Vahekiht_0 \odot B_{0|3}).$$

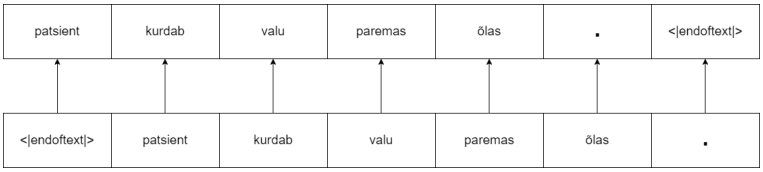
Eeltoodud protsess kirjeldas, kuidas üks *transformer* plokk töötab, mis on Joonisel 2 sinise kastiga ümbritsetud. Saadud plokki rakendatakse GPT-2 puhul 12 korda. See tähendab, et viimasena saadud tulemus H_1 on sisend uude plokki. Viimase ploki väljund on seega H_{12} mis avaldub H_{11} abil järgnevalt

$$\begin{aligned} Vahekiht_{11} &= \text{Norm}(\text{MultiHead}(H_{11}, H_{11}, H_{11}) \odot B_{11|0} + H_{11} \odot B_{11|1}), \\ H_{12} &= \text{Norm}((\text{GELU}(Vahekiht_{11} W_{11|0} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{11|0})W_{11|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{11|1})) \odot B_{11|2} \\ &\quad + Vahekiht_{11} \odot B_{11|3}). \end{aligned}$$

Peale 12. plokki tuleb *text prediction*. See tähendab, et maatriksile H_{12} rakendatakse ilma ühegi peidetud kihita närvivõrku ehk veergudest võetakse lineaarkombinatsioonid. Maatriksit H_{12} korrutatakse paremalt maatriksiga $W_{pred} : d_{model} \times d_{vocab}$: Soov on saada tokenite tõenäosused, siis väljundile rakendatakse softmax funktsiooni, mille tulemusel on väljundid mitte negatiivsed ning iga rea summa on üks. Kui alguses oli sisendvektor $X = (x_1, \ldots, x_{d_n})$, siis väljundimaatriksis on esimeses reas ennustused tokeni x_2 kohta, teades ainult tokeni x_1. Teises reas on ennustused tokeni x_3 kohta, teades ainult tokeneid x_1 ja x_2.

1.3.2 Mudeli treenimine

GPT-2 on sarnase arhitektuuriga, mis sellele eelnenud GPT mudel (Radford *et al.*, 2018; Radford *et al.*, 2019) ja mudeli treenimise protsessi ei ole muudetud. Treenides mudelit on eesmärk mudeliga ennustada järgmist tokenit. Seega on soov, et mudel



Joonis 4: Mudeli treenimine

annaks võimalikult suure tõenäosuse õigele tokenile. Treenides mudelit saab võtta tokeniseeritud teksti ning seda nihutada ühe koha võrra nagu näha Joonisel 4, kus alumisel real on viimane token sisendis ning ülemisel real on järgmine token lauses. Mudelit treenides on eesmärk leida mudeli parameetrid Θ, mis maksimeerivad tõepära

$$L(C) = \sum_i \log P(x_i | x_{i-d_n:i-1}; \Theta). \tag{8}$$

Summa võetakse üle kõigi paaride korpuses, kus $x_{i-d_n} : x_{i-1}$ tähistab tokeniseeritud sisend teksti ja x_i tähistab tokenit, mis tuleb andmestikus peale $x_{i-d_n} : x_{i-1}$ (Radford *et al.*, 2018). Kui korpuses oleks ainult üks lause "patsient kurdab valu paremas õlas.", siis sellisel juhul tuleb summa

$$\begin{aligned} L(C) = &\log P(\text{patsient}| < |\text{endoftext}| >) + \log P(\text{kurdab}| < |\text{endoftext}| >, \text{patsient}) + \\ &\log P(\text{valu}| < |\text{endoftext}| >, \text{patsient}, \text{kurdab}) + \ldots + \\ &\log P(< |\text{endoftext}| > | < |\text{endoftext}| >, \text{patsient}, \text{kurdab}, \text{valu}, \text{paremas}, \text{õlas}, .). \end{aligned}$$

Näites on ka lisatud lause algus ja lauselõpu token. Mudeli treenimine toimub stohhastilise gradient laskumise abil. Sellel põhjusel vaadatakse mudeli treenimise ajal minimiseerimis ülesannet (Dürr, Sick ja Murina, 2020). Minimiseerimis ülesanne saadakse kui maksimiseerimis ülesanne korrutada negatiivse arvuga. Seega vaada-

takse negatiivset log-tõepära

$$-L(C) = -\sum_{i} \log P(x_i | x_{i-d_n:i-1}; \Theta). \tag{9}$$

Positiivse konstandiga korrutamine ei muuda miinimum kohta, siis negatiivne log-tõepära korrutatakse suurusega $\frac{1}{n}$, kus n on paaride arv korpuses, ehk kõigi tokeniseeritud lausete pikkus. Saadakse suurus

$$-\frac{1}{n}\sum_{i} \log P(x_i | x_{i-d_n:i-1}; \Theta), \tag{10}$$

mida tuntakse kui ristentroopiat (Dürr, Sick ja Murina, 2020). Valemist on näha, et ristentroopia on minimaalne ehk 0 kui mudel suudab eelnevate tokenite põhjal täpselt ära ennustada järgmise tokeni ehk anda selle tõenäosuseks 1.

1.4 Kontrollitud genereerimine

Mudeliga on soov suunata genereeimist, et see annaks teksti, mis on kindla stiiliga, omadustega või kindlast domeenist. Teksti genereerimise suunamise eesmärgil tulid Keskar *et al.*, 2019 välja ideega lisada treenimisel teksti algusse võtmesõnad, mis käivad lause kohta. Mudelit treenides on eesmärk minimiseerida järgnevat ristentroopiat

$$-\frac{1}{n}\sum_{i} \log P(x_i | x_{i-d_n:i-1}, c_i; \Theta),$$

kus c_i tähistab teksti i kohta käivat lisainformatsiooni.

1.5 Autoregressiivne genereerimine

Autoregressiivne teksti genereerimine põhineb eeldusel, et lause tõenäosusjaotuse saab kirja panna tinglike tõenäosuste korrutisena

$$P(x_{1:T}) = \prod_{t=1}^{T}(x_t|x_{1:t-1}),$$

(Bengio, Ducharme ja Vincent, 2000). Nagu eelnevalt mainitud, siis tinglikud tõenäosused saab treenitud keelemudelist. Suurus T, mis tähistab lauselõpu indeksit, määratakse jooksvalt ja tuleneb siis, kui $t = T$ hetkel on väljundiks lauselõpu token või saavutakse maksimaalne tekstipikkus mis on ette antud enne genereerimist (Platen, 2020). Järgnevalt vaadatakse algoritme, mida saab kasutada teksti genereerimisel. Järgnevad alapeatükid erinevate teksti genereerimise algoritmide kohta põhinevad allikast (Holtzman *et al.*, 2019).

1.5.1 *Greedy search*

Algoritm *greedy search* võtab igal sammul kõige suurema tõenäosusega sõna. On selge, et sellisel juhul saab sama sisendi korral alati sama väljundi. Probleem algoritmi puhul on ka see, et inimtekst ei ole mudeli järgi alati kõige tõenäolisem, mistõttu *greedy search* toodab kummalist teksti. Samuti rakendades *greedy search*'i võib genereeritud tekst olla korduv, mis tähendab, et algoritm jääb sama tokenit või nende kombinatsiooni kordama.

1.5.2 Juhuslik valik

Juhuslikus valikus võetakse juhuslikult üks token vastavalt keelemudeli antud tõenäosustele. Juhusliku valiku poolt genereeritud tekst ei ole alati kõige kvaliteetsem, kuna madala tõenäosusega sõnadel on tavaliselt liiga suur osakaal. Tõstmaks suurema tõenäosusega tokenite osakaalu, siis mudelist saadud väljundid enne softmax

funktsiooni jagatakse läbi arvuga $t \in (0, 1]$ nagu näha järgnevast valmist

$$softmax(\vec{x})_i = \frac{\exp(x_i/t)}{\sum_{j=1}^{n} \exp(x_j/t)}. \tag{11}$$

Kirjanduses on see meetod tuntud kui *sampling with temperature*. Peale softmaxi, kui tokeni tõenäosus on väike, siis järelikult enne softmaxi, oli selle väärtus negatiivne. Seega jagades läbi positiivse konstandiga $t \in (0, 1]$, muutub väärtus väiksemaks, mistõttu on selle väärtus peale softmaxi väiksem.

1.5.3 Suurima k valik

Selle asemel, et valida üks token juhuslikult kogu väljundist, võib valida hoopis k tokeni seast, millel on kõige suurem tõenäosus. Sellisel juhul valitakse sõnastikust $V^{(k)} \subset V$ kus $V^{(k)}$ on valitud nii, et see maksimiseerib

$$\sum_{x \in V^{(k)}} P(x|x_{1:i-1}) \tag{12}$$

Alles jäänud tokenite tõenäosused skaleeritakse, et nende summa oleks 1. Saadakse, et

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & x \in V^{(k)} \\ 0 & mujal \end{cases}, \tag{13}$$

kus $p' = \sum_{x \in V^{(k)}} P(x|x_{1:i-1})$.

1.5.4 Suurima p valik

Holtzman *et al.*, 2019 tulid välja meetodiga *nucleus sampling* ehk suurima p valik. Erinevalt suurima k valikust võetakse antud juhul minimaalne hulk tokeneid $V^{(p)} \subset$

V, mille korral on $V^{(p)}$ summa tõenäosus suurem kui mingi etteantud arv p nagu on näha valemist 14. Meetodist võib mõelda kui suurimast k valikust, aga k on dünaamiline.

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p. \tag{14}$$

Sarnaselt suurima k valikule skaleeritakse allesjäänud tokenite tõenäosused. Antud meetod välistab olukorra, mis võib esineda suurima k valiku korral, kus $k = 100$ puhul on peale skaleerimist esimese 10 tokeni tõenäosus 0,98 ja ülejäänud 90 tõenäosus 0,02. Piisavalt palju kordi genereerides satuvad teksti harva esinevad sõnad, mistõttu võib genereeritud tekst olla "ebaselge".

1.6 Mudelite valideerimine

Mudelite valideerimisel on võimalik kasutada automaatseid hinnanguid. Inimese hinnang on parim, mida saab genereeritud tekstide kvaliteedi hindamisel kasutada. See-eest on automaatsete hinnangutega kergem hinnata tekstide varieeruvust ning automaatsete hinnangute eelis seisneb selles, et need ei vaja inimtööjõudu (Hashimoto, Zhang ja Liang, 2019). Viimane ongi põhjus, miks töös vaadatakse ainult automaatseid hinnanguid, mille abil valitakse genereerimise algoritm ning hinnatakse tekstide originaalsust. Tekstide kvaliteedi hindamisel treenitakse genereeritud tekstidel klassifitseerimismudel ning vaadatakse kas see annab sarnase tulemuse päris tekstidega. Kasutatakse just klassifitseerimismudelit, kuna soov on genereeritud tekste kasutada terviseinformaatika kursustel näiteks klassifitseerimismudelite loomisel.

1.6.1 Perpleksus

Treenitud keelemudeli abil on võimalik arvutada iga teksi tõenäosus (Koehn, 2009). Antud ideed saab kasutada genereeritud tekstide hindamisel. Soov on, et genereе-

ritud tekstid on võimalikult inimteksti sarnased. Üks viis selle hindamiseks on kasutada perpleksust (Holtzman *et al.*, 2019). Perpleksus põhineb ristentroopial. Ristentroopia arvutatakse järgneva valemiga

$$H(p) = -\frac{1}{n}\log p(x_1, x_2, \ldots, x_n) = -\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|x_1, \ldots, x_{i-1}) \qquad (15)$$

Perpelksus on ristentroopia eksponent

$$PP = e^{H(p)}$$

(Koehn, 2009). Kuna mudeli treenimisel minimiseeritakse ristentroopiat, siis võib öelda ka, et mudeli treenimisel minimiseeritakse perpleksust. Ristentroopia arvutamist saab vaadata näite lausega "Patsient kurdab valu paremas õlas.". Sellisel juhul saab mudelit kasutades arvutada iga sõna tõenäosuse. Kuna tekstidele ei eelne ühtegi lauset, siis sellisel juhul lisatakse algusse "<|endoftext|>" token. Samuti kuna tegu on lõpuga ning tekstile ei järgne midagi, siis lisatakse lõppu "<|endoftext|>" token. Antud näite puhul eeldatakse, et iga sõna saab kodeerida täpselt ühe tokeniga ning näite huvides näidatakse sõnu mitte tokeneid. Fiktiivseid näite tõenäosusi on näha Tabelis 1 ning perpleksuse saab kui Tabelis 1 olevast keskmisest võtta eksponentfunktsioon. Võttes eksponentfunktsiooni saab $PP = e^{1,335} = 3,780$.

Tabel 1: Sõnade tinglikud tõenäosused

Ennustus	p	$-\log p$
p(patsient | < |endoftext| >)	0,052	2,957
p(kurdab | < |endoftext| >, patsient)	0,312	1,165
p(valu | < |endoftext| >, patsient, kurdab)	0,256	1,363
p(paremas | < |endoftext| >, patsient, kurdab, valu)	0,123	2,095
p(õlas | < |endoftext| >, patsient, kurdab, valu, paremas)	0,181	1,709
p(. | < |endoftext| >, patsient, kurdab, valu, paremas, õlas)	0,421	0,865
p(< |endoftext| > | < |endoftext| >, patsient, kurdab, valu, paremas, õlas, .)	0,824	0,194
	Keskmine	1,335

1.6.2 *Self*-BLEU

Bilingual evaluation understudy, lühendatult BLEU, on automaatne hinnang, mis loodi masintõlke hindamiseks (Papineni *et al.*, 2002). BLEU võrdleb automaatses tõlkes esinevaid n-gramme inimese poolt tõlgitud tekstidega. BLEU võib võtta väärtusi 0 ja 1 vahel. Kui tõlge on hea, siis BLEU väärtus on ühele lähemal, kuid kui tõlge on kehv, siis on väärtus nullile lähemal. BLEU avaldub valemiga

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \ln p_n\right).$$

Valemis w_n on positiivne kaal ning $\sum_{n=1}^{N} w_n = 1$. BP on *brevity penalty* ehk lühiduse karistus, mis avaldub valemiga

$$BP = \begin{cases} 1, & c > r \\ e^{1-\frac{r}{c}} \end{cases}.$$

Valemis c on võrreldava teksti ehk automaatselt tõlgitud teksti pikkus, r on *effective reference corpus length* ehk lühim tõlgitud tekst inimese poolt ning p_n on n-grammi täpsuse geomeetriline keskmine. N näitab maksimaalset n-grammi pikkust mida vaadatakse. Kui $N = 4$, siis vaadatakse uni-, bi-, tri- ja tetragramme. BLEU puhul kasutatakse modifitseeritud n-grammi täpsust, mis arvutatakse valemiga

$$p_n = \frac{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{n\text{-}gramm \in \mathcal{C}} Count_{clip}(n\text{-}gramm)}{\sum_{\mathcal{C}' \in \{Candidates\}} \sum_{n\text{-}gramm' \in \mathcal{C}'} Count(n\text{-}gramm')}, \tag{16}$$

kus $\{Candidates\}$ on genereeritud tekstid ja $Count(n\text{-}gramm)$ loeb kokku, mitu korda antud n-gramm inimese poolt tõlgitud tekstides esines. $Count_{clip}(n\text{-}gramm) = \min\left(Count, \max(n\text{-}gramm)\right)$, kus $max(n\text{-}gramm)$ on maksimaalne n-grammi esinemis sagedus üle kõigi inimese poolt tõlgitud tekstide. Kui masintõlkes esineb n-gramm kolm korda ja inimese poolt tõlgitud tekstides esineb see kokku ühe kor-

ra, siis täpsus ei ole mitte $\frac{3}{3}$, vaid $\frac{1}{3}$.

BLEU'st on saadud *self*-BLEU. Selle asemel, et võrrelda masintõlke teksti inimtekstidega, saab hoopis võrrelda genereeritud teksti teiste genereeritud tekstidega. Sellisel juhul näitab *self*-BLEU, kui palju genereeritud tekst sarnaneb teistele genereeritud tekstidele ehk saab hinnata genereeritud tekstide varieeruvust (Zhu *et al.*, 2018). *Self*-BLEU väärtus, mis on nulli lähedane näitab suurt varieeruvust ja ühe lähedane väärtus näitab väikest varieeruvust. *Self*-BLEU arvutamist on võimalik vaadata järgnevate lausetega: "patsient kurdab valu vasakus õlas ja vasakus puusas", "patsient kurdab valu vasakus õlas", kus esimest lauset võrreldakse teisega.

Võrreldes esimest lauset teisega ehk esimene lause on nagu masintõlke lause ning teine on inimese poold tõlgitud, saadakse unigrammi täpsus $\frac{5}{8}$, kuna teises lauses ei esine sõnu "ja" ning "puusas". Samuti, kuna teises lauses esineb sõna "vasakus" üks kord, kuid esimeses esineb see kaks korda. Bigrammide täpsus tuleb $\frac{4}{7}$, kuna teises lauses ei esine bigrammid "õlas ja", "ja vasakus" ning "vasakus puusas". Sarnaselt saadakse trigrammide täpsus $\frac{3}{6}$ ja tetragrammide korral $\frac{2}{5}$. Kuna lause on pikem kui võrreldav lause, siis lühiduse karistust ei rakendata ehk $BP = 1$ ning *self*-BLEU tulemus on $\exp\{\frac{1}{4}(\ln\frac{5}{8} + \ln\frac{4}{7} + \ln\frac{3}{6} + \ln\frac{2}{5})\} = 0{,}5170$.

Kui esimest lauset võrrelda lausega "saabunud patsient kurdab valu vasakus õlas ja vasakus puusas", siis on näha, et esimene lause sisaldub teises, mistõttu tuleb uni-, bi-, tri- ja tetragrammi täpsus 1. Sellisel juhul on esimene lause lühem kui lause millega seda võrreldi. Esimese lause pikkus on 8 ja teise lause pikkus 9. Seega $BP = e^{1-\frac{9}{8}} = 0{,}8825$. Kuna kõik täpsused olid ühed, siis tulevad logaritmi väärtused 0 ning eksponent väärtus 1. Seega *self*-BLEU tulemus on 0,8825.

Esimest lauset saab ka korraga võrrelda mõlema lausega. Sellisel juhul loetakse täpsused ülekõigi lausete, millega võrreldakse. Võrreldes esimest lauset mõlemaga, siis kuna endiselt esimene lause sisaldub kolmandas, siis kõik n-grammide täpsused tulevad ühed. See-eest kuna võrdluses esineb lause, mis on lühem, siis lühiduse karistust ei tule ning seega on *self*-BLEU tulemus 1.

1.6.3 ROUGE

Alapeatükk pärineb allikast (Lin, 2004). Viimane lõik, kus seletatakse ROUGE kasutamist tekstide orginaalsuse hindamisel ei pärine nimetatud allikast. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) on loodud tekstide kokkuvõttete kvaliteedi hindamise eesmärgil. ROUGE jaotub neljaks: ROUGE-N, ROUGE-L, ROUGE-W ja ROUGE-S. Täpsemalt vaadatakse neist ROUGE-L, mis baseerub kahe lause pikimal ühisjadal. ROUGE-L avaldub valemiga

$$R_{lcs} = \frac{LCS(s,r)}{m} \tag{17}$$

$$P_{lcs} = \frac{LCS(s,r)}{n} \tag{18}$$

$$ROUGE\text{-}L = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}}, \tag{19}$$

kus m on lause r pikkus, n on lause s pikkus ja $LCS(s,r)$ tähistab lausete s ning r pikimat ühisjada. Valemis 19 $\beta = P_{lcs}/R_{lcs}$. Valemist on näha, et kui $s = r$, siis $LCS(s,r) = m = n$ ning $ROUGE\text{-}L = 1$. Kui teostatakse mitut võrldemist, siis leitakse üle lausete r_i maksimaalne ROUGE-L nagu on näha järgnevast valemist

$$ROUGE\text{-}L_{multi} = \text{argmax}_i ROUGE\text{-}L(r_i, s).$$

Kuigi ROUGE on loodud teksti kokkuvõtete hindamiseks, siis antud töös kasutatakse seda tekstide orignaalsuse hindamisel, võrreldes genereeritud teksti treeningandmestikuga. Sellisel juhul saab ROUGE-L tulemuse põhjal otsusta, kas genereeritud tekstidele leidub treeningandmestikus vaste. Samuti, kuna ROUGE-L sõltub nii genereeritud teksti pikkusest, kui teksti millega seda võrreldakse, siis vaadatakse ka ainult valemis 18 esinenud suhet P_{lcs}, mis sõltub ainult genereeritud teksti pikkusest, kuna $LCS(s,r) \leq \min(n,m)$, siis $0 \leq P_{lcs} \leq 1$. Sellisel juhul on P_{lcs} väärtus 1 isegi kui genereeritud tekst sisaldub mõnes suuremas treeningandmestiku tekstis.

Seega vaadatakse samuti suurust

$$P_{lcs_{multi}} = \text{argmax}_i P_{lcs}(r_i, s),$$

ehk pikimat ühisjada suhet.

2 GPT-2 treenimine haigusloo tekstidel

GPT-2 mudelist on olemas erinevaid versioone. Kõige väiksem on GPT-2 *small*, millel on 117 miljonit parameetrit (Radford *et al.*, 2019). GPT-2 *small* on näidanud häid tulemusi varasemalt meditsiini tekstide genereerimisel ja hoolimata suurtest parameetrite arvust saab hakkama väiksema andmestikuga (Amin-Nejad, Ive ja Velupillai, 2020). Samuti tuleb arvestada, et suurendades parameetrite arvu, suureneb mudeli treenimiseks kuluv aeg, mistõttu kasutatakse GPT-2 *small* mudelit (edaspidi GPT-2). Kõik arvutused ja mudelite treenimised siin lõigus ja järgnevates tehti kasutades Tartu Ülikooli teadusarvutus keskuse ressursse (Tartu Ülikool, 2018).

2.1 Andmed ja tekstide töötlemine

Mudel treeniti kõigi geenivaramuga liitunud patsientide anonümiseeritud epikriisi tekstidel. Tekste on kokku 12,8 miljonit, mis andmemahult on ∼6,5GB. Võrreldes GPT-2 treenimisega, kus kasutati ∼40GB teksti (Radford *et al.*, 2019) on seda vähe, kuid on näidatud, et GPT-2 arhitektuuriga mudelit on võimalik meditsiini tekstidel treenida, kasutades 55,404 unikaalset teksti (Amin-Nejad, Ive ja Velupillai, 2020). Kogu andmestikust eraldati treening andmeteks 11 635 679 teksti, 612 405 eraldati valideerimisandmestikuks ja 644 636 seati kõrvale testimiseks. GPT-2 eeltreenimisel ei ole testandmestikku vaja, kuid eraldades praegu tekstid testimise eesmärgil, tehakse kindlaks, et kui tulevikus testita genereeritud andmetel treenitud mudelit, siis tekste genereeriv mudel ei ole varem test tekste näinud.

Enne tokeniseerija loomist ning mudeli treenimist tekstid puhastati. Tekstidest eemaldati tabelid, kõik kuupäevad asendati sõnaga “DATE” ja samuti asendati patsiendi ning arsti anonümiseeritud koodid sõnaga “NAME” või “ta”. Tabelite eemaldamisel ning patsiendi ja arsti koodide asendamisel kasutati sama meetodi, mida Meelis Perli kasutas oma magistritöös (Perli, 2021), mis põhineb Pythonis estNltk

teegi kasutusel (Laur *et al.*, 2020). Teegis on olemas meetod kuupäevade eemaldamiseks, kuid selle rakendamisel oli näha, et see asendas osad mõõtmistulemused sõnaga "DATE". Seetõttu sai loodud uus kuupäeva asendaja regulaaravaldiste abil, mis paarisaja teksti läbivaatlusel andis parema tulemuse.

Vaadates teksti liike Tabelis 2, siis nelja liiki tekste on andmestikus vähe ning kõigis on tekstid lühikesed ning praktiliselt varieeruvus puudub. Samuti on "allergia esinemine" ja "allergeen" tekstid ühe kuni nelja sõnalised. Seetõttu mudeli treenimisel neid tekste ei kasutata.

Tabel 2: Tekstide jaotumine liigiti

Teksti liik	**Kokku**
Anamnees	5 401 219
Kokkuvõte	4 028 432
Uuringud	1 396 156
Objektiivne leid	1 187 179
Protseduur	554 055
Operatisooni protokoll	112 425
Operatisoon	91 446
Allergia	86 190
Allergeen	32 569
Surm	1 964
Allergia esinemine	1 085
Kokku	12 892 720

2.2 Tokeniseerija treenimine

Üks võimalus enne mudeli treenimist on kasutada olemasolevat tokeniseerijat, kuid sarnast eesti keele epikriisi tekstide BPE tokeniseerijat ei ole veel loodud. Meelis Perli 2020 lõi oma magistri töös eesti keele epikriisi tekstidel tokeniseerija, kuid seal on tegu SentencePiece tokeniseeriga, mistõttu ei ole seda võimalik kasutada (Perli, 2021). Samuti on Tartu Ülikooli keeletehnoloogia õppetooli poolt loodud GPT-2 mudel koos BPE tokeniseerijaga, kuid seal ei ole kasutatud epikriise tekste (TartuNLP, 2023). Sellisel juhul on sõnastikus vähe erialaseid sõnu, mis üldjuhul

vähendab mudeli sooritusvõimet (Kalyan, Rajasekharan ja Sangeetha, 2021). Seega loodi uus BPE tokeniseerijaga epikriisi tekstidel.

Tokeniseerija loomisel kasutati treeningandmestikuks eraldatud tekste. Tokeniseerija sõnastiku suuruseks valiti 50 257, kuna sama kasutasid GPT-2 loojad ning sarnaselt GPT-2 loojatele lisati ainult üks eritoken "<|endoftext|>", millega tähistatakse lause algust, lause lõppu ja *padding*'ut (Radford *et al.*, 2019). Tokeniseerija treenimisel kasutati Pythonis *transformers* teeki (Wolf *et al.*, 2020).

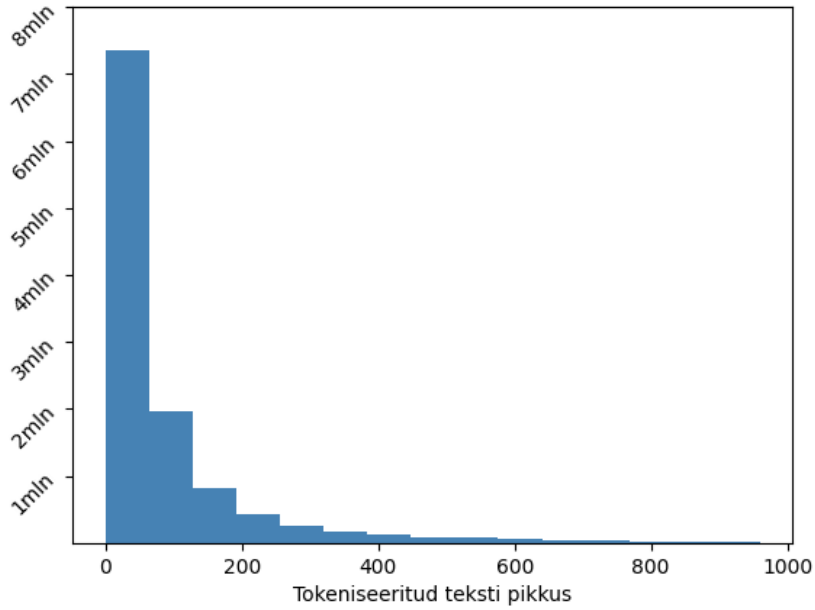
2.3 Mudeli eeltreenimine

Enne mudel treenimist tehakse eeltreenimine. Eeltreenides ei lisata lisainformatisooni teksti algusse, et tekstide genereerimist suunata. Mudel eeltreenitakse, et seda saaks hiljem mõne teise ülesande jaoks treenida nagu näiteks klassifitseerimine. GPT artikli autorid on eeltreenitud mudeli treeninud teksti kokkuvõtteid kirjutama ja küsimustele vastama (Radford *et al.*, 2018). Meditsiiniteksidel eeltreenitud mudeli võib samuti treenida kokkuvõtteid kirjutama või küsimustele vastama. Mõlemal juhul on selleks vaja vastavat andmestikku.

Mudeli treenimiseks tekstid tokeniseeriti. Enne tekstide tokeniseerimist lisati teksti algusse "<|endoftext|>" ja teksti lõppu "<|endoftext|>". Seejärel kõik tekstid tükeldati 256 pikkusega tokeniteks ehk tehti eeldus, et tegu on 256. järku Markovi ahelaga. Valiti väiksem pikkus kui GPT-2 puhul, et mudeli treenimiseks kuluv aeg oleks lühem. Seega kui tekst on 400 tokenit pikk, siis tekst jaotub kaheks. Üks osa on 256 tokenit ja teine 144 tokenit. Teisele osale lisatakse 112 tokenit juurde, et kogu pikkus oleks 256. Antud juhul lisatakse 112 "<|endoftext|>" tokenit. Samuti kui ristentroopiat arvutatakse mudeli treenimisel, siis *padding* tokeneid ei arvestata, mistõttu need ei mõjuta mudeli treenimist.

Tokeniseeritud treeningandmestiku pikkust on näha Joonisel 5. Joonisel on välja jäetud tokeniseeritud tekstid, mille pikkus on suurem kui 1024 tokenit ning mis moodustavad 5,46% treeningandmestikust. Nagu joonisel näha, siis enamus

tekste on lühemad kui 256 tokenit. Sellised tekste on 10 559 169 ning need moodustavad 91,65% treeningandmestikust. Treeningandmestikust tokeneid on kokku 1 075 485 061.



Joonis 5: Treeningandmestiku tokeniseeritud tekstide pikkuse jaotus

Pärast tekstide tokeniseerimist tekivad samuti lühikesed tekstid. Mõne tokeniseeritud teksti pikkus on näiteks 3 tokenit. See tuleneb sellest, et tokeniseeritud lause on näiteks 259 tokenit pikk või tegemist ongi väga lühikese tekstiga. Kui tokeniseeritud tekst on lühike, ehk väiksem kui 5 tokeni, siis selliseid lauseid treenimisel ei kasutata. Tokeniseeritud laused mis on peale tokeniseerimist lühemad kui 5 tokenit on kokku 1 625 282, millest 1 216 393 ei ole saadud tükeldamise moel. Kokku on nende lausete peale 3 951 930 tokenit, mis moodustab kogu tokenite hulgast 0,37%.

Mudeli treenimisel kasutati Pythonis *transformers* teeki (Wolf *et al.*, 2020). Mudeli treenides kasutati samu hüperparameetreid mis GPT-2 puhul (Radford *et al.*, 2018; Radford *et al.*, 2019), kuna mudeli treenimine on ajamahukas ja nende täpsem otsimine on liiga ressursi nõudlik. Ainuke parameeter mille väärtus erines oli

epohhite arv. Treenides mudelit käiakse ühe epohhi jooksul kogu treeningandmestik üle. Mudeli treenimisel kasutati ühte Nvidia a100 80gb graafikakaarti, millele mahub treenimisel plokk suurusega 64, kus plokkis üks element on tokeniseeritud tekst pikkusega 256. Mudelit treeniti 8 epohhi ning ühe epohhi aeg oli 26 tundi. Kokku võttis mudeli eeltreenimine 215 tundi.

2.4 Mudeli treenimine

Saamaks mudelit, millega on võimalik tekstide genereerimist suunata, siis mudeli treenimisel lisati tekstide ette lisainformatsiooni. Lisainformatsioon on teksti liik, vanuse grupp, patsiendi sugu ja põhidiagnoos. Antud informatsioon lisatakse teksi algusse nimetatud järjestuses ja eraldatakse komadega (Näide: anamnees, vanusegrupp 4, mees, B15-19, ...). Töös hiljem küll kasutatakse ainult diagnoosikoodigruppi, kuid ülejäänud lisainformatsioon lisati, kuna magistritöö väliselt on soov mudelit rakendada teistsuguste tekstide genereerimisel, kus saab eeltoodud omaduste järgi liigitada. Teksti liikide jaotust treening andmete seas on näha Tabelis 3, kust on näha, et andmestikus on kõige rohkem "anamnees" tüüpi tekste.

Tabel 3: Treening tekstide jaotumine liigiti

Teksti liik	**Kokku**
Anamnees	4 874 965
Kokkuvõte	3 634 935
Uuringud	1 260 120
Objektiivne leid	1 071 511
Protseduur	500 141
Operatisooni protokoll	101 492
Operatisoon	82 725
Kokku	11 525 889

Tekstid jaotati gruppidesse vanuse järgi. Grupid moodustati 5 aastaste vahedega. Alates 80. eluaastast võeti kõik patsiendid ühte vanuse gruppi ning kõige noorem grupp on 15-20 aastased. Viimasena lisati igale tekstile põhidiagnoosikood, mis

on RHK-10 formaadis (Maailma Terviseorganisatsioon, 1993). Diagnoosikoodide puhul teostati samuti grupeerimist. Näiteks koodid A15-A19 tähistavad erinevaid tuberkuloose ehk need võetakse üheks koodiks kokku mida tähistati A15-19. Teine näide on, et H80-H83 tähistavad sisekõrvaga seotud probleeme ning antud koodid grupeeriti tähisega H80-83. Andmestikus on 134 361 naist ja 70 866 meest ning tekstidest 9 474 319 kuulub naistele ja 3 418 401 meestele. Keskmiselt on ühe naise kohta 70,51 teksti ja ühe mehe kohta 25,44 teksti, mis näitab seda, et eesti mees käib keskmiselt vähem arsti juures kui naine.

Mudelit treeniti 8 epohhi. Mudeli täpsus valideerimisandmestikul on 25,82%. Mudeli täpsus 25,82% tähendab, et veerandil juhtudest valideerimisandmestikus suudab mudel järgmise sõna ära arvata. Kui tegu oleks juhuslikku ennustajaga, siis täpsus oleks 1/50257 ehk $\approx$ 0,002%. Valideerimisandmestikus saadi perpleksus 5,0647. Antud suurus on oluline, sest genereerimise algoritmi valikul on soov, et genereeritud tekstid annaks lähedase tulemus.

3 Genereerimise algoritmi valik

Sobiva genereerimise algoritmi väljavalimisel kasutati sarnast ideed nagu on kasutatud artiklis "*The curious case of neural text degeneration*" (Holtzman *et al.*, 2019). Idee seisneb selles, et leitakse inimteksti perpleksus ja Self-BLEU ning vaadatakse, milline genereerimise algoritm saavutab sarnased tulemused. Võimalus on kasutada antud artiklis leitud parimat meetodi, kuid seal ei ole hinnatud algoritme eesti epikriisi tekstide peal. Eeldatakse, et eesti keeles kirjutatud epikriisi tekstides võib näha vähem varieeruvust kui igapäevases inglise keeles.

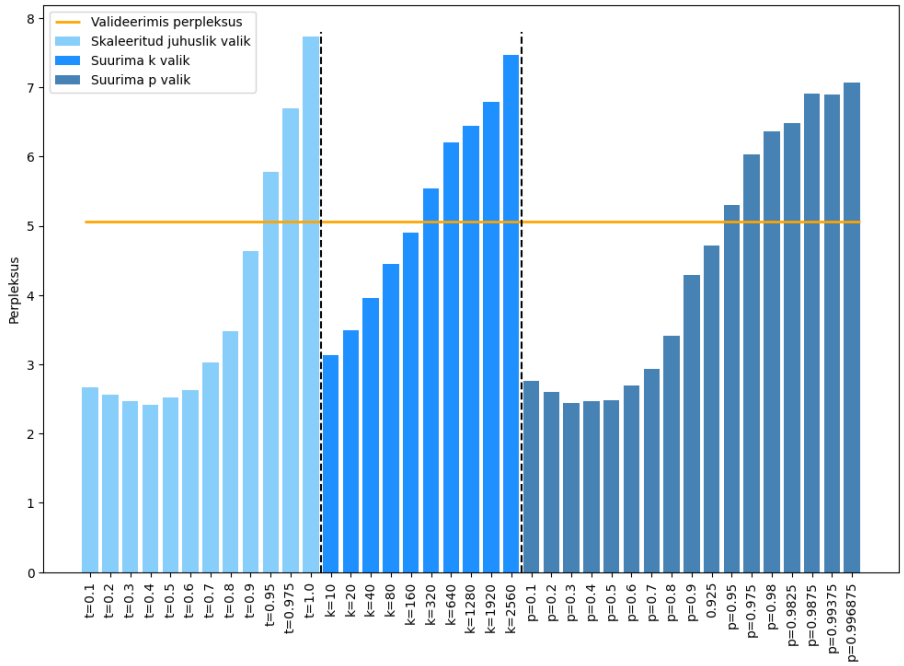
Tekstide genereerimisel vaadati kolme erinevat algoritmi: skaleeritud juhuslik valik, suurima k valik ja suurima p valik. Kõigi algoritmide puhul prooviti erinevaid parameetrite väärtusi. Iga algoritmi puhul genereeriti 5000 teksti sarnaselt artiklis tehtud meetodile (Holtzman *et al.*, 2019). Piirduti 5000 teksti genereerimisega, sest tekstide genereerimine on ajakulukas. Keskmiselt võttis 5000 teksti genereerimine 7 tundi.

3.1 Genereeritud tekstide perpleksus

Kõigepealt hinnati genereeritud tekstide perpleksused. Genereeritud tekstide perpleksusi on näha Joonisel 6, kus on näha erinevad parameetri väärtused mida algoritmide puhul testiti. Joonisel vasakul, kus parameetri väärtused on tähistatud t'ga, on tegu skaleeritud juhuslikku valikuga. Keskel on suurima k valik ja paremal on suurima p valik. Joonisel on näha, kuidas kõigi kolme algoritmi korral parameetri väärtust tõstes perpleksus suureneb. Antud algoritmide puhul kui $t = 1{,}0$, $k = 50257$ ja $p = 1{,}0$, siis erinevust algoritmides ei ole ning tegu on juhuslikku valikuga, kus mudelist saadud tõenäosusi ei ole muudetud. Parameetri $t = 1{,}0$ korral on näha, et genereeritud tekstide perpleksus on suurem kui valideerimisandmestiku perpleksus.

Vaadates erinevaid perpleksusi, siis on näha, et skaleeritud juhuslikku valiku puhul annab kõige lähedasema tulemuse skaleerimise väärtus $t = 0{,}9$ ja $t = 0{,}95$. Teiste

skaleerimise väärtuste korral on tulemused kaugemad. Suurima k valiku korral saab kõige lähedasema tulemuse $k = 160$ ja $k = 320$ korral. Suurima p valiku korral saab kõige lähedasema tulemuse $p = 0{,}925$ ja $p = 0{,}95$.

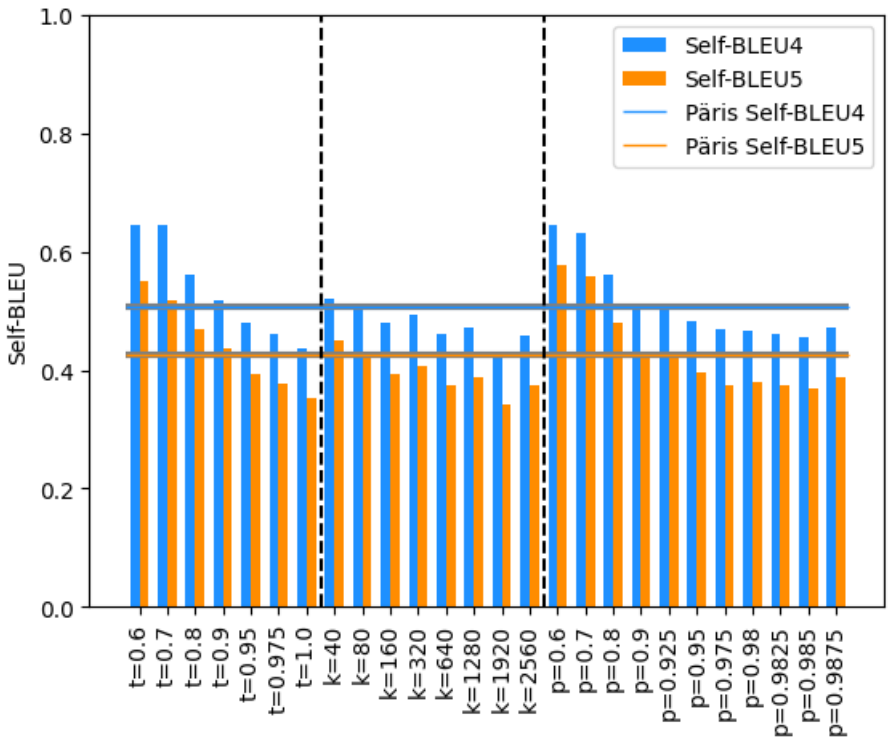


Joonis 6: Genereeritud tekstide perpleksuse võrdlemine päristekstide perpleksusega

3.2 Genereeritud tekstide *self*-BLEU

Sarnaselt artiklile (Holtzman *et al.*, 2019) vaadati *self*-BLEU puhul kui maksimaalne n-gramm on pikkusega 4 ja 5. Samuti *self*-BLEU arvutamisel valiti juhuslikult 5000 teksti seast 1000 (Holtzman *et al.*, 2019). Igat teksti tuhandest võrreldi ülejäänud 4999 tekstiga. *Self*-BLEU ei arvutada kõigi tektide puhul, kuna tegu on arvutuslikult mahuka ülesandega. Kui võrrelda 1000 teksti 4999 tekstiga, siis tehakse 4 999 000 võrdlust. Keskmiselt võttis ühe teksti võrdlemine 4999 tekstiga 90 sekundit. Samuti ei leitud *self*-BLEU nende algoritmide parameetrite korral,

mille perpleksuse vahe inimtekstiga oli suur. Saadud *self*-BLEU tulemusi on näha Joonisel 7. Joonisel on näha, et inimtekstiga annab kõige sarnasema tulemuse suurima p valiku algoritm parameetri väärtusega $p = 0{,}925$. Perpleksuse puhul andis suurima k valiku korral lähedase tulemuse $k = 160$, mille *self*-BLEU on madalam päris tekstide omast.



Joonis 7: Genereeritud tekstide *self*-BLEU võrdlemine päristekstide *self*-BLEU'ga

Arvestades seda, et artiklis "*The curious case of neural text degeneration*" leiti, et suurima p valik annab parima tulemuse ning antud juhul suurima p valik annab lähedased tulemused, siis edaspidi kasutati tekstide genereerimisel suurimat p valikut parameetriga $p = 0{,}925$. Võrreldes artikliga on näha, et epikriisi tekstide korral on *self*-BLEU väärtused suuremad, mis viitab väiksemale varieeruvusele tekstides.

4 Genereeritud tekstide hindamine

Genereeritud tekstide puhul hinnati, kas suunatud genereerimine töötas, kas genereeritud tekstidel on võimalik treenida klassifitseerimismudel eeldusel, et suunatud genereerimine töötab ja kas genereeritud tekstid on originaalsed. Tekstide kvaliteedi ja genereerimise suunamist hinnati diagnoosikoodigruppide abil. Diagnoosikoodigruppidest valiti välja kuus, mis olid järgnevad: I10-16, M50-54, J40-47, K20-31, L60-75 ja F40-48. I10-16 puhul on tegemist hüpertensiivsete haigustega ehk vererõhuga seotud haigused. M50-54 puhul on tegemist lülisammaste probleemidega. J40-47 puhul on tegemist krooniliste alumiste hingamisteede haigustega. K20-31 puhul on tegemist söögitoru, mao ja kaksteistsõrmiksoole haigustega. L60-75 on tegemist nahahaigustega ning küünte ja juustega seotud probleemidega. F40-48 puhul on tegemist ärevuse ja stressiga seotud probleemidest. Eeltoodud 6 gruppi valiti välja, kuna tegu on üksteisest selgelt eristuvate haigustega ehk võib oodata nende gruppide puhul üksteisest selgemini eristuvaid tekste.

Treenimiseks valiti välja 200 000 teksti ja valideerimiseks 50 000 teksti. Treening- ja valideerimisandmestik valiti samast hulgast, mida kasutati GPT-2 treenimisel ja valideerimisel. Inimtekste on antud kuue kategooria kohta küll rohkem olemas, kuid 250 000 teksti genereerimine on ajakulukas ning mudelite võrdluse eesmärgil vähendati päristekstide hulka. 200 000 treeningteksti ja 50 000 valideerimisteksti genereerimisel kasutati suurimat p valiku parameetriga $p = 0{,}925$. Tekstide genereerimisel kasutati sama informatsiooni, mis oli pärisandmetes, ehk tekstiliik, vanus, sugu ja diagnoosikoodigrupp. Sellega saavutati olukord, kus genereeritud tekstide diagnoosikoodigruppide osakaalud oli samad mis päris tekstides. Diagnoosikoodigruppide osakaale treening- ja valideerimisandmestikus on näha Tabelis 4.

Tabel 4: Klassifitseerimistekstide jaotumine liigiti

Diagnoosikoodigrupp	**Treening**	**Osakaal**	**Valideerimine**	**Osakaal**
I10-16	63 606	31,80%	16 255	32,51%
M50-54	51 660	25,83%	12 825	25,65%
J40-47	29 297	14,65%	6 932	13,86%
K20-31	22 961	11,48%	5 747	11,49%
L60-75	20 108	10,05%	5 089	10,18%
F40-48	12 368	6,18%	3 152	6,30%

4.1 Tokenite jaotuse hindamine

Tekstide sarnasuse kontrollimisel on võimalik võrrelda tokenite jaotust. Võttes juhuslikult tokeni X, tingimusel, et see on diagnossigrupist Y, siis mis on selle saamise tõenäosus. Jaotuse võimalikud väärtused on sõnastikus olevad tokenid mida on 50 257. Samuti on võimalik võrrelda tokenite paaride jaotust. Võimalikke paare on küll 50257^2, kuid siin tuleb meeles pidada, et kõik võimalikud paarid ei realiseeru. Kahe hinnatud jaotuse võrdlemisel kasutati $L1$-kaugust (Lember, 2022).

Tabel 5: Tokenite jaotuste hinnangute $L1$-kaugus

		Genereeritud tekst					
		F40-48	I10-16	M50-54	J40-47	K20-31	L60-75
Päris tekst	F40-48	**0,2565**	0,7330	0,7137	0,7651	0,7058	0,9085
	I10-16	0,6783	**0,1859**	0,7574	0,735	0,5717	0,9358
	M50-54	0,6618	0,7557	**0,2005**	0,7758	0,6922	0,9065
	J40-47	0,7149	0,7186	0,7557	**0,2448**	0,6917	0,9089
	K20-31	0,6642	0,6202	0,6974	0,7262	**0,2365**	0,8990
	L60-75	0,9207	0,9760	0,9348	0,9298	0,9162	**0,2322**

Tabelis 5 on näha tokenite jaotuste võrdlus. Peadiagonaalil on selgelt väiksemad väärtused kui mujal. See näitab, et $L1$-kaugus päris tekstidest saadud jaotuste ja genereeritud tekstidest saadud jaotuste vahel on väiksem juhul kui mõlemad on samast diagnoosigrupist. Samuti Tabelis 6 on näha, et paaride jaotuste võrdluse

korral on peadiagonaalil selgelt väiksemad väärtused. Saadud tulemus kinnitab, et teksti genereerimise suunamine töötab.

Tabel 6: Tokeni paaride jaotuste hinnangute $L1$-kaugus

		Genereeritud tekst					
		F40-48	I10-16	M50-54	J40-47	K20-31	L60-75
Päris tekst	F40-48	**0,8267**	1,2047	1,2816	1,3853	1,2480	1,5339
	I10-16	1,1504	**0,5160**	1,2844	1,329	1,0558	1,5627
	M50-54	1,2369	1,2686	**0,5765**	1,3836	1,2395	1,5165
	J40-47	1,3110	1,2844	1,3543	**0,6678**	1,2684	1,5470
	K20-31	1,2370	1,1361	1,2843	1,3424	**0,6570**	1,5370
	L60-75	1,5514	1,5847	1,5438	1,5721	1,5467	**0,6684**

4.2 Klassifitseerimine

Vaatamaks, kas genereeritud tekstid on kasutatavad, siis päristekstidel ja genereeritud tekstidel treeniti klassifitseerimismudel. Klassifitseerimine teksti genereerides põhineb sarnasal ideel, mis on küsimusele vastamine (Radford *et al.*, 2019). Klassifitseerides on kogu epikriisi tekst küsimus, peale mida tuleb diagnoosikood ehk vastus. Küsimuse ja vastuse eraldamiseks kasutatakse vahetokeni. Vahetoken on kas üks unikaalne token või tokenite kombinatsioon, mis annab mudelile mõista, kus lõppeb küsimus ja algab vastus. GPT loojad kasutasid vahetokenina "$" märki (Radford *et al.*, 2018). Kuna epikriisi tekstides baastokeni "$" ei esinenud, siis sobis ka klassifitseerija treenimisel seda kasutada. Kuna mudel näeb treenides alati peale vahe tokenit kuute erinevat kombinatsiooni, siis võttes alati kõige suurema tõenäosusega tokenit ehk kasutates *greedy search*'i, saab tulemuse, mis vastab ühele kuuest klassist.

Keelemudeliga klassifitseerides tehti samuti piirang, et sisendtekst ei ole pikem kui 246. Selline piirang tehti, kuna maksimaalne konteksti pikkus mudelit eeltreenides

oli 256. Sellisel juhul 10 tokenit jääb selleks, et lisada lause algus token, vahetoken, vastus ehk diagnoosikoodigrupp ja lauselõpu token. Kuna kõik epikriisi tekstid ei ole väiksemad kui 246, siis tuleb osadele tekstidele teostada kärpimine. Tekste kärbiti sellisel juhul lõpust, kuna eeldati, et teksti alguses esinev informatsioon on olulisem kui teksti lõpus.

Tabelis 4 tuleb välja, et naiivne klassifitseerija, mis ennustab koguaeg treeningandmestikus esinevat kõige suuremat klassi saavutab täpsuse 32,51%. Seega saadud klassifitseerimismudel peab olema valideermisandmestikul täpsem kui 32,51%.

Päris epikriisi tekstide ja genereeritud tekstide korral võeti eeltreenitud GPT-2 ning treeniti mudel uutel andmetel 8 epohhi. Saadud mudelite täpsusi on näha Tabelis 7. Nagu tabelis näha, siis päris andmetel ja genereeritud andmetel treenitud mudel saavutab parema tulemuse kui naiivne klassifitseerija. Mõlemal juhul on näha, et valideerimisandmestikul saadud täpsus on natuke madalam.

Tabel 7: Mudelite täpsus

Andmestik	**Täpsus treeningandmetel**	**Täpsus valideerimisandmetel**
Päris tekstid	85,8475%	82,0360%
Genereeritud tekstid	87,9165%	85,3180%

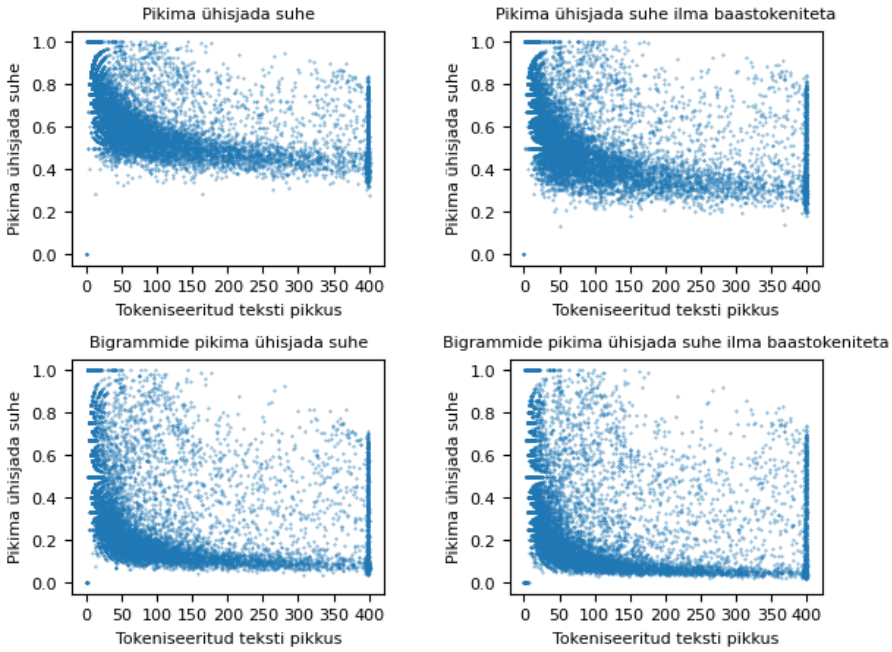
Järgmisena vaadati, kas genereeritud tekstidel treenitud mudeliga on võimalik päris tekste klassifitseerida. Valideerimisandmestikul, mis sisaldis päris tekste, saavutas genereeritud tekstidel treenitud mudel täpsuse 80,0860%. Hinnates päris tekstidel treenitud mudeli täpsust genereeritud valideerimisandmestikul, saadi täpsus 84,5560%. Kuna genereeritud teksidel treenitud mudel sai päris tekstidel täpsuse 80,0860%, siis võib öelda, et genereeritud tekstid on kvaliteetsed. Samuti annab see võimaluse treeningandmestikku suurendada, genereerides sinna tekste juurde.

4.3 Genereeritud tekstide originaalsuse hindamine

Teksti originaalsuse hindamisel võrreldi genereeritud tekste treeningandmestikuga. Kuna GPT-2 treeningandmestiku suurus oli 11,5 miljonit teksti, siis iga genereeritud teksti võrdlemine kõikide treeningandmestikku kuuluvate tekstitega ei olnud teostatav ajalise kulu tõttu. See-eest saab võrreldavate tekstide arvu vähendada vaadates ainult tekste, mis esinevad samas diagnoosikoodigrupis. Selline piirang tehakse, kuna oli näha, et tokenite jaotus on sellisel juhul sarnane ning klassifitseerimismudeliga saadi sarnane täpsus nii päris kui genereeritud tekstidel. Pikimat ühisjada hinnati ainult diagnoosikoodigruppi "F40-48" korral, kuna neid oli võrreldes ülejäänud 5 gruppiga genereeritud tekstide seas ja GPT-2 treeningandmestikus kõige vähem. Kokku oli diagnoosikoodigrupiga "F40-48" tekste GPT-2 treeningandmestikus 84 275.

Genereeritud tekstide originaalsust hinnati ROUGE-L ja P_{lcs} abil. Pikim ühisjada arvutati tokeniseeritud tekstidel neljal järgneval viisil: unigrammide pikim ühisjada koos ja ilma baastokeniteta ning bigrammide pikim ühisjada koos ja ilma baastokeniteta. Bigrammidega ja baastokeniteta ühisjadad kaasati, sest teatud juhtudel esines suur kattuvus genereeritud ja päristekstide kirjavahemärkide ja üksikute sõnade osas.

Saadud P_{lcs} tulemusi on näha Joonisel 8, kust tuleb välja, et pikima ühisjada suhe on seotud genereeritud lause pikkusega. Üldjuhul on näha, et pikemad laused on rohkem unikaalsed kui lühemad, mis tundub loogiline, sest juhusliku valikuga tokeneid genereerides on suurem tõenäosus päris lausest kõrvale kalduda. See-eest on näha uni- ja bigrammide korral, et on pikemaid genereeritud tekste, millele leidub treeningandmestikus mingi vaste ehk pikima ühisjada suhe on suurem kui 0,6. Lisaks on näha, et suur erinevus on selles, kuidas pikimat ühisjada on arvutatud. Hajuvusdiagrammilt on näha, et kui baastokeneid ei arvestata, siis näeb madalamaid pikima ühisjada suhte väärtuseid. Suurem erinevus tuleneb sellest, kas pikimat ühisjada arvutada uni- või bigrammide abil, kus on näha, et saadud

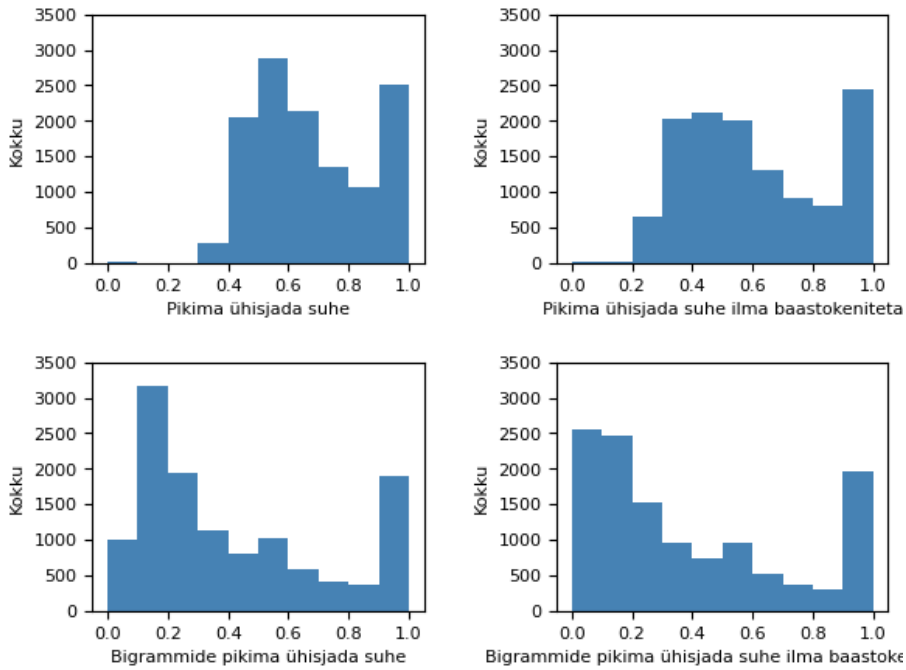


Joonis 8: Genereeritud tekstide pikima ühisjada suhe. Joonistel, kus on baastokenid välja jäetud, näidatakse teksti pikkust koos baastokenitega

väärtused on bigrammide korral selgelt väiksemad.

Joonisel 9 on näha pikima ühijada suhete jaotus. Histogrammilt tuleb paremini välja tekstide hulk, millele leidub treeningandmestikus vaste, kuna hajuvusdiagrammil väärtusega 1 olevad tekstid olid ühel joonel. Samuti bigrammide korral on näha, et pikima ühisjada suhe on märgatavalt väiksem kui unigrammide korral.

Seda, et unigrammide korral andis pikima ühisjada treeningandmestikus esinevad pikad tekstid on näha Joonisel 10, kus on treeningandmestikust suurima ühisjada andnud teksti pikkuse erinevus genereeritud tekstiga. Vaadates Joonist 10 tuleb arvestada, et parempoolsetel joonistel on vahed arvutatud peale baastokenite eemaldamist. Samuti on näha, et vahe treeningandmestiku tekstiga on paljudel juhtudel üle 20 000 tokeni ning unigrammide korral, kus baastokeneid arvestati, saadi 5216 korral pikim ühisjada ühe ja sama tekstiga treeningandmestikust, mille pikkus oli 27 074 tokenit. Näha on, et üks pikk lause, millel enamjaolt ei ole palju

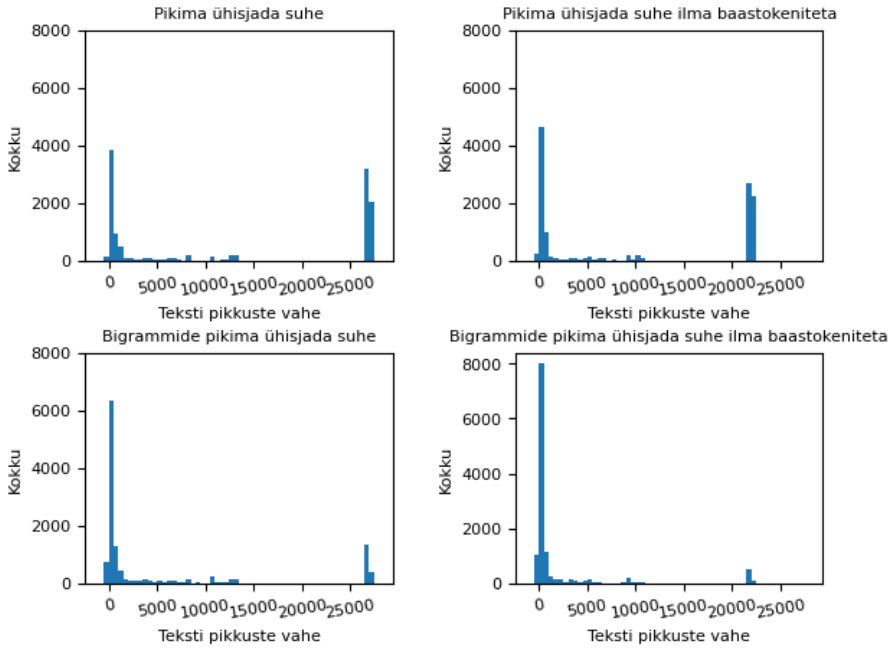


Joonis 9: Genereeritud tekstide pikima ühisjada suhte jaotus

ühist genereeritud lausega saab puhtalt oma pikkuse tõttu pika ühisjada. Kuigi bigrammide korral esineb tekste, millega on vahe üle 20 000 tokeni, siis on neid märgatavalt vähem. Bigrammide korral, kus baastokenid ei ole arvestatud, annavad pikima ühisjada enamjaolt sama või natukene pikemad tekstid. Tingimus, et kaks järjestikust tokeni peavad olema samad, selgelt aitab olukorra vastu mis esines unigrammi korral.

Vaadates Joonisel 11 suurust ROUGE-L, mis penaliseerib kui treeningandmestikus olev tekst on pikem kui genereeritud tekst, on näha, et nii uni- kui bigrammide korral on hajuvusdiagrammid sarnasemad, erinevalt pikima ühisjada suhtest. Samuti hajuvusdiagrammide vahel on väiksem erinevus, kas vaadata pikimat ühisjada koos baastokenitega või ilma.

Vaadates ROUGE-L histogramme Joonisel 12 on samuti näha erinevust. Suurem erinevus võrreldes pikima ühisjada suhtega on selles, et unigrammide korral on

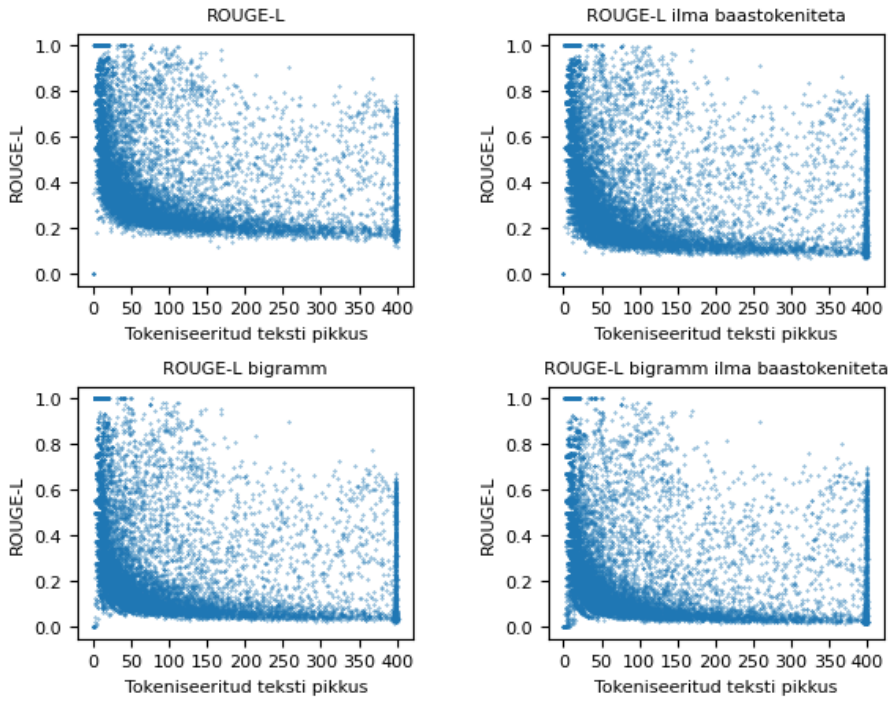


Joonis 10: Suurima ühisjada andnud teksti pikkuse vahe genereeritud tekstiga.

näha, et saadud tulemused on oluliselt madalamad ja histogrammid oma kujult sarnanevad sellele kui leiti bigrammide pikim ühisjada koos baastokenitega.

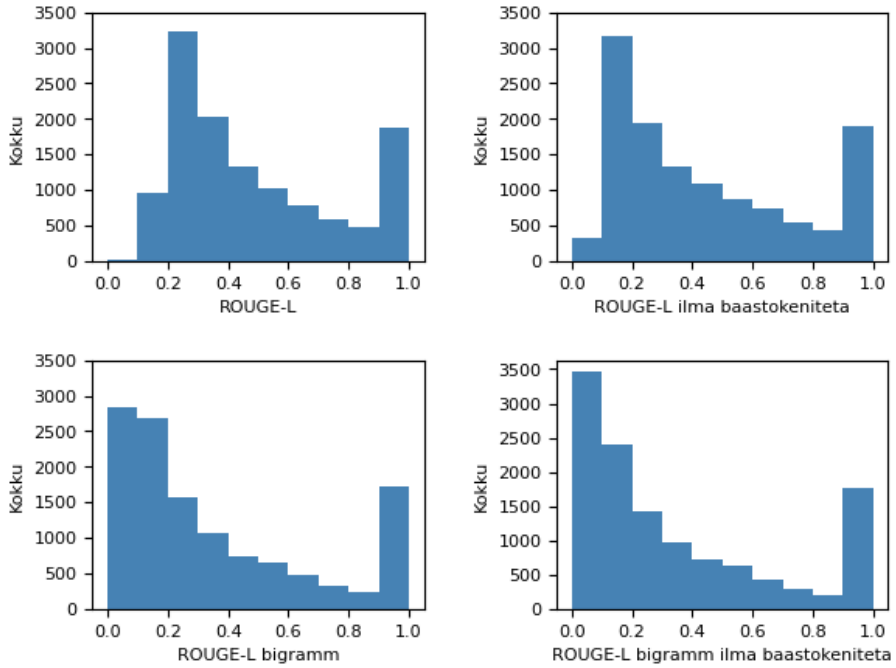
Seda, et ROUGE-L penaliseerib treeningteksti, kui see on oluliselt pikem kui genereeritud tekst on näha Joonisel 13. Joonisel on näha, et saadud tulemus erineb palju võrreldes pikima ühisjada suhtega. ROUGE-L korral on näha, et tekstid, mis andsid suurima ROUGE-L väärtuse on üldjuhul sama pikad kui genereeritud tekstid. Samuti on näha ROUGE-L korral, et saadud tulemus on kõigi pikima ühisjada arvutuse meetodite korral sarna.

Kokkuvõttes nii P_{lcs} kui ROUGE-L põhjal oli näha, et mudeliga on võimalik genereerida originaalseid tekste. Kuna täpset piiri ei ole paigas, mis hetkest lugeda tekst originaalseks, siis on muidugi rakse otsustada kui suur osa tekstidest on originaalsed. Samuti võib mõelda, et originaalsus sõltub genereeritud teksti pikkusest. Kui 10 tokenit pikk lause annab pikima ühisjada suhte tulemuseks 0,7, siis see ei



Joonis 11: Genereeritud tekstide ROUGE-L. Joonistel, kus on baastokenid välja jäetud, näidatakse teksti pikkust koos baastokenitega

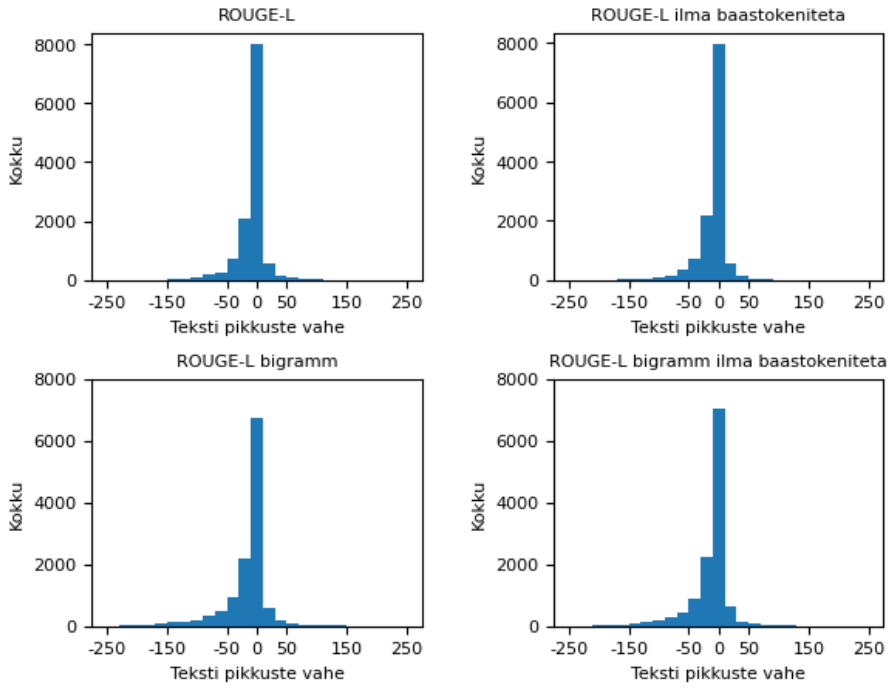
ole päris sama kui 300 tokenit pikk lause annab sama tulemuse. Samuti oli näha, et kui pikima ühisjada suhte arvutamisel kasutada bigramme, siis saadi ROUGE-L'ga sarnasemad tulemused. Eelis pikima ühisjada suhte kasutamisel ROUGE-L asemel on selles, et saadud tulemust on lihtsam interpreteerida. Kui $P_{lcs} = 0{,}6$, siis saab selgelt öelda, et treeningandmestikus leidub tekst, mis sisaldab 60% genereeritud teksti tokenitest. See-eest ROUGE-L korral saadav tulemus ei ole nii lihtsalt interpreteeritav. Samuti oli näha, et P_{lcs} korral on probleem kui treeningandmestikus esineb pikk tekst, millel on genereeritud tekstiga vähe seost, kuid oma pikkuse tõttu saavutab pika ühisjada. Muidugi kui genereeritud tekst on alamlause pikas päris tekstis, siis see ei ole originaalne. Üks võimalus sellist probleemi vältida on vaadata kaalutud pikimat ühisjada. Kaalutud pikimas ühisjadas saavad tokenid mis esinevad järjest või on lähestikku suurema skoori võrreldes tokenitega mille kaugus on



Joonis 12: Genereeritud tekstide ROUGE-L jaotus. Joonised, kus on baastokenid välja jäetud näidatakse teksti pikkust, koos baastokenitega

üksteisest 100 tokenit. Samuti on veel teine võimalus vaadata pikemaid n-gramme ühisjada arvutamisel nagu näiteks tri- ja tetragramme.

Pikima ühisjada arvutamise kiiruse osas on oluline, kas kaasata baastokenid või mitte. Kõik pikimad ühisjadad arvutati paraleelselt 42 tuuma peal, ehk iga tuum arvutas pikimad ühisjadad 300 genereeritud teksti jaoks, väljaarvatud üks tuum, mis arvutas 68 genereeritud teksti jaoks. Pikima ühisjada arvutamisel, kus ei võetud arvesse baastokeneid, kulus keskmiselt 300 teksti peale 28h. See-eest pikima ühisjada arvutamisel koos baastokenitega kulus keskmiselt 55h. Tegemist on pea kahe kordse ajalise kulu vahega. Seega tasub tulevikus kindlasti arvutada pikimat ühisjada ilma baastokeniteta, sest üldjuhul tekstis tokeniseeritakse baastokenitega üksikud numbrid ja kirjavahemärgid. Samuti on oluline mõelda, et praegu võrreldi igat genereeritud teksti 84 275 tekstiga. See-eest kui tahta orignaalsust hinnata



Joonis 13: Suurima ROUGE-L andnud teksti pikkuse vahe genereeritud tekstiga.

mõnel muul diagnoosigrupi tekstil, mille tekste on 4 korda rohkem, siis võib eeldada, et ka ajaline kulu kasvab 4 korda. Samuti kui paika oleks pandud piir, mille korral on tekst originaalne, siis on võimalik pikima ühisjadade arvutamisel jätta välja teatud määral genereeritud lausest lühemad tekstid, et kiirendada arvutusi. Näiteks, kui originaalsuse piir on 0,6, siis saab võrdlemisel kõrvale jätta 0,6 korda lühemad tekstid, kuna sellisel juhul ei saa suurima ühisjada suhe olla suurem kui 0,6.

4.4 Probleemid

Treeningandmestikus esines harva anonümiseerimata arsti nimesid anonümiseerimata, mistõttu võib genereeritud tekst neid sisaldada (genereeritud tekstide korral töö autor seda kordagi ei näinud). Seetõttu tuleb genereeritud tekstid enne nen-

de jagamist üle vaadata, et vältida tundlike andmete leket. Teine võimalus on anonüümiseerida genereeritud tekste, lootes eemaldada potentsiaalsed alles jäänud isikuandmed, kuid sellisel juhul ei saa nende esinemist täielikult välistada.

Kokkuvõte

Uurimistöö raames näidati, et keelemudeliga on võimalik genereerida originaalseid epikriisi tekste, mida on võimalik kasutada erinevatel terviseinformaatika kursustel või võrdlusandmebaaside loomisel. Tekstide genereerimiseks eel- ja põhitreeniti GPT-2 arhitektuuriga mudel kõigil Tartu Ülikooli geenivaramuga liitunud patisentide epikriisi tekstidel. Mudel eeltreeniti, et oleks võimalus ka tulevikus mudelit muudes ülesannetes kasutada nagu näiteks kokkuvõtete kirjutamine ning küsimsutele vastamine. Põhitreenimisel lisati tekstidele lisainformatsioon, mis oli patisentide kohta teada, et teksti genereerimist suunata.

Esimesena leiti treenitud mudeli jaoks parim tekstide genereerimise algoritm. Prooviti kolme järgnevat algoritmi: skaleeritud juhuslik valik, suurima k valik ja suurima p valik. Parima algoritmi valikul genereeriti iga parameetri väärtuse kohta 5000 teksti ning võrreldi nende perpleksust ja *self*-BLEU'd valideerimisandmestikul arvutatud väärtustega. Parim neist oli suurima p valik, kus $p = 0{,}925$.

Järgnevalt uuriti genereeritud tekstide omadusi. Selleks valiti välja kuus diagnoosikoodigruppi, millest genereeriti treeningandmestik suurusega 200 000 ja valideerimisandmestik suurusega 50 000 teksti. Genereeritud tekstidega vastati järgnevatele küsimustele: kas mudeliga on võimalik teksti genereerimist suunata, kas genereeritud tekstide abil on võimalik treenida päristekstide klassifitseerija ja kas genereeritud tekstid on originaalsed.

Esimesena uuriti, kas treenimisel teksti algusse informatsiooni lisamine aitab hiljem teksti genereerimist suunata. Selleks võrreldi päristekstide ja genereeritud tekstide tokenite jaotust, kasutades $L1$-kaugust. Genereeritud tekstid, mis olid päristekstidega samast diagnoosikoodigrupist andisid väiksema $L1$-kauguse. Tulemused näitasid, et teksti genereerimist on võimalik sellise meetodiga suunata.

Teiseks, kas genereeritud tekstide abil on võimalik treenida päristekstide klassifitseerija. Tulemused näitasid, et genereeritud tekstidel treenitud klassifitseerija

oli võrreldav päristekstidel treenitud klassifitseerijaga, mis näitab, et genereeritud tekste on võimalik näiteks terviseinformaatika kursustel kasutada. Saadud tulemust tasub kindlasti edasi uurida juba olemasolevatel mudelitel, mis kasutavad epikriisi tekste ning vaadata kas nende tulemusi on võimalik parandada genereeritud treeningandmeid kaasates.

Viimasena uuriti, kas genereeritud tekstid on originaalsed. Originaalsuse hindamisel vaadati ROUGE-L ja P_{lcs} väärtusi, kus genereeritud tekste võrreldi GPT-2 treeningandmestiku tekstidega. Oli näha, et esines madala ROUGE-L ja P_{lcs} väärtusega ehk väikese kattuvusega tekste. Saadud tulemus näitab, et mudeliga on võimalik genereerida originaalsed tekste, kuid arvesse tuleb võtta originaalse teksti definitsiooni puudumist. Töös leitud väärtuste põhjal võib paika seada soovitusliku piiri, millest suuremate väärtuste korral ei tohiks tekste jagada, kuid täpne piir ei ole töö autori otsustada. Samuti esines genereeritud tekstide seas mitte originaalseid tekste, mille kattuvus treenigandmestikus esinevate tekstidega oli soovitust suurem. Sellest lähtuvalt ei ole mudelit võimalik jagada ilma juriidilise nõusolekuta.

Kasutatud allikad

Amin-Nejad, Ali, Julia Ive ja Sumithra Velupillai (2020). "Exploring transformer text generation for medical dataset augmentation". Teoses: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, lk. 4699–4708.

Ba, Jimmy Lei, Jamie Ryan Kiros ja Geoffrey E Hinton (2016). "Layer normalization". *arXiv preprint arXiv:1607.06450*.

Bahdanau, Dzmitry, Kyunghyun Cho ja Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". *arXiv preprint arXiv:1409.0473*.

Bengio, Yoshua, Réjean Ducharme ja Pascal Vincent (2000). "A neural probabilistic language model". *Advances in neural information processing systems* 13.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee ja Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". *arXiv preprint arXiv:1810.04805*.

Dürr, Oliver, Beate Sick ja Elvis Murina (2020). *Probabilistic deep learning: With python, keras and tensorflow probability*. Manning Publications.

Gage, Philip (1994). "A new algorithm for data compression". *C Users Journal* 12.2, lk. 23–38.

Goodfellow, Ian, Yoshua Bengio ja Aaron Courville (2016). *Deep learning*. MIT press, lk. 73, 200–208.

Hashimoto, Tatsunori B, Hugh Zhang ja Percy Liang (2019). "Unifying human and statistical evaluation for natural language generation". *arXiv preprint arXiv:1904.02792*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren ja Jian Sun (2016). "Deep residual learning for image recognition". Teoses: *Proceedings of the IEEE conference on computer vision and pattern recognition*, lk. 770–778.

Hendrycks, Dan ja Kevin Gimpel (2016). "Bridging nonlinearities and stochastic regularizers with gaussian error linear units". *CoRR, abs/1606.08415* 3.

Holtzman, Ari, Jan Buys, Li Du, Maxwell Forbes ja Yejin Choi (2019). "The curious case of neural text degeneration". *arXiv preprint arXiv:1904.09751*.

Hugging Face (2023a). *Building a tokenizer, block by block*. URL: https://huggingface.co/course/chapter6/8?fw=pt (vaadatud 16.03.2023).

– (2023b). *Byte-Pair Encoding tokenization*. URL: https://huggingface.co/learn/nlp-course/chapter6/5?fw=pt (vaadatud 19.03.2023).

– (2023c). *Padding and truncation*. URL: https://huggingface.co/docs/transformers/pad_truncation (vaadatud 19.03.2023).

Kalyan, Katikapalli Subramanyam, Ajit Rajasekharan ja Sivanesan Sangeetha (2021). "Ammus: A survey of transformer-based pretrained models in natural language processing". *arXiv preprint arXiv:2108.05542*.

Keskar, Nitish Shirish, Bryan McCann, Lav R Varshney, Caiming Xiong ja Richard Socher (2019). "Ctrl: A conditional transformer language model for controllable generation". *arXiv preprint arXiv:1909.05858*.

Koehn, Philipp (2009). *Statistical machine translation*. Cambridge University Press.

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

Lember, Jüri (2022). *Informatsiooniteooria*. URL: https://courses.ms.ut.ee/MTMS.02.040/2022_spring/uploads/Main/info22.pdf (vaadatud 19.04.2023).

Lin, Chin-Yew (2004). "Rouge: A package for automatic evaluation of summaries". Teoses: *Text summarization branches out*, lk. 74–81.

Liu, Peter J, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser ja Noam Shazeer (2018). "Generating wikipedia by summarizing long sequences". *arXiv preprint arXiv:1801.10198*.

Maailma Terviseorganisatsioon (1993). *The ICD-10 classification of mental and behavioural disorders: diagnostic criteria for research*. World Health Organization.

Mielke, Sabrina J, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot *et al.* (2021). "Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP". *arXiv preprint arXiv:2112.10508*.

Papineni, Kishore, Salim Roukos, Todd Ward ja Wei-Jing Zhu (2002). "Bleu: a method for automatic evaluation of machine translation". Teoses: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, lk. 311–318.

Perli, Meelis (2021). "Representation Learning on Free Text Medical Data". Magistritöö. Tartu, Eesti: Tartu Ülikool.

Platen, Patrick von (2020). *How to generate text: using different decoding methods for language generation with Transformers*. URL: https://huggingface.co/blog/how-to-generate (vaadatud 19.03.2023).

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever *et al.* (2018). "Improving language understanding by generative pre-training".

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever *et al.* (2019). "Language models are unsupervised multitask learners". *OpenAI blog* 1.8, lk. 9.

Sennrich, Rico, Barry Haddow ja Alexandra Birch (2015). "Neural machine translation of rare words with subword units". *arXiv preprint arXiv:1508.07909*.

Sotsiaalministeerium (2016). *Tervishoiuteenuse osutamise dokumenteerimise ning nende dokumentide säilitamise tingimused ja kord*. URL: https://www.riigiteataja.ee/akt/126042016002 (vaadatud 25.03.2023).

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever ja Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". *The journal of machine learning research* 15.1, lk. 1929–1958.

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

TartuNLP (2023). *gpt-4-est-base*. URL: https://huggingface.co/tartuNLP/gpt-4-est-base (vaadatud 19.03.2023).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser ja Illia Polosukhin (2017). "Attention is all you need". *Advances in neural information processing systems* 30.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest ja Alexander M. Rush (oktoober 2020). "Transformers: State-of-the-Art Natural Language Processing". Teoses: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Lin-

guistics, lk. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Zhu, Yaoming, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang ja Yong Yu (2018). "Texygen: A benchmarking platform for text generation models". Teoses: *The 41st international ACM SIGIR conference on research & development in information retrieval*, lk. 1097–1100.

Lisa 1. BPE tokeniseerijaga sõnastiku loomise algoritm

Algoritm 1 BPE tokeniseerijaga sõnastiku loomine.

Sisend: treeningtekstid C, loodava sõnastiku suurus d_{vocab}

// Initsialiseeria esialgne sõnastik
// Tehakse eeldus, et eritokeneid on ainult üks. Muidu võtab algoritm
// sisendina eritokenid ja lisab need tokeniseerija algusse.

$vocab \leftarrow \{< |endoftext| >: 0, \, ! : 1, \, \backslash : 2, \ldots, \, 0 : 16, \, 1 : 17, \ldots, \, z : 257\}$

// Treeningtekstidest C eraldatakse tükeldatud sõnad s_i,
// mis loetakse kokku $count(s_i)$.

$S \leftarrow \{s_1 : count(s_1), s_2 : count(s_2) \ldots, s_m : count(s_m)\}$
iga i=$(258, \ldots, d_{vocab})$ korral
    // Leitakse kõige sagedasem tähe/tähtede paar
    // KõigeSagedasem tagastab kõige sagedasema tähe/tähtede paari,
    // kui see esineb. Kui enam pole midagi ühendada, siis $(t_1, t_2) = None$
    $(t_1, t_2) \leftarrow$ KõigeSagedasem(S)
    kui $(t_1, t_2) \neq None$ **siis**
        // Kõigis tükeldatud sõnades ühendatakse saadud paar kokku
        $S \leftarrow$ Ühenda$(S, (t_1, t_2))$
        // Ühendatud paar lisatakse sõnastikku
        $vocab = Lisa(vocab, (t_1, t_2))$
    muidu kui $(t_1, t_2) = None$ **siis**
        // Lõpeta iga. Sellisel juhul on sõnastik lühem kui etteantud
        // pikkus
        break
    lõpeta kui
lõpeta iga

Lisa 2. Näide BPE tokeniseerijaga tokeniseerimisest

Olgu tegemist BPE tokeniseerijaga, kus sõnastiku on näha järgnevast tabelis.

Tabel 8: Näidis sõnastik

Tekst	**Token**	Ühendatud
“<eos>”	0	13,2,6,9,14
“a”	1	
“e”	2	
“i”	3	
“k”	4	
“m”	5	
“o”	6	
“p”	7	
“r”	8	
“s”	9	
“ä”	10	
“ ”	11	
“.”	12	
“<”	13	
“>”	14	
“pa”	15	7,1
“ k”	16	11,4
“si”	17	9,3
“ma”	18	5,1
“par”	19	15,8
“em”	20	2,5
“äsi”	21	10,17
“ käsi”	22	16,21

Sõnastiku alguses on eritoken ja 14 baastokenit ning alates tokenist numbriga 15 (ehk 16. token) on saadud sõnastikku õppimise teel. Sõnastikus on näha, et kohe alguses on eritoken, mis on sõnastikku lisatud. Olgu tokeniseeritav sõna “parem käsi”. Kõigepealt lausest eraldatakse sõnad, ehk saadakse

((“parem”) (“ käsi”))

Seejärel iga täht/tähemärk lauses tokeniseeritakse. Sõnastiku põhjal saab

$$((7, 1, 8, 2, 5)\ (11, 4, 10, 9, 3)).$$

Järgnevalt ühendatakse tokenid samas järjestudes nagu nad sõnastikus on ehk alustatakse eritokenist ja siis õppimise käigus saadud ühendustest ning liigutakse suuremaks. Kuna eritokeneid ei esine, siis esimesena saab ühendada tokenid 7 ja 1 nagu on näha sõnastiku viimasest veerust. Saadakse

$$((15, 8, 2, 5)\ (11, 4, 10, 9, 3)).$$

See järel ühendatakse 11 ja 4 ning saadakse

$$((15, 8, 2, 5)\ (16, 10, 9, 3)).$$

Ühendatakse 9 ja 3

$$((15, 8, 2, 5)\ (16, 10, 17)).$$

Kuna järjestust 5 ja 1 ei eksisteeri, siis ei ole midagi ühendada. Liigutakse edasi ja ühendatakse 15 ja 8 ning nii edasi.

$$((19, 2, 5)\ (16, 10, 17))$$
$$((19, 20)\ (16, 10, 17))$$
$$((19, 20)\ (16, 21))$$
$$((19, 20)\ (22))$$

Tehes kõik ühendused saadakse, et tokeniseeritud lause on

$$(19, 20, 22).$$

Rakendades sõnastikku on näha, et algus lause saab tagasi kui asendada iga number

sõnastikus esineva tekstiga.

Lisa 3. GPT-2 osaline rakendamine

Järgnev on näide osadest GPT-2 mudeli kasutusel tehtavadest sammudest, kus esialgsed väärtused on fiktiivised. Olgu GPT-2 mudel, mis on treenitud kasutades Lisas 2 olevat sõnastikku, ehk sellisel juhul $d_{vocab} = 23$. Mudelit rakendatakse samuti lausele "parem käsi" nagu oli Lisas 2 ning mille lisatakse lause algus token. Olgu mudeli puhul tehtud eeldus, et $d_n = 5$. Kuna d_n on suurem kui tokenite arv sisendvektoris, siis kärpimist teostama ei pea. Kuna sisend on lühem kui d_n siis teostatakse *padding* ning sisend on $X = (0, 19, 20, 22, 0)$. *Padding*'una kasutatakse samuti tokenit "<eos>". See järel viiakse sisendvektor X one-hot kujule U.

Lisa 3.1 *Embedding* ja *positional encoding*

One-hot kujul $U : 5 \times 23$.

$$U = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

Olgu $d_{model} = 3$, siis

$$W_e = \begin{pmatrix} w_e^0 \\ w_e^1 \\ w_e^2 \\ \dots \\ w_e^{21} \\ w_e^{22} \end{pmatrix} = \begin{pmatrix} 0{,}1 & 0{,}3 & -0{,}2 \\ 0{,}8 & -0{,}1 & 0{,}04 \\ -0{,}1 & -0{,}4 & 0{,}2 \\ & \dots & \\ 0{,}3 & 0{,}2 & 0{,}1 \\ 0{,}4 & 0{,}6 & 0{,}2 \end{pmatrix}.$$

Korrutades U ja W_e saab maatriksi UW_e

$$UW_e = \begin{pmatrix} w_e^0 \\ w_e^{19} \\ w_e^{20} \\ w_e^{22} \\ w_e^0 \end{pmatrix} = \begin{pmatrix} 0{,}1 & 0{,}3 & -0{,}2 \\ 0{,}2 & -0{,}2 & 0{,}2 \\ 0{,}15 & -0{,}45 & -0{,}3 \\ 0{,}4 & 0{,}6 & 0{,}2 \\ 0{,}1 & 0{,}3 & -0{,}2 \end{pmatrix}.$$

Maatriksile UW_e liidetakse *positional encoding* maatriks

$$W_p = \begin{pmatrix} w_p^0 \\ w_p^1 \\ w_p^3 \\ w_p^4 \\ w_p^4 \end{pmatrix} = \begin{pmatrix} 0{,}05 & 0{,}03 & -0{,}03 \\ -0{,}04 & -0{,}09 & 0{,}08 \\ 0{,}05 & 0{,}04 & 0{,}02 \\ 0{,}01 & 0{,}07 & 0{,}06 \\ 0{,}01 & 0{,}04 & 0{,}07 \end{pmatrix}.$$

Liites mõlemad maatriksid saadakse

$$UW_e + W_p = \begin{pmatrix} w_e^0 + w_p^0 \\ w_e^{19} + w_p^1 \\ w_e^{20} + w_p^3 \\ w_e^{22} + w_p^4 \\ w_e^0 + w_p^4 \end{pmatrix} = \begin{pmatrix} 0{,}15 & 0{,}33 & -0{,}23 \\ 0{,}16 & -0{,}29 & 0{,}28 \\ 0{,}20 & -0{,}41 & -0{,}28 \\ 0{,}41 & 0{,}67 & 0{,}26 \\ 0{,}11 & 0{,}34 & -0{,}13 \end{pmatrix}.$$

Lisa 3.2 Väljajätumeetod ja normaliseerimine

Järgmisena genereeritakse maatriks B_0, kus elemendid on Bernoulli jaotusest ning leitakse $UW_e + W_p$ ja B_0 Hadamardi korrutis. Olgu Bernoulli jaoutuse parameeter

0,1, siis genereerides väärtused saab

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Tehes Hadamardi korrutise saadakse

$$A = (UW_e + W_p) \odot B = \begin{pmatrix} (w_e^0 + w_p^0) \odot b_0^0 \\ (w_e^{19} + w_p^1) \odot b_0^1 \\ (w_e^{20} + w_p^3) \odot b_0^2 \\ (w_e^{22} + w_p^4) \odot b_0^3 \\ (w_e^0 + w_p^4) \odot b_0^4 \end{pmatrix} = \begin{pmatrix} 0{,}15 & 0{,}33 & 0 \\ 0{,}16 & -0{,}29 & 0{,}28 \\ 0 & -0{,}41 & -0{,}28 \\ 0{,}41 & 0{,}67 & 0{,}26 \\ 0{,}11 & 0{,}34 & -0{,}13 \end{pmatrix}.$$

Järgmisena A normaliseeritakse. Selleks leitakse kõigepealt ridade keskväärtused ja standardhälbed.

$$\vec{\mu} = \frac{1}{3} A \vec{\mathbb{1}} = \begin{pmatrix} \frac{0{,}15+0{,}33-0{,}23}{3} \\ \frac{0{,}16-0{,}29+0{,}28}{3} \\ \frac{0{,}20-0{,}41-0{,}28}{3} \\ \frac{0{,}41+0{,}67+0{,}26}{3} \\ \frac{0{,}11+0{,}34-0{,}13}{3} \end{pmatrix} = \begin{pmatrix} \frac{0{,}25}{3} \\ \frac{0{,}15}{3} \\ \frac{-0{,}49}{3} \\ \frac{1{,}34}{3} \\ \frac{0{,}32}{3} \end{pmatrix}$$

$$\vec{\sigma} = \begin{pmatrix} \sqrt{\frac{(0{,}16-\frac{0{,}25}{3})^2+(0{,}29-\frac{0{,}25}{3})^2+(0{,}28-\frac{0{,}25}{3})^2}{3}} \\ \sqrt{\frac{(0{,}15-\frac{0{,}15}{3})^2+(0{,}33-\frac{0{,}15}{3})^2-(0{,}23-\frac{0{,}15}{3})^2}{3}} \\ \sqrt{\frac{(0{,}20-\frac{-0{,}49}{3})^2+(0{,}41-\frac{-0{,}49}{3})^2-(0{,}28-\frac{-0{,}49}{3})^2}{3}} \\ \sqrt{\frac{(0{,}41-\frac{1{,}34}{3})^2+(0{,}67-\frac{1{,}34}{3})^2+(0{,}26-\frac{1{,}34}{3})^2}{3}} \\ \sqrt{\frac{(0{,}11-\frac{0{,}32}{3})^2+(0{,}34-\frac{0{,}32}{3})^2-(0{,}13-\frac{0{,}32}{3})^2}{3}} \end{pmatrix} \approx \begin{pmatrix} 0{,}1706 \\ 0{,}1366 \\ 0{,}2967 \\ 0{,}1694 \\ 0{,}1341 \end{pmatrix}$$

Olgu vektor $\vec{\mathbf{g}} = (0{,}1, -0{,}1, 0{,}3)'$ ja vektor $\vec{\mathbf{b}} = (-0{,}1, -0{,}05, 0{,}2)'$. Normaliseerides A saab

$$Norm(A) = \begin{pmatrix} \frac{(0{,}1,-0{,}1,0{,}3)}{0{,}1706} \odot (0{,}15 - \frac{0{,}25}{3}, 0{,}33 - \frac{0{,}25}{3}, 0 - \frac{0{,}25}{3}) + (-0{,}1, -0{,}05, 0{,}2) \\ \frac{(0{,}1,-0{,}1,0{,}3)}{0{,}1366} \odot (0{,}16 - \frac{0{,}15}{3}, -0{,}29 - \frac{0{,}15}{3}, 0{,}28 - \frac{0{,}15}{3}) + (-0{,}1, -0{,}05, 0{,}2) \\ \frac{(0{,}1,-0{,}1,0{,}3)}{0{,}2967} \odot (0 + \frac{0{,}49}{3}, -0{,}41 + \frac{0{,}49}{3}, -0{,}28 + \frac{0{,}49}{3}) + (-0{,}1, -0{,}05, 0{,}2) \\ \frac{(0{,}1,-0{,}1,0{,}3)}{0{,}1694} \odot (0{,}41 - \frac{1{,}34}{3}, 0{,}67 - \frac{1{,}34}{3}, 0{,}26 - \frac{1{,}34}{3}) + (-0{,}1, -0{,}05, 0{,}2) \\ \frac{(0{,}1,-0{,}1,0{,}3)}{0{,}1341} \odot (0{,}11 - \frac{0{,}32}{3}, 0{,}34 - \frac{0{,}32}{3}, -0{,}13 - \frac{0{,}32}{3}) + (-0{,}1, -0{,}05, 0{,}2) \end{pmatrix} =$$

$$\begin{pmatrix} -0{,}0609 & -0{,}1946 & 0{,}0535 \\ -0{,}0195 & 0{,}1989 & 0{,}7051 \\ -0{,}0450 & 0{,}0331 & 0{,}0820 \\ -0{,}1216 & -0{,}1818 & -1{,}1306 \\ -0{,}0975 & -0.2240 & -0{,}3295 \end{pmatrix} = H_0$$

Lisa 3.3 *Attention*

Näites rakendadatakse lihtsuse huvides *self-attention*'i mitte *Multi-Head Attention*'i. Rakendades *self-attention*'i saab

$$Attention(H_0, H_0, H_0) = softmax\left(Mask\left(\frac{H_0 H_0^T}{\sqrt{3}}\right)\right) H_0 =$$

$$softmax\left(Mask\left(\begin{pmatrix} 0{,}0257 & 0{,}0001 & 0{,}0004 & -0{,}0102 & 0,0184 \\ 0{,}0001 & 0{,}3101 & 0{,}0377 & -0{,}4798 & -0,1588 \\ 0{,}0004 & 0{,}0377 & 0{,}0057 & -0{,}0538 & -0,0173 \\ -0{,}0102 & -0{,}4798 & -0{,}0538 & 0{,}7656 & 0,2454 \\ 0{,}0184 & -0{,}1588 & -0{,}0173 & 0{,}2454 & 0,0971 \end{pmatrix}\right)\right) H_0 =$$

$$softmax\left(\begin{pmatrix} 0{,}0257 & -\infty & -\infty & -\infty & -\infty \\ 0{,}0001 & 0{,}3101 & -\infty & -\infty & -\infty \\ 0{,}0004 & 0{,}0377 & 0{,}0057 & -\infty & -\infty \\ -0{,}0102 & -0{,}4798 & -0{,}0538 & 0{,}7656 & -\infty \\ 0{,}0184 & -0{,}1588 & -0{,}0173 & 0{,}2454 & 0,0971 \end{pmatrix}\right) H_0 =$$

$$\begin{pmatrix} 1{,}0000 & 0{,}0000 & 0{,}0000 & 0{,}0000 & 0{,}0000 \\ 0{,}4231 & 0{,}5769 & 0{,}0000 & 0{,}0000 & 0{,}0000 \\ 0{,}3286 & 0{,}3411 & 0{,}3303 & 0{,}0000 & 0{,}0000 \\ 0{,}2103 & 0{,}1315 & 0{,}2013 & 0{,}4569 & 0{,}0000 \\ 0{,}1946 & 0{,}1630 & 0{,}1878 & 0{,}2442 & 0{,}2105 \end{pmatrix} \begin{pmatrix} -0{,}0609 & -0{,}1946 & 0{,}0535 \\ -0{,}0195 & 0{,}1989 & 0{,}7051 \\ -0{,}0450 & 0{,}0331 & 0{,}0820 \\ -0{,}1216 & -0{,}1818 & -1{,}1306 \\ -0{,}0975 & -0.2240 & -0{,}3295 \end{pmatrix} \approx$$

$$\begin{pmatrix} -0{,}0609 & -0{,}1946 & 0{,}0535 \\ -0{,}0370 & 0{,}0324 & 0{,}4294 \\ -0{,}0415 & 0{,}0148 & 0{,}2852 \\ -0{,}0800 & -0{,}0912 & -0{,}3961 \\ -0{,}0737 & -0{,}0908 & -0{,}2047 \end{pmatrix}$$

Olgu Bernoulli jaotuse korral genereeritud järgnevad maatriksid

$$B_{0|0} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

ja

$$B_{0|1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

ning normaliseerimis jaoks olgu vektor $\vec{\mathbf{g}} = (-0{,}27, 0{,}59, 0{,}18)'$ ja vektor $\vec{\mathbf{b}} = (0{,}01, -0{,}38, 0{,}07)'$. Siit saab, et

$$Vahekiht_0 = Norm(Attention(H_0, H_0, H_0) \odot B_{0|0} + H_0 \odot B_{0|0}) =$$

$$Norm\left(\begin{pmatrix} -0{,}0609 & -0{,}1946 & 0{,}0535 \\ -0{,}0370 & 0{,}0324 & 0{,}4294 \\ -0{,}0415 & 0{,}0148 & 0 \\ -0{,}0800 & -0{,}0912 & 0 \\ -0{,}0737 & -0{,}0908 & -0{,}2047 \end{pmatrix} + \begin{pmatrix} -0{,}0609 & -0{,}1946 & 0{,}0535 \\ -0{,}0195 & 0 & 0{,}7051 \\ -0{,}0450 & 0{,}0331 & 0{,}0820 \\ -0{,}1216 & -0{,}1818 & -1{,}1306 \\ -0{,}0975 & -0.2240 & -0{,}3295 \end{pmatrix}\right) =$$

$$Norm\left(\begin{pmatrix} -0{,}1218 & -0{,}3892 & 0{,}1070 \\ -0{,}0565 & 0{,}0324 & 1{,}1345 \\ -0{,}0865 & 0{,}0479 & 0{,}0820 \\ -0{,}2016 & -0{,}2730 & -1{,}1306 \\ -0{,}1712 & -0{,}3148 & -0{,}5342 \end{pmatrix}\right) \approx \begin{pmatrix} -0{,}0745 & -4{,}0322 & 1{,}1279 \\ 0{,}4025 & -1{,}0590 & 0{,}5389 \\ 5{,}1624 & 3{,}3482 & 2{,}3675 \\ -0{,}4953 & 0{,}4878 & -0{,}5316 \\ -2{,}0363 & 0{,}2891 & -1{,}4984 \end{pmatrix}$$

Lisa 3.4 Ühe peidetud kihiga närvivõrk

Olgu peidetud kihi suurus 6. Sellisel juhul on fiktiivsed kaalude maatriksid järgnevad

$$W_{0|0} = \begin{pmatrix} -0{,}74 & 1{,}31 & -0{,}38 & 0{,}006 - 1{,}03 & 2{,}68 \\ 0{,}26 & -0{,}01 & -1{,}67 & 0{,}5960{,}52 & 0{,}90 \\ 0{,}48 & 0{,}21 & 0{,}05 & 0{,}7060{,}97 & 2{,}14 \end{pmatrix}$$

$$W_{0|1} = \begin{pmatrix} -0{,}77 & 0{,}62 & 0{,}67 \\ -2{,}04 & -0{,}98 & -0{,}03 \\ -0{,}60 & -1{,}77 & -0{,}57 \\ -1{,}73 & 0{,}97 & -0{,}98 \\ -0{,}68 & 0{,}84 & -1{,}22 \\ -0{,}28 & 0{,}39 & 0{,}09 \end{pmatrix}$$

$$\vec{\mathbf{b}}'_{0|0} = (0{,}40, -0{,}36, -0{,}73, -2{,}36, 0{,}00, 0{,}51)$$
$$\vec{\mathbf{b}}'_{0|1} = (-1{,}17, 0{,}41, 0.08)$$

Rakendades närvivõrku saab

$$\text{GELU}(Vahekiht_0 W_{0|0} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0})W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1} =$$

$$\text{GELU}\left(\begin{pmatrix} -0{,}0745 & -4{,}0322 & 1{,}1279 \\ 0{,}4025 & -1{,}0590 & 0{,}5389 \\ 5{,}1624 & 3{,}3482 & 2{,}3675 \\ -0{,}4953 & 0{,}4878 & -0{,}5316 \\ -2{,}0363 & 0{,}2891 & -1{,}4984 \end{pmatrix} \begin{pmatrix} -0{,}74 & 1{,}31 & -0{,}38 & 0{,}006 & -1{,}03 & 2{,}68 \\ 0{,}26 & -0{,}01 & -1{,}67 & 0{,}596 & 0{,}52 & 0{,}90 \\ 0{,}48 & 0{,}21 & 0{,}05 & 0{,}706 & 0{,}97 & 2{,}14 \end{pmatrix} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0}\right)$$

$$W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1} \approx$$

$$\text{GELU}\left(\begin{pmatrix} -0{,}4518 & 0{,}1796 & 6{,}8185 & -1{,}5895 & -0{,}9259 & -1{,}4149 \\ -0{,}3145 & 0{,}6510 & 1{,}6425 & -0{,}2476 & -0{,}4425 & 1{,}2788 \\ -1{,}8132 & 7{,}2264 & -7{,}4348 & 3{,}6327 & -1{,}2797 & 21{,}9151 \\ 0{,}2382 & -0{,}7654 & -0{,}6530 & -0{,}0843 & 0{,}2482 & -2{,}0260 \\ 0{,}8628 & -2{,}9851 & 0{,}2161 & -0{,}8783 & 0{,}7943 & -8{,}4037 \end{pmatrix} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|0}\right)$$

$$W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1} \approx$$

$$\text{GELU}\left(\begin{pmatrix} -0{,}0518 & -0{,}1804 & 6{,}0885 & -3{,}9495 & -0{,}9259 & -0.9049 \\ 0{,}0855 & 0{,}2910 & 0{,}9125 & -2{,}6076 & -0{,}4425 & 1.7888 \\ -1{,}4132 & 6{,}8664 & -8{,}1648 & 1{,}2727 & -1{,}2797 & 22.4251 \\ 0{,}6382 & -1{,}1254 & -1{,}3830 & -2{,}4443 & 0{,}2482 & -1.5160 \\ 1{,}2628 & -3{,}3451 & -0{,}5139 & -3{,}2383 & 0{,}7943 & -7.8937 \end{pmatrix}\right) W_{0|1} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1} \approx$$

$$\begin{pmatrix} -0{,}0250 & -0{,}0799 & 6{,}0878 & -0{,}0181 & -0{,}2171 & -0{,}2166 \\ 0{,}0451 & 0{,}1722 & 0{,}6957 & -0{,}0825 & -0{,}1605 & 1{,}6273 \\ -0{,}1978 & 6{,}8662 & 0{,}0000 & 1{,}0646 & -0{,}2077 & 22{,}4251 \\ 0{,}4423 & -0{,}2154 & -0{,}2003 & -0{,}0962 & 0{,}1436 & -0{,}1888 \\ 1{,}0541 & -0{,}0378 & -0{,}1757 & -0{,}0427 & 0{,}5829 & 0{,}0000 \end{pmatrix} \begin{pmatrix} -0{,}77 & 0{,}62 & 0{,}67 \\ -2{,}04 & -0{,}98 & -0{,}03 \\ -0{,}60 & -1{,}77 & -0{,}57 \\ -1{,}73 & 0{,}97 & -0{,}98 \\ -0{,}68 & 0{,}84 & -1{,}22 \\ -0{,}28 & 0{,}39 & 0{,}09 \end{pmatrix} + \vec{\mathbb{1}}\vec{\mathbf{b}}'_{0|1} \approx$$

$$\begin{pmatrix} -3{,}2308 & -10{,}9970 & -3{,}2213 \\ -1{,}0072 & -0{,}9524 & 0{,}0516 \\ -21{,}8343 & 2{,}7525 & 0{,}8898 \\ 0{,}3407 & 0{,}7935 & 0{,}3191 \\ -0{,}9516 & 1{,}4498 & 0{,}1382 \end{pmatrix} + \begin{pmatrix} -1{,}17 & 0{,}41 & 0.08 \\ -1{,}17 & 0{,}41 & 0.08 \\ -1{,}17 & 0{,}41 & 0.08 \\ -1{,}17 & 0{,}41 & 0.08 \\ -1{,}17 & 0{,}41 & 0.08 \end{pmatrix} =$$

$$\begin{pmatrix} -4{,}4008 & -10{,}5870 & -3{,}1413 \\ -2{,}1772 & -0{,}5424 & 0{,}1316 \\ -23{,}0043 & 3{,}1625 & 0{,}9698 \\ -0{,}8293 & 1{,}2035 & 0{,}3991 \\ -2{,}1216 & 1{,}8598 & 0{,}2182 \end{pmatrix}$$

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Mihkel Lepson,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Epikriisi tekstide genereerimine GPT-2 mudeliga, mille juhendaja on Raivo Kolde, 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 4.0, 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 intellektuaaloman-di ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Mihkel Lepson

16.05.2023