TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu kolledž

REAALAJAS KÕNELEJATE ERISTAMINE JA TEEMAJAOTUS MULTIMEEDIA INDEKSEERIMISEKS

Real-time speaker diarization and topic segmentation for multimedia indexing

BAKALAUREUSETÖÖ

Üliõpilane: Joe Reiman
/nimi/

Üliõpilaskood 192987EDTR

Juhendaja: Aime Ruus
/nimi, amet/

Tartu 2023

(Tiitellehe pöördel)

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

05.2023
Autor: Joe Reiman / allkirjstatud digitaalselt /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

".....” ..... 20…..

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud

".....”.....20… .

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Joe Reiman

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Reaalajas kõnelejate eristamine ja teemajaotus multimeedia indekseerimiseks", mille juhendaja on Aime Ruus,

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

05.2023

TalTech Instituudi nimetus

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Joe Reiman, 192987EDTR

Õppekava, peaeriala: EDTR17/18 Telemaatika ja arukad süsteemid

Juhendaja(d): Programmijuht Aime Ruus, +372 53402823

Lõputöö teema:

Reaalajas kõnelejate eristamine ja teemajaotus multimeedia indekseerimiseks.

Real-time speaker diarization and topic segmentation for multimedia indexing

Lõputöö põhieesmärgid:

1. Sobiva eeltreenitud kõnetuvastus mudeli või abistava teegi leidmine.
2. Närvivõrgustike optimeerimistehnikad.
3. Sobiva teemamudeli arendamine.

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Lõputöö teema valimine	25.02.23
2.	Eelkaitsmine	05.05.23
3.	Lõputöö esitamine	22.05.23
4.	Lõputöö kaitsmine	09.05.23

Töö keel: Eesti keel **Lõputöö esitamise tähtaeg:** 22. mai 2023. a

Üliõpilane: Joe Reiman / allkirjastatud digitaalselt /

Juhendaja: Aime Ruus / allkirjastatud digitaalselt /

Programmijuht: Aime Ruus / allkirjastatud digitaalselt /
/allkiri/

SISUKORD

TABELITE LOETELU

JOONISTE LOETELU

LÜHENDITE JA TÄHISTE LOETELU

ASR (Automatic Speech Recognition) - Automaatne kõnetuvastus.
CNN (Convolutional Neural Network) - Konvolutsiooniline närvivõrk.
DNN (Deep Neural Network) - Süvanärvivõrk.
RNN (Recurrent Neural Network) – Rekurrentne võrk.
MFCC(Mel Frequency Cepstral Coefficients) – Mel-frekventsi tsepstraal koefitsiendid.
HMM (Hidden Markov Machine) – Peidetud Markovi mudel.
LLM (Large Language Model) – Suur keelemudel.
NLP (Natural Language Processing) - Loomuliku keele töötlus.

SISSEJUHATUS

Multimeedia sisu, eriti taskuhäälingute, veebikonverentside ja videoloengute kiire kasv on tekitanud nõudluse efektiivsete ja täpsete viiside järele selle suure hulga andmete indekseerimiseks, otsimiseks ja läbivaatamiseks. Sellise süsteemi eesmärk oleks teha multimeedia sisu kasutajate jaoks kättesaadavamaks, lähtudes kõnelejatest ja arutatud teemadest.

Üha enam kasvab vajadus ja soov kasutada tehnoloogiat, et hõlbustada ja automatiseerida igapäevaseid toiminguid. Üks valdkond, kus on täheldatav kasvav huvi, on helifailide automaatne transkribeerimine. Kujutage ette, et kuulate huvitavat taskuhäälingut, kuid soovite selle sisu üle vaadata või jagada - selle asemel, et kogu sisu uuesti läbi kuulata, oleks kasulik saada kiire ja täpne transkriptsioon, mis toob esile peamised teemad ja lõigud.

Antud töö eesmärgiks oli luua rakendus, mis suudab transkribeerida helifaili sisu tekstina. See võimaldab inimestel saada kiire ülevaate sellest, mida helifail sisaldab, ilma et peaks kogu sisu läbi kuulama. Kuid see ei ole ainus funktsioon. Töödeldes helifaili, jagab rakendus selle lõikudeks, igaüks erineva teemaga. Seejärel määrab rakendus igale teemalõigule sobiva pealkirja. See funktsioon võimaldab kasutajal leida kiiresti ja hõlpsalt konkreetseid teemasid või punkte, mida soovitakse üle vaadata või jagada.

See rakendus on eriti kasulik taskuhäälingute kontekstis. Taskuhäälingud on viimastel aastatel muutunud väga populaarseks meediumiks erinevate teemade tutvustamiseks ja arutamiseks. Kuid nende pikkus ja sisu võivad muuta nende kuulamise ajamahukaks. Rakendus, mis luuakse antud uurimistöö käigus, võimaldab taskuhäälingute sisu kiiremini ja hõlpsamalt omastada. Lisaks aitab see kaasa taskuhäälingute laiemale levikule, muutes need hõlpsamini jagatavaks ja arusaadavamaks kõigile, ka neile, kes ei pruugi olla valdkonna eksperdid.

Töö esimene pool keskendub valdkonna levinumate töö tausta, arhitektuuride ja tööriistade uurimisele, et anda taustainfo uurimistöö rakendusliku osa mõistmiseks. Teises osas antakse ülevaade kirjandusest ja vaadatakse eksisteerivaid lahendusi. Kolmas osa kirjeldab reaalaja mudeli kavandatavat rakenduse arhitektuuri, metoodikaid ja selle komponente. Neljandas osas on kogu süsteemi kokkupanemine ja peale seda on süsteemi analüüsimine.

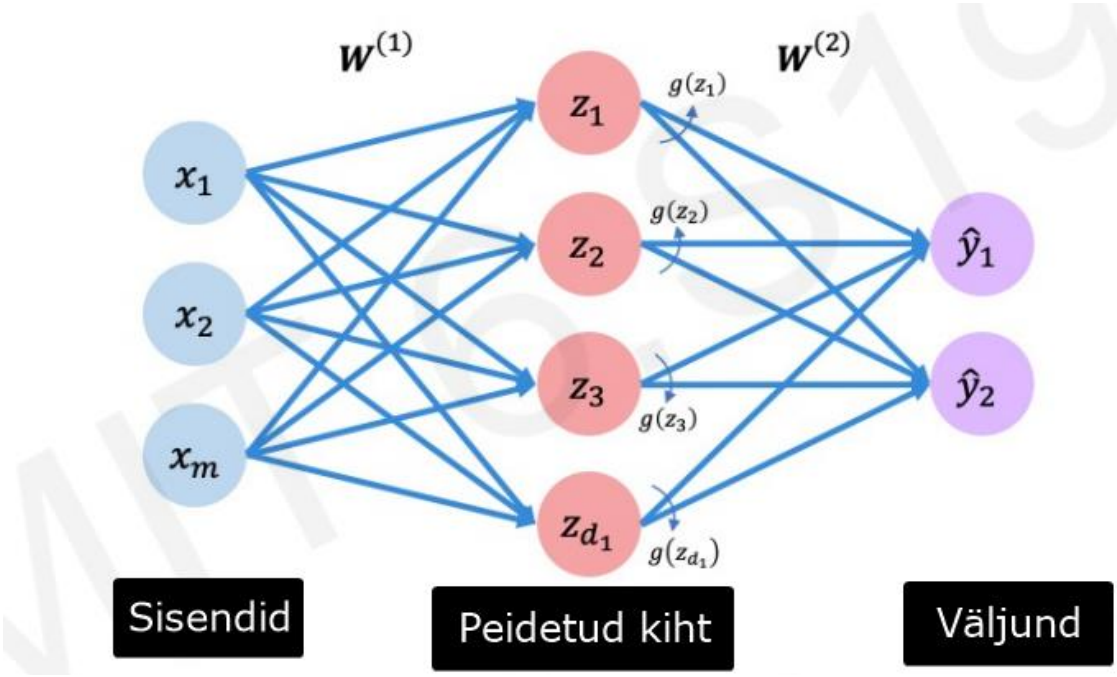
1 KIRJANDUSE ÜLEVAADE

Selles peatükis antakse ülevaade olemasolevatest uurimistest, metoodikatest ja tehnoloogiatest kõnetuvastuse, kõnelejate eristamise ja teemade modelleerimise valdkondades. See ülevaade aitab luua konteksti ja tausta antud uurimistööle, et oleks paremini mõistetav, miks tehnoloogiliste edasiminekute tõttu antud valdkonnas on võimalik tänapäevasemate metooditega saavutada reaalajas kõneeristus.

1.1 Tehisnärvivõrk

Tehisnärvivõrkude tuumaks on neuronite kogum, mis on korraldatud erinevatesse kihtidesse: sisendkiht, üks või mitu peidetud kihti ning väljundkiht. Neuronitel on üks kuni mitu sisendit ja üks väljund. Sisendkiht võtab vastu toorandmeid, olenevalt kas muudetud või muutmata kujul, samal ajal kui väljundkiht toodab lõplikke ennustusi või klassifikatsioone. Nende kahe vahel võib olla üks või mitu peidetud kihti, mis teostavad arvutusi, et muuta sisend millekski, mida väljundkiht saab kasutada[1][2].

Neuronid on närvivõrgu põhielemendid. Need moodustavad erinevaid kihte, mida närvivõrk kasutab sisenditest mustrite leidmiseks ja probleemide lahendamiseks. Üksik neuron töötleb sisendeid, mida ta saab teistelt neuronitelt või otse sisendkihilt. See töötlus toimub kahe põhietapi kaudu: kaalutud summa arvutamine ja aktiveerimisfunktsiooni rakendamine[2].



Joonis 1.1 Lihtsaim kolmekihiline närvivõrgustik[3].

Joonisel 1.1 on näha tavalise 3 kihilise närvivõrgustiku arhitektuuri, mis koosneb sisendite, peidetud kihi ja väljundi kihist. Tegemist on ka täielikult ühendatud kihtidega, sest igas kihis asetsevad neuronid on ühendatud järgneva kihi kõigi neuronitega[3].
X_n neuronid on närvivõrgustiku sisendandmed, Z_n kiht on peidetud kiht ja Y kihis on väljundi neuronid.
W näitab neuronite ühenduste vahelisi kaalusid.

Kaalutud summa arvutamine seisneb selles, et neuron korrutab igale saabuvale sisendile vastava kaalu ning liidab need kokku. Seejärel lisatakse kaalutud summale nihe (ing. k *bias*), mis on omamoodi reguleeritav konstant[2] ja aitab optimeerida närvivõrgustike kaalusid.
Arvutuskäik näeb välja selline: kaalutud_summa = Σ(kaal_i * sisend_i) + nihe

Aktiveerimisfunktsioon võtab kaalutud summa ja muundab selle neuronilt väljuvaks väärtuseks. Erinevate aktiveerimisfunktsioonide kasutamine võimaldab närvivõrgul modelleerida keerukamaid mudeleid, mis suudavad esindada mitte-lineaarseid seoseid. Näiteks, lineaarne aktiveerimisfunktsioon lihtsalt väljastab sisendi muutmata kujul, samas kui mittelineaarne aktiveerimisfunktsioon nagu ReLU (*Rectified Linear Unit*) väljastab sisendi kui see on positiivne ja nulli kui see on negatiivne[1][2].
Arvutuskäik näeb välja selline: ReLU_funktsioon(x) = max(0, x)

Aktiveerimisfunktsioonide hulka kuuluvad ka **sigmoid, hüperboolne tangensfunktsioon** (tanh) ja **softmax** funktsioonid.

Sigmoidfunktsioon muudab sisendi vahemikku 0 kuni 1, mis on kasulik tõenäosuste modelleerimiseks. **Hüberboolne tangensfunktsioon** funktsioon on sarnane sigmoidfunktsioonile, kuid muudab sisendi vahemikku -1 kuni 1, andes seega rohkem paindlikkust. **Softmax** funktsioon on eriti kasulik mitmeklassiliste klassifikatsiooniprobleemide korral, kuna see muudab sisendi vektori tõenäosuste vektoriks[1][2].

Närvivõrgu **õppimisprotsess** seisneb nende kaalude ja niheparameetrite kohandamises nii, et võrk suudaks andmeid võimalikult hästi modelleerida. See õppimine toimub algoritmi, näiteks stohhastilise gradientlanguse (SGD) abil, mis minimeerib veafunktsiooni, mõõtes võrgu ennustuste ja tegelike väärtuste vahelist

erinevust. Algoritm kohandab kaalusid ja nihet gradientide abil, mis on veafunktsiooni derivaadid nende parameetrite suhtes. Gradientide abil "liigub" algoritm parameetrite ruumis suunas, mis vähendab veafunktsiooni väärtust[1][2].
Arvutuskäik näeb välja selline: kaal_i = kaal_i - õppimiskiirus * gradient

Aktiveerimisfunktsioonid mängivad olulist rolli gradientide arvutamisel, kuna need määravad, kuidas vead levivad võrgus tagasi. Näiteks, kui aktiveerimisfunktsioon on ReLU, on gradient positiivse sisendi korral 1 ja negatiivse sisendi korral 0. See tähendab, et positiivse sisendi korral levivad vead tagasi ilma muutusteta, samas kui negatiivse sisendi korral vead ei levi üldse. See omadus võib mõjutada õppimise dünaamikat ja võrgu võimet erinevaid mustreid ära tunda[1][2].
Arvutuskäik näeb välja selline: ReLU_gradient(x) = 1 if x > 0 else 0

Kokkuvõtteks, neuronid ja aktiveerimisfunktsioonid on närvivõrgu põhielemendid, mis võimaldavad võrgul õppida ja modelleerida seoseid andmetes.
Aktiveerimisfunktsioonid määravad, kuidas neuronid sisendit töötlevad ja milliseid väärtuseid nad väljastavad, samal ajal kui õppimisalgoritmid kohandavad võrgu kaalusid ja nihet, et minimeerida ennustusvead.

1.1.1 Närvivõrkudest sügavate närvivõrkudeni

Närvivõrkude areng on viimase mõnekümne aasta jooksul olnud silmatorkav, eriti arvestades üleminekut lihtsatest tehisnärvivõrkudest (ANN) sügavate õppimisvõrkude (DNN) poole. Võrgud millest eelpool juttu oli, nimetatakse ANN arhitektuurideks. ANN arhitektuurid on tehisnärvivõrkude esimene ja kõige lihtsam tüüp. Neid iseloomustab see, et kõik neuronid on korraldatud järjestikustesse kihtidesse, kusjuures iga neuroni sisend on pärit ainult varasematest kihtidest. ANN võrke võib olla erinevat tüüpi, sõltuvalt nende kujundusest: võivad olla tagasisidevõrgud, kus signaal võib liikuda mõlemas suunas, või edasisidevõrgud, kus signaal liigub ühes suunas - sisendkihist väljundkihini[1][4][5].

Sügavad närvivõrgud (DNN) on tehisnärvivõrkude loomulik ja vajalik areng. Need võrgud sisaldavad mitmeid peidetud kihte, mis võimaldab neil õppida keerukamaid ja abstraktsemaid funktsioone. DNNid võimaldavad masinõppesüsteemidel tegeleda palju keerukamate probleemidega kui traditsioonilised väiksemad närvivõrgud[5][6]. Tasub ära mainida, et ei ole päris selget definitsiooni kust maalt üks ANN muutub DNN võrgustikuks, aga üldiselt kui on tegemist juba enama kui kahe varjatud kihiga, siis loetakse see DNN närvivõrguks[1][5].

Ülemineku põhjuseks oli vajadus parema täpsuse, keerukuse ja paindlikkuse järele. Näiteks kui tahame tuvastada pilte, võib lihtne ANN tuvastada ainult lihtsaid ja väga konkreetseid omadusi, näiteks konkreetsed värvid või sirged jooned. Seevastu DNN suudab õppida keerukamaid ja abstraktsemaid tunnuseid, näiteks näojooni või spetsiifilisi objekte. Seega saame DNN-ga paremaid tulemusi keerukamate ülesannete puhul[5][7][8].

Lisaks võimaldas sügavamate võrkude kasutuselevõtt ka keerukamate struktuuride kasutamist. Näiteks konvolutsioonilised närvivõrgud (CNN) kasutavad ruumilisi hierarhiaid, et tuvastada kohalikke mustreid piltidel, ja pika lühiajalise mälu (LSTM) võrgud võimaldavad süsteemil õppida pikki ajaliselt seotud mustreid, nagu seda on vaja näiteks kõne- või tekstiandmete korral[9]–[12].

Näiteks, 2015. aastal ImageNeti andmekogumiga töötades suutsid DNN-d saavutada täpsuse, mis ületas inimeste võimet tuvastada objekte fotodel. Samuti on DNN-id leidnud laialdast rakendust kõnetuvastuses, masintõlkes ja paljudes teistes valdkondades[13].

1.1.2 Rekurrentsed närvivõrgud (*Recurrent Neural Network*)

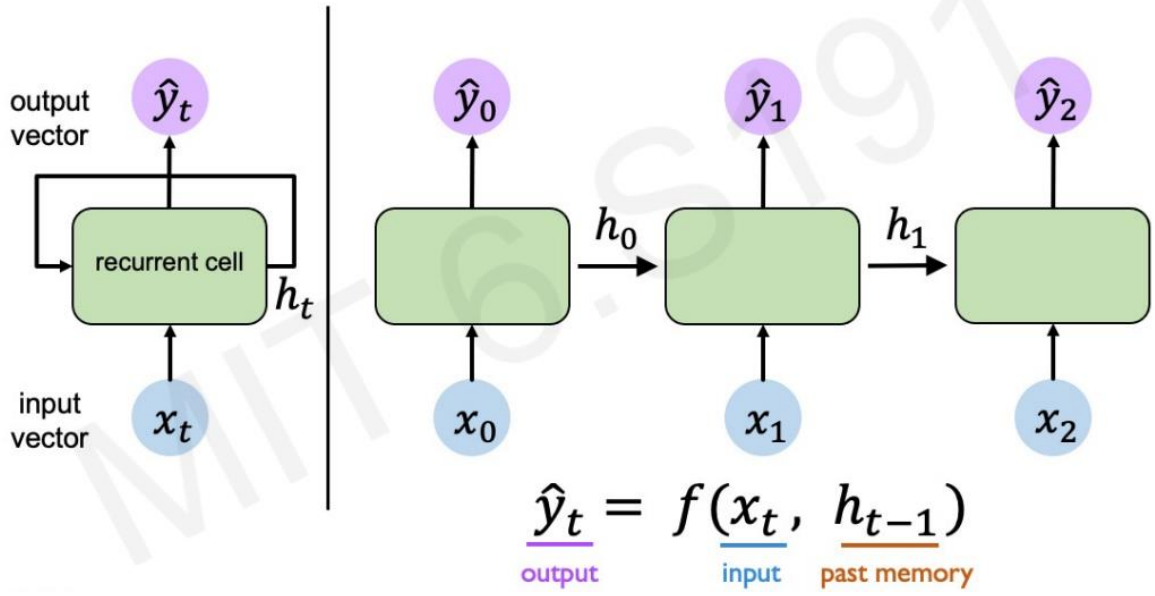
on täiendav klass närvivõrkudest, millel on eripärane võime töödelda järjestuslikke andmeid. See võime tuleneb asjaolust, et RNN-id kasutavad sisemist mälu, mis võimaldab neil eelmisi sisendeid "mäletada" ja seda teavet kasutada uute otsuste tegemisel. See teeb RNN-idest suurepärase valiku selliste ülesannete jaoks, kus on oluline järjestus, nagu loomuliku keele töötlemine (ingl k *Natural Language Processing*) ja kõnetuvastus[4][9][11][14].

RNN arhitektuuri eripära on see, et see sisaldab tagasipöörduvat sidet, mis ühendab iga võrgu kihi tagasi iseendaga. See tähendab, et võrgu seisund ajahetkel 't' mitte ainult ei sõltu praegusest sisendist, vaid ka võrgu varasemast seisundist ajahetkel 't-1'. See võimaldab RNN-il säilitada teavet, mis on "ajas" järjestikune, ja kasutada seda teavet otsuste tegemisel. Selline arhitektuur on eriti kasulik ülesannete puhul, kus on oluline mõista andmete sisemist struktuuri ja sõltuvusi[14].

Klassikalise RNN-i arhitektuuri kõige elementaarsem vorm koosneb ühest või mitmest sisemisest (peidetud) kihist, mis on ühendatud sisend- ja väljundkihtidega. Iga peidetud kiht sisaldab neuroneid, mis kasutavad aktiveerimisfunktsioone, nagu näiteks hüperboolne tangensfunktsioon (tanh) või ReLU (*Rectified Linear Unit*),

andmete töötlemiseks. Igal peidetud kihil on ka tagasisideühendus, mis võimaldab tal saada sisendit varasemate ajahetkede seisundist[14].

RNN-id on laialdaselt kasutusel loomuliku keele töötlemise ja kõnetuvastuse valdkonnas. Näiteks võib RNN-i kasutada selliste ülesannete jaoks nagu masintõlge, kus on vaja mõista sisendteksti järjestuslikku struktuuri ja genereerida vastav tõlge. RNN-id on samuti kasulikud kõnetuvastuse ülesannete jaoks, kuna nad suudavad "mäletada" varasemaid helisegmente ja kasutada seda teavet uute segmentide klassifitseerimiseks[4][10][14].



Joonis 1.1 RNN arhitektuur[15].

Vaadates täpsemalt RNN-i tööd Joonise 1.4 põhjal, võtame näiteks kõige lihtsama vormi, mis koosneb ühest peidetud kihist. Iga ajahetke 't' korral saab peidetud kiht sisendi 'x[t]' ja arvutab uue sisemise seisundi 'h[t]'. See seisund arvutatakse kasutades kahte maatriksit 'x[t]' ja varasemat sisemist seisundit 'h[t-1]', ning aktiveerimisfunktsiooni, tavaliselt hüperboolne tangensfunktsioon või ReLU.

RNN-i võime "mäletada" varasemaid sisendeid on selle tugevus, kuid samas ka nõrkus, kuna see võib põhjustada kahe olulise probleemi ilmnemist: lühiajalise mälu probleemi ja gradientide kadumise või plahvatamise probleemi. Lühiajalise mälu probleem tähendab, et RNN-idel on raskusi olulise teabe meeldejätmisega pikema aja jooksul, kuna varasemate sisendite mõju väheneb järk-järgult. Gradientide kadumise või plahvatamise probleem tuleneb aga sellest, et RNN-i ajas tagasi pöörates võivad gradientide väärtused kas väga kiiresti kasvada või väheneda, muutes võrgu õppimise ebastabiilseks[14].

Sellest hoolimata on tänapäeval saadaval mitmeid meetodeid, mis aitavad nendest probleemidest üle saada. Üks populaarsemaid on LSTM-id ehk pika lühiajalise mälu (*Long Short-Term Memory*) üksused, mis on spetsiaalselt välja töötatud gradientide kadumise probleemi lahendamiseks ja mis võimaldavad RNN-il tõhusalt töödelda pikemaid järjestusi. Teine meetod on väravatega tagasipöörduvad üksused (*Gated Recurrent Units*), mis on LSTM-idega sarnased, kuid mõnevõrra lihtsamad ja vähem ressursimahukad[14].

RNNide kasutamine loomuliku keele töötlemise (*Natural Language Processing*) ja kõnetuvastuse kontekstis tuleneb nende võimest töödelda järjestikuseid andmeid. Kuna nii tekst kui ka kõne esinevad järjestikusena, on RNNid ideaalsed selliste andmete modelleerimiseks. Näiteks kõnetuvastuse puhul võib RNN-i kasutada sisendina kõne raamidena ja väljundina vastavad transkriptsioonid. RNN "mäletab" varasemaid raame ja kasutab seda teavet praeguse raami mõistmiseks. Sarnaselt saab RNN-i kasutada ka teksti genereerimiseks, kus iga sõna või sümboli genereerimine põhineb eelneval kontekstil. Loomuliku keele töötlemise puhul on RNN-id kasulikud näiteks teksti genereerimiseks, kus iga järgmine sõna või sümbol genereeritakse eelmise konteksti põhjal. Samuti kasutatakse neid sageli masintõlkes, kus üks RNN (nn kooder) loeb sisendteksti ja teine RNN (nn dekooder) genereerib tõlke. RNN-e saab kasutada ka teksti klassifitseerimiseks, näiteks emotsioonide tuvastamiseks või uudiste rubriikide ennustamiseks, kus kogu tekst loetakse sisendiks ja väljundiks on klassi silt[4][10][11][14][16][17].

1.1.3 Transformerid

Transformerid on võrdlemisi uus arhitektuur, mis on loodud järjestuste töötlemiseks, kuid erinevalt varasematest meetoditest, näiteks korduvnärvivõrkudest (*Recurrent Neural Networks*), ei kasuta nad järjestikust töötlust. Selle asemel kasutavad transformerid tähelepanumehhanisme, mis võimaldavad neil "vaadata" kõiki sisendjärjestusi korraga ja otsustada, millisele osale järjestusest tuleb kõige rohkem tähelepanu pöörata[4][10][18].

Transformerite arhitektuur koosneb kahest peamisest osast: kooderist ja dekooderist. Kooder võtab sisendi ja teisendab selle sisemiseks esituseks. Dekooder võtab selle sisemise esituse ja teisendab selle väljundiks[8][18][19].

Kooder koosneb mitmest identsest kihist, millest igaüks koosneb kahest alamkihist: isetähelepanu kiht ja positsiooni-tundlik täisühendusega kiht. Isetähelepanu kiht

võimaldab mudelil pöörata tähelepanu erinevatele positsioonidele sisendjärjestuses, et paremini mõista konteksti. Positsiooni-tundlik täisühendusega kiht on lihtsalt täisühendusega kiht, mida rakendatakse positsiooniliselt (eraldi iga positsiooni jaoks järjestuses)[1][19].

Dekooder on sarnaselt kooderile üles ehitatud mitmest identsest kihist, kuid sisaldab lisaks kahte alamkihti: maskeeritud isetähelepanu kiht ja kooderi-tähelepanu kiht. Maskeeritud isetähelepanu kiht on sarnane isetähelepanu kihiga, kuid see ei luba "tulevikusse vaadata", see tähendab positsioonil i oleva väljundi arvutamisel ei pääse see ligi positsioonidel i+1, i+2, jne olevale sisendile. Kooderi-tähelepanu kiht võimaldab dekooderil pöörata tähelepanu kooderi väljundile[1][19].

Transformerite üks olulisemaid tunnuseid on positsiooniline kodeerimine. Kuna transformerid ei kasuta järjestikust töötlust, on vaja mehhanismi, mis aitaks mudelil mõista järjestuse järjestust. Selleks lisatakse sisendile positsiooniline kodeering, mis annab mudelile teavet iga elemendi positsiooni kohta järjestuses. Positsiooniline kodeering on reaalne väärtus, mis lisatakse sisendvektorile. See võib olla kas staatiline (sama igasuguse sisendi korral) või dünaamiline (sõltub sisendist). Klassikaline positsiooniline kodeering kasutab siinus- ja koosinusfunktsioone erinevate sagedustega[1][19].

Kõnetuvastuse puhul võimaldavad transformerid keerukate akustiliste signaalide mõistmist ja tõlkimist tekstiks. Transformerid suudavad tuvastada erinevaid kõneelemente, nagu foneeme ja sõnu, ning mõista nende vahelisi seoseid. Kõnetuvastuse rakendustes on transformerid näidanud suurepäraseid tulemusi, olles võimelised täpselt transkribeerima nii üksikisikute kõnet kui ka rühmade vestlusi[4][9][17][20].

1.2 Automaatne kõnetuvastus

Akustiline mudel ASR-süsteemis on keskne komponent, mis vastutab helisignaali kõneelementide, nagu näiteks foneemideks teisendamise eest. See ülesanne nõuab tavaliselt keeruka mitmetasemelise mudeli kasutamist, mis hõlmab mitmeid erinevaid komponente[7].

Tunnuste eraldaja (*Feature Extractor*): See on esimene samm kõne analüüsimisel. See teisendab helisignaali tunnuste kogumikuks, mis esindavad kõne olulisi omadusi. Tavaliselt hõlmab see selliseid samme nagu Fourier'i teisendus, mis muudab aja-

domeeni signaali sagedus-domeeni signaaliks, ning Mel-frekventsi tsefstraalkordajate (MFCC) või log-Mel spektrogrammide arvutamine, mis on kõne tunnuste esindused[21].

Akustilise mudeli arhitektuur: See komponent kasutab sisendfunktsioone, et ennustada kõneelementide tõenäosusi. Klassikalised akustilised mudelid kasutasid tihti Gaussi segumudeleid (*Gaussian Mixture Models*), kuid tänapäeva süsteemides on levinumad sügavad närvivõrgud. Konvolutsioonilised närvivõrgustikud on populaarsed nende võime tõttu töödelda kohalikke mustreid ja ajalisi järjestusi, mis on olulised kõnesignaali omadused. Korduvad närvivõrgustikud sealhulgas nende pikaaegse mälu variandid on samuti kasutusel, kuna need suudavad efektiivselt töödelda pikaajalisi sõltuvusi aegrea andmetes[7][8][16][17].

Õppimisalgoritm: Akustilise mudeli koolitamine nõuab tõhusat õppimisalgoritmi. Tagasilevi koos stohhastilise gradiendi langusega (*Stochastic Gradient Descent*) on standardne meetod närvivõrkude koolitamiseks. Seda tehnikat saab kasutada nii CNN, kui ka korduvates võrgustikes, nagu LSTM, koolitamiseks. Samuti võidakse kasutada optimeerimistehnikaid, nagu näiteks Adam või RMSprop, mis aitavad suurendada õppimiskiirust ja parandada õppe stabiilsust[1][18].
Keelemudel ASR-süsteemis
Keelemudel on ASR-süsteemi komponent, mis ennustab sõnade järjestuste tõenäosusi. Keelemudel aitab süsteemil mõista, millised sõnad või fraasid on kõige tõenäolisemad, arvestades eelnevaid sõnu[22].

N-gramm keelemudelid: Traditsioonilised keelemudelid tuginesid n-grammidele - sõnade järjestuste hulgale, kus "n" viitab järjestuse pikkusele. Need mudelid arvutavad tõenäosuse sõna "n" ilmumiseks, arvestades eelnevat "n-1" sõna. Kuigi need mudelid on lihtsad ja efektiivsed, ei suuda nad hästi käsitleda pikaajalisi sõltuvusi[23].

Sügavatel võrkustikel keelemudelid: Sügavate õppimismeetodite kasutuselevõtt on viinud uute keelemudelite arenguni. RNN ja transformeritel põhinevad mudelid on populaarsed, kuna need võivad modelleerida sõnade vahelisi sõltuvusi, mis ulatuvad kaugemale kui n-gramm mudelite võimekused. Näiteks BERT (Bidirectional Encoder Representations from Transformers) on transformeritel põhinev mudel, mis on saavutanud häid tulemusi mitmes keeleandmetega seotud ülesannetes, sealhulgas teksti genereerimises ja mõistmises[10][20].

1.3 Teemade modelleerimine

Teemamodelleerimine on masinõppe tehnika, mis klassifitseerib dokumente teemade järgi. See on järelevalveta õppe tüüp, mis tähendab, et see ei vaja toimimiseks märgistatud andmeid. Selle asemel õpib see andmetest mustreid. Teemamodelleerimise lõppeesmärk on automaatselt tuvastada peidetud temaatilisi struktuure dokumentide kogumis.

Üks tavalisemaid algoritme, mida teemamodelleerimisel kasutatakse, on *Latent Dirichlet Allocation* (LDA). LDA käsitleb iga dokumenti teemade kogumina teatud proportsioonis. Iga teema on sõnade jaotus, kus teemas sagedamini esinevaid sõnu antakse suurem kaal. See algoritm toimib eeldusel, et iga dokumendi sõnad aitavad kaasa dokumendi teemadele[24][25].

Top2Vec on uuem meetod, mis pakub efektiivsemat ja täpsemat viisi teemade leidmiseks dokumentide kogumis, sest selle mudeli treenimine ei eelda andmete eeltöötlemist. Selle meetodi põhiline eesmärk on luua andmetest hierarhiline esitus, kus igal tasemel on erineva üksikasjalikkusega teemad. Ülemisel tasemel tuvastab see kõige üldisemad teemad, madalamad tasemed aga jäädvustavad spetsiifilisemaid teemaisid[26].

Teemamodelleerimise rakenduse näide võib olla uudiste artiklite kogumi analüüsimine. Algoritm võib tuvastada teemasid nagu poliitika, tervis, majandus, sport ja meelelahutus. Iga artikli kohta annaks see seejärel nende teemade jaotuse.

1.3.1 LDA teemamudel

Eeltöötlus: Esmalt tuleb teha tekstiandmete eeltöötlus, et eemaldada müra ja valmistada andmed ette LDA mudelile. See hõlmab teksti puhastamist erimärkidest, kirjavahemärkidest ja numbritest. Järgmisena eemaldatakse stoppsõnad, mis on sagedamini esinevad sõnad, nagu 'on', 'ja', 'kuid', kuna need ei anna olulist panust teemade leidmisel. Seejärel jagatakse tekstiandmed üksiksõnadeks, et saada sõnastik, mis koosneb kõigist tekstis esinevatest sõnadest. Lisaks võib kasutada sõnade tüvestamist, et viia sõnad nende alusvormile ja vähendada erinevate sõnavormide mitmekesisust[24][25].

Dokumendi-sõna maatriks: Järgmine samm on dokumendi-sõna maatriksi loomine. Selleks kasutatakse *Bag of Words* (BoW) mudelit. BoW mudelis esindatakse dokumente vektoritena, kus iga sõna on esitatud oma sagedusega selles dokumendis. Dokumendi-sõna maatriksis on read vastavad dokumentidele ning veerud vastavad

sõnadele. Iga maatriksi element näitab, kui sageli konkreetne sõna esineb vastavas dokumendis[27].

LDA mudeli treenimine: Pärast dokumendi-sõna maatriksi loomist saab LDA mudelit treenida. Selleks kasutatakse dokumentide sõnaesinemiste andmeid ning LDA algoritmi, mis põhineb tõenäosuslikel meetoditel. LDA mudel otsib optimaalse jaotuse, mis kirjeldab dokumentides esinevate sõnade jaotust teemade vahel. Mudel genereerib tõenäosusjaotused, mis näitavad, kui suur on iga sõna tõenäosus kuuluda erinevatesse teemadesse[24][27].

Tulemused: LDA mudeli väljundiks on teemade loend koos nendega seotud võtmeterminitega. Iga teema esitatakse võtmeterminite kogumina, mis kirjeldavad teema sisu ja olulisust. Samuti saab iga termini kaalu/olulisust hinnata, mis võimaldab mõõta, kui oluline see termin on teema esindamisel.

1.3.2 Top2vec teemamudel

Dokumentide esitamine vektorite kujul: Top2vec alustab dokumentid esitamist vektorite kujul, kasutades Word2vec meetodit. Word2vec loob sõnade vektorid, mis näitavad sõnade tähenduslikke seoseid. Dokumentide esitamiseks kombineeritakse sõnade vektorid, et luua dokumendi vektor[26].

Dokumentide klasterdamine: Rakendatakse dokumentide klasterdamise algoritme, et sarnaste dokumentide vahel seoseid leida. Top2vec kasutab hierarhilist klasterdamist, mis võimaldab dokumente klasterdada nii üleüldisemate kui ka spetsiifilisemate teemade jaotuses, sest üks dokument võib kuuluda mitmesse klastrisse[26].

Teemade tuvastamine: Klasterdamise tulemusena leitakse teemad, millel on sarnased dokumendid. Igale teemale antakse võtmesõnad, mis kirjeldavad teema sisu. Võtmesõnad moodustatakse olulisematest ehk enim esinevatest sõnadest, mis on antud teemaga seotud. See tähendab, et teemasid saab esitada nende võtmesõnadega[26].

1.3.3 Top2Vec'i eelised võrreldes LDA-ga

Dokumendipõhine lähenemine: Top2Vec keskendub otse dokumentidele, mitte sõnadele. See võimaldab avastada teemasid, mis ei pruugi olla seotud ühegi konkreetse sõnaga, kuid mida saab siiski tuvastada dokumentide sarnasuse põhjal[24][26].

Paremad tulemused suurte andmekogude puhul: Top2Vec'i tugevus on eriti märgatav suurte tekstikorpuste(ehk dokumendi kogumikud) puhul, kus traditsiooniliste

meetodite, nagu LDA, jõudlus võib langeda. Top2Vec suudab hõlpsasti töödelda suuri andmekogusid ja leida sarnased dokumendid ning tuvastada selgelt eristuvaid teemasid[25][26].

Paindlikum klasterdamine: Top2Vec'i hierarhiline klasterdamine võimaldab dokumendid kuuluda mitmesse klasterisse. LDA meetodil on kalduvus käsitleda dokumente sõnapõhiselt, jättes tähelepanuta dokumentide sisu ja semantika. See võib viia ebapiisava teemade eristamiseni ja vähendada modelleerimise täpsust. Top2Vec meetod võimaldab aga tõhusamat ja paremat teemade eristamist, arvestades dokumentide sisu konteksti ning semantilisi sarnasusi.See annab rohkem paindlikkust teemade jaotamisel ja aitab kaasa teemade keerukuse mõistmisele[24][26].

Võimsus uute dokumentide lisamiseks: Top2Vec võimaldab hõlpsalt uute dokumentide lisamist juba loodud mudelisse, säilitades olemasolevate teemade struktuuri. See on oluline funktsioon dünaamiliste andmekogude puhul, kus dokumentide sisu võib ajas muutuda[26].

Top2Vec on tõestanud oma tõhusust mitmesugustes tekstianalüüsi ja teemade modelleerimise ülesannetes ning pakub uusi võimalusi teemade avastamiseks ja dokumentide analüüsimiseks.

1.4 Eeltreenitud mudelid ja ajatõhusus

Masinõppe valdkonnas on üks kõige ajakulukamaid aspekte mudeli nullist treenimine. See hõlmab mudeli parameetrite, nagu kaalude juhuslikku algväärtustamist ning nende kohandamist lähtuvalt veast, mida ta koolitusandmekogumil teeb - protsessi, mida sageli nimetatakse õppimiseks. Õppimisfaas nõuab suures koguses arvutusressursse ja aega, eriti keerukate ülesannete puhul, nagu automaatne kõnetuvastus või loomuliku keele töötlemine, mis nõuavad efektiivseks õppimiseks suuri andmekogumeid[4][9].

Kuid seda aja- ja ressursikulu on võimalik oluliselt vähendada, kasutades eeltreenitud mudeleid. Eeltreenitud mudelid on masinõppe mudelid, mis on juba treenitud suurte võrdlusandmekogumite peal. Need mudelid on treeningkogumilt õppinud palju teavet andmete jaotuse kohta, mis võib olla kasulik paljude seotud ülesannete jaoks[4][9].

Põhimõtteliselt on eeltreenitud mudelite taga mõte "ülekande õpe", kus ühel ülesandel treenitud mudelit kasutatakse lähtepunktina teisele, ehk sellega seotud ülesandele. Kasutades eeltreenitud mudeleid, saab ära kasutada mudeli poolt originaalse treenimisprotsessi käigus õpitud tunnuseid ja neid uue spetsiifilise ülesande jaoks kohandada või "peenhäälestada"[4][9].

See lähenemine pakub mitmeid eeliseid. Esiteks säästab see märkimisväärselt aega, kuna õppimise algne juhuslik parameetriotsingu faas jääb vahele ja alustada saab parema parameetrite komplektiga. Teiseks on uue ülesande jaoks vähem andmeid vaja, kuna mudel on juba õppinud palju kasulikke tunnuseid oma algsest koolituskomplektist.

Lõpuks toovad eeltreenitud mudelid sageli paremaid tulemusi, kuna need mudelid on juba õppinud üldisi tunnuseid suurtelt andmekogumitelt, millel nad algselt treeniti, mis sageli sisaldavad laia valikut andmeid[4][9].

1.5 Uurimistööd

Selles peatükis kirjeldatakse loomuliku keele töötlusega (NLP) seotud uurimistöid ja olemasolevaid arhitektuure. Tuuakse välja olemasolevaid lahendusi, mis ühtlasi on kaasa aidanud käesoleva uurimistöö probleemistike lahendamisele. Üldised lähenemised on seotud helide klassifitseerimisega ja kõne puhul eraldamisega.

Kõne tuvastamine ja kõneleja eristamine närvivõrkude abil üldjuhul lähtub kvaliteedist, mitte kvantiteedist[28].

Radford *et al* pakkusid välja kooder-dekooder arhitektuuri, mis tugineb närvivõrgustiku eeltreenimisele[29]. Autorite pakutud arhitektuur saavutab ligilähedasi tulemusi professionaalsete transkribeerijatega[17].

Wang ja Downey *et al* pakkusid välja kombineeritud süsteemi, mis koosneb LSTM mudeli baasil d-vektorite sisetamisega ja mitte-parameetrilisest klasterdamisest. Leides, et d-vektor süsteemid teevad vähem vigu, kui i-vektor süsteemid[11].

Spina *et al* uurisid teemade modeleerimist transkribeeritud multimeedia sisu põhjal. Autorid avastasid, et mudelit sõnaselgelt informeerides erinevate kõnelejate segmentidest aitab kaasa teemade modeleerimisele[25].

Zheng *et al* pakkus välja lahenduse riistvara näol, kasutades kõnelejate eristamiseks mitut mikrofoni[22], mis andis märgatavalt paremad tulemused kui eristamine ühe mikrofoniga.

Wang *et al* pakkusid välja lahenduse kõneleja efektiivsemaks eristamiseks mitme-etapilise klasterdamise abil sõltuvalt sisendipikkusest[28].

Lee *et al* uurisid muusika-,kõne- ja keskkonnahelide klassifitseerimist, pakkudes välja konvolutsioonilise arhitektuuri kasutades väiksemõõtmelisi filtreid ühemõõtmelise konvolutsiooniga, mida rakendati helide töötlemata lainekujule. Pakutud arhitektuur osutus üheks võimekamaks lahenduseks muusika automaatses sildistamises ja heli tuvastuses[30].

Huang ja Leanos pakkusid välja konvolutsioonilise arhitektuuri AclNet, mis saavutas ESC-50 andmestiku[31] klassifitseerimises parema tulemuse kui inimene[32].

Hussain ja Haque uurisid kas varieeruva kestusega kõne, muusika ja müra klassifitseerimine ja segmenteerimine on võimalik. Pakutud arhitektuur SwishNet töötab ühemõõtmeliste konvolutsioonide põhimõttel rakendades seda helide mel-sageduste spektri koefitsientidel[33].

1.6 Olemasolevad rakendused

Olemasolevad rakendused on suletud lähtekoodiga ja tasulised. Saadaval on osalisi uurimistöid rakenduste mõningate osade kohta, mis ei anna aimu mudelite sisemisest loogikast.

Tõenäoliselt üks tuntumaid rakendusi on Google Speech-to-Text mida on võimalik kasutada otse mikrofoni rääkides, kui ka lihtsalt faile üles laadides ja tulemusi oodates. Küll aga tundub, et Google pakub mitmeid erinevaid konfigureerimise-ja isikupärastamise võimalusi[34], seega on sellel absoluutne eelis igasuguste muude võimalike mudelite jooksutamisega võrreldes. Muidugi lahenduse piirangud määravad rangelt ära mida on selle rakendusega võimalik teha, isegi kui kasutada rakenduse programmeerimise liidest[35].

Otter.ai pakub tasuta kasutamise aega, küll aga piiratult ja ühe kuu arvestuses. Otter.ai eelis see, et neil on äpid kõigi platvormide jaoks ja üleüldiselt tundub, et

nende sihtgrupiks on firmad ja seetõttu on neil ka kallim hinnaklass võrreldes teistega[36].

Ka symbl.ai pakuvad tasuta kasutamise aega, küll tasulise variandina tundub symbl.ai odavaim variant olevat. Symbl.ai tundub olevat suunaga teha kõne ja tekstitöötlus kättesaadavamaks kõigi jaoks, võimaldades ka arendajatele rakenduse programeerimise liidese võimalusi[37][38].

Viimasematest lahendustest on saadaval ka OpenAI Whisper[39] mudel, mida kasutatakse ka baasmudelina antud uurimistöö probleemistike lahendamisel. Whisper'i mudelit saavad kõik alla laadida, aga vajalik juhtimise kood on siiski vaja kasutajatel endal kirjutada, et Whisper mudelit kasutada[39]. Whisper mudeli praegused puudused on selles, et kuna mudelit treeniti andmestikuga mille sisendandmed oli 30 sekundi pikkused helifailid, siis ka mudel on ehitatud töötama sellise 30 sekundilise puhverajaga. Vaatamata sellele on Whisper mudel antud uurimistöö baasmudeliks, et üritada Whisper mudel saada töötama natuke lähemale reaalajale.

Eksisteerib C++ keele põhine implementatsioon Whisperi Pythoni mudelist, millel on isegi algeline GPU toetus. Kuna C++ on võimekam ja enamjaolt kiirem keel, siis tundub, et see lahendus võimaldab mahuliselt mudeleid isegi telefonide (mainitud iPhone 13) peal jooksutada. Ühtlasi tundub olevat teatud võimekus reaalajas töötluseks, aga see eeldab endiselt audio eeltöötlust ja mudeli kvantimist. Antud raamistikul puudub võimekus teemade mudeldamise jaoks[40].

Mozilla DeepSpeech[41] ASR raamistik mida arendati aktiivselt 2017 kuni 2020, aga tundub, et projekt on hüljatud. Tundub, et antud mudel oli reaalaja eristuse võimeline, aga autor ei saanud seda testida. Projekti käima saamiseks peaks kulutama aega projekti uuendamisele, sest raamistik on vananenud ja autor peaks seda saadavaloleva riistvara jaoks sobitama hakkama.

1.7 Järeldused

Erinevate rakenduste uurimisel leidis autor mitmeid olulisi ja üldisi omadusi, mida meie soovitud rakendusel peab olema. Kõigepealt peab see olema võimeline töötlema suures koguses heliandmeid, teiseks tuvastama erinevaid kõnelejaid, kolmandaks transkribeerima kõne tekstiks ja lõpuks jagama transkriptsiooni teemalõikudeks.

On olemas Whisper mudeli C++-põhine raamistik, mis on loodud selleks, et töödelda heliandmeid kiiremini kui enamik Pythoni põhiseid raamistikke. C++ on tuntud oma jõudluse poolest, mis muudab selle ideaalseks keeleks suuremahuliste andmete töötlemisel. Kuid see raamistik keskendub peamiselt heliandmete töötlemisele ja ei paku laiaulatuslikke kõne tuvastamise ja transkribeerimise funktsioone[40].

Peale selle on mitmeid avatud lähtekoodiga projekte, mis on loodud kõnetuvastuseks. Näiteks, "Mozilla DeepSpeech" projekt kasutab sügavõppe mudeleid, nagu konvolutsioonilised neuronaalsed võrgud (*Convolutional Neural Networks*) ja korduvaid närvivõrke, kõne tuvastamiseks ja transkribeerimiseks[41]. Kuid see projekt ei keskendu spetsiifiliselt teemade tuvastamisele transkriptsioonis, küll aga tundub mudelil olevat võimekus reaalaja eristusele, see muidugi tähendaks ajakulu raamistiku ja mudeli uuendamisele, mis ei tundu mõistlik.

Kui rääkida teemade modelleerimisest, siis "Latent Dirichlet Allocation" (LDA) on tuntud masinõppe mudel, mida sageli kasutatakse teemalõikude eraldamiseks tekstis. Näiteks, Githubis leiduvad projektid nagu "gensim"[42] pakuvad vahendeid LDA kasutamiseks Pythonis. Kuid nende projektide puhul puudub sidusus kõnetuvastuse ja transkribeerimisega.

Eelneva põhjal võib teha järelduse: kuigi on olemas mitmeid tõhusaid raamistikke ja projekte, mis võimaldavad kõnetuvastust ja teemamodelleerimist, ei ole leidnud ühtegi rakendust, mis kombineeriks kõik soovitud omadused üheks tervikuks. See avastus toetab vajadust luua uus rakendus, mis suudaks ühendada need kõik.

Seetõttu on saadaval olevatest vahenditest kõige mõistlikum rakendada Whisper mudelit, sest mudel on ajakohane, ei vaja ümbertreenimist ja mudeli kaalud on avalikud. Teemade mudeldamises ei leidu eriti palju vabas kasutuses olevaid rakendusi või teeke peale LDA põhineva "gensim"[42] teegi, mis võimaldab LDA hõlpsamat rakendamist.

2 KAVANDATAV RAKENDUS

Töö eesmärgiks oli luua kõneeristuse arhitektuur, mis suudab kõnest tekitada tekstilise väljundi, mille erilisus on võime teha tööd ligilähedal reaalajale. Ühtlasi suudab mudel multimeedia sisu indekseerida vastavalt sisu temaatika järgi.

2.1 Arvuti spetsifikatsioon

Testimise jaoks kasutatud spetsifikatsioon on toodud selle jaoks, et oleks võimalik hinnata erinevate riistvarade suhtelist potentsiaali rakendades sarnaseid meetodeid nagu antud töös on välja toodud.

Töö käigus kasutati arvutisüsteemi, millel on järgmised tehnilised andmed:

- Protsessor: AMD Ryzen 5800X. Sellel on kaheksa tuuma, kuusteist lõimu ja need jooksevad kiirusega kuni 4.7 GHz[43].
- Graafikakaart: NVIDIA RTX 3070. Sellel graafikakaardil on 5888 tuuma, millest igaüks jookseb kuni 1.73 GHz ja kaardil endal on ka 8 GB mälu[44].

Kõneleja tuvastamise ja kõnetuvastuse optimeerimiseks võib olla vajalik kasutada spetsiifilisi teeke ja tööriistu, mis on loodud töötama NVIDIA riistvaraga. Eriti mis puudutab CUDA ja PyTorch põhiseid teeke ja raamistikke.

2.2 Eeltreenitud mudel

Autor otsustas kasutada eeltreenitud mudelit Whisper[39], et vähendada süsteemi jaoks arvutuslikku keerukust ja hoida aega kokku selle arvelt, et mudelit ei pea nullist üles töötama. Autori otsus valida Whisper teiste mudelite, näiteks Mozilla DeepSpeech üle, tugines mitmetele kaalutlustele. Kõigepealt, arengud masinõppe valdkonnas on kiired ja uued tehnoloogiad võivad kiiresti vananeda. Kuigi Mozilla DeepSpeech oli oma aja tipptehnoloogia ei ole see tänapäevaste lahendustega võrreldav.

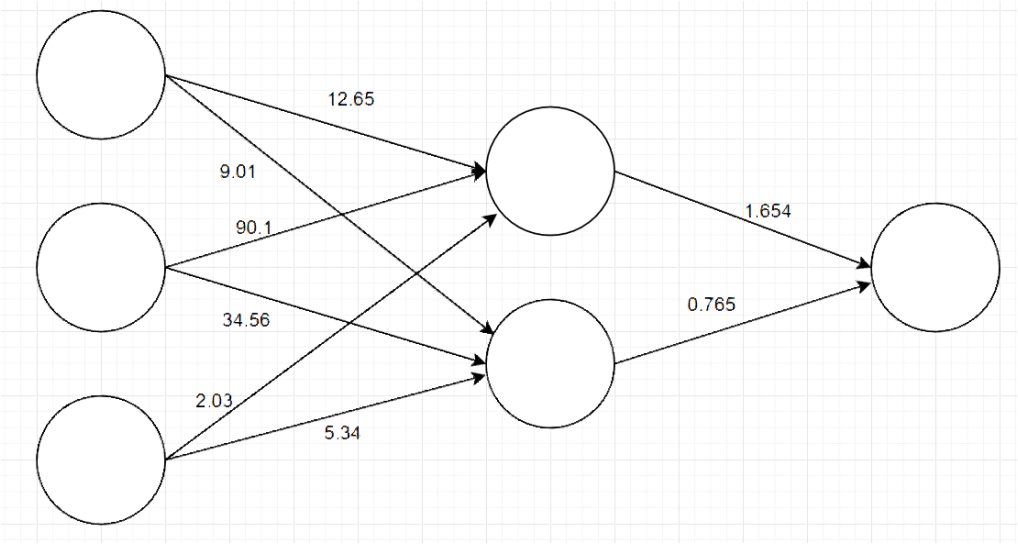
Samuti oli oluline, et Whisper on ette koolitatud suurel hulgal andmetel, mis võimaldas mudelil paremini tuvastada ja tõlgendada kõnes sisalduvat informatsiooni. DeepSpeech oleks vajanud palju rohkem treeningandmeid ja ressursse, et saavutada sama taset, mis Whisperil oli juba saavutatud. See viis järgmise punktini, mis on see, et teiste rakenduste mudelid ei ole saadaval ja nõuaksid suuri koguseid treeningandmeid.

Whisper on eeltreenitud mudel, mis on treenitud erinevate keelte ja kokku umbes 680 000 tunni sisendandmestiku peal, mistõttu on see võimeline käsitsema paljusid kõnetuvastuse-ja eristamise aspekte. Selle eeltreenitud mudeli kasutamisega saab kõne analüüsi rakenduse oluliselt kiiremini ja tõhusamalt välja töötada.

Lisaks võib eeltreenitud mudelite kasutamine vähendada ka vajaliku arvutusvõimsuse nõudeid. Kuna nende mudelite treenimine on juba tehtud, vajavad need uue ülesande jaoks ainult peenhäälestamist, mis tavaliselt nõuab vähem arvutusressursse kui nullist treenimine.

2.3 Mudeli kvantimine

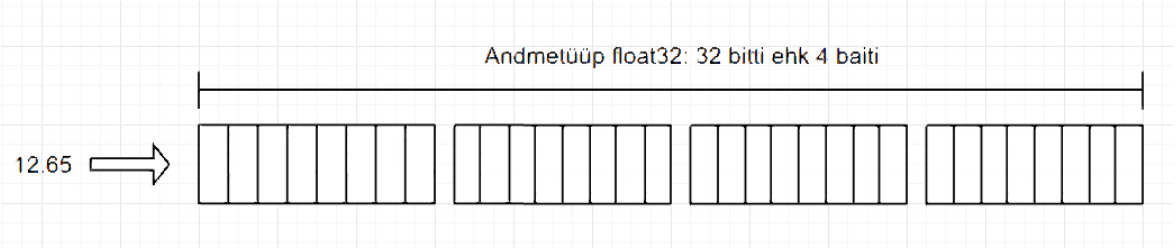
Kvantimine on protsess, mida kasutatakse masinõppe mudelite numbriliste esituste täpsuse vähendamiseks, mis tegelikult neid kompresseerib. Kõige lihtsamal kujul hõlmab see arvu esitava bittide arvulist vähendamist.



Joonis 2.1 Skeem närvivõrgustikust koos neuronite ja kaaludega.

Joonisel 2.1 on skeem lihtsamast närvivõrgustikust, milles kasutatakse ujukomaarve, et algoritmidel oleks võimalik saavutada täpsemaid tulemusi, mida täisarvud ei võimaldaks.

Näiteks võib 32-bitise ujupunktarv kvantida 8-bitiseks täisarvuks. Eesmärk oli vähendada mudeli arvutus- ja mälunõudeid, võimaldades kasutuselevõttu piiratud ressurssidega seadmetes või kiirendades arvutusi suure jõudlusega riistvaral.

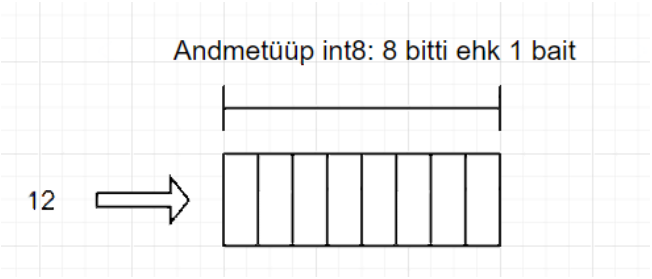


Joonis 2.2 Float32 andmetüüp.

Joonisel 2.2 on kujutatud float32 andmetüüpi, mida on võimalik arvuti mälus kujutada kui 32 järjestikust kastikest. Iga kastike saab olla binaarsüsteemi järgi kas 1 või 0.

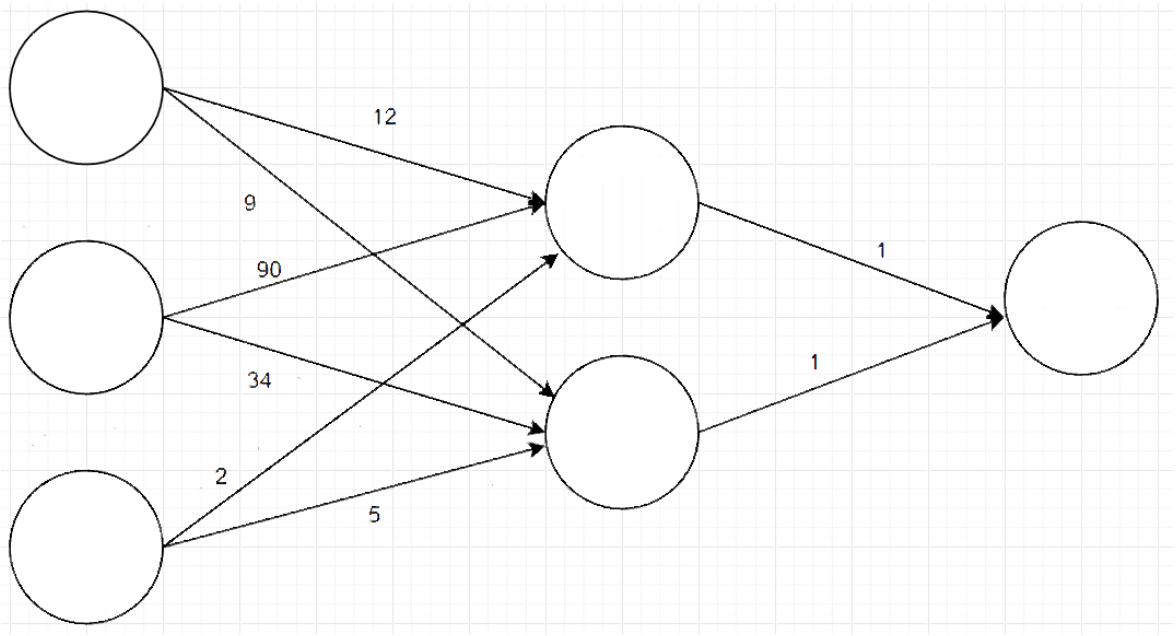
Float32 andmetüüp kasutab esimest bitti, et näidata kas arv on positiivne või negatiivne, järgnevad 8 bitti kasutatakse tüvearvu näitamiseks ja ülejäänud 23 bitti on komakoha täpsuse jaoks.

Arvutamisel on enamjaolt oluline täpsus, mida võimaldab float32, aga mõnikord nende arvude teisendamine ei tähenda ilmtingimata suuri kadusid närvivõrgustike täpsuses. Seega on võimalik teisendada float32 ümber vähemamahulisteks andmetüüpideks, mis ühtlasi nõuavad arvutuslikult vähem ressursse, näiteks int8[17][45][46].



Joonis 2.3 Int8 andmetüüp.

Joonisel 2.3 on int8 andmetüüp, millel on kõigest 8 bitti, ehk 4 korda vähem bitte kui Joonisel 2.2 nähtud float32 andmetüübil. Nähtub, et seda sama arvu 12 on võimalik esitada arvuti jaoks 4 korda väiksemal kujul ja olenevalt arvuti arhitektuurist võib see tähendada oluliselt kiiremat protsessori tööd[9][45][46].



Joonis 2.4 Närvivõrgustik kvanditud kujul.

Joonisel 2.1 kujutati lihtsamat närvivõrgustikku koos neuronite ja kaaludega, mida on võimalik esitada kvanditud kujul nagu on näha Joonisel 2.4. Skeemid on mõeldud olema illustreerivad ja ei näita detailselt ühtegi kindlat kvantimise meetodit, sest kasutusel on erinevaid kvantimise tehnikaid, millede kasutamine sõltub nii närvivõrkude arhitektuurist kui ka eesmärgist.

Tänapäevaste NLP ja ASR arhitektuuride mudelid on näidanud, et kvantimisel on positiivne mõju mudelite efektiivsusel, vähendades nii treenimise mahukust kui ka ajakulu. Ühtlasi on täheldatud positiivset mõju mudelite tuletamisvõimes[9].
Selle põhjal võib eeldada, et Joonisel 2.4 kujutatud kvanditud mudel suudab teha efektiivsemat tööd kui Joonisel 2.1 kujutatud närvivõrk.

Siiski on oluline tasakaal, vähendatud ressursinõuete ja mõju vahel mudeli sooritusele. Liigne kvantimine võib põhjustada olulist mudeli täpsuse langust, seetõttu on oluline mudelit testida enne ja pärast kvantimist.

Kvantimine on põhjendatud mudelite kasutuselevõtu kontekstis reaalajas või peaaegu-reaalajas rakendustes. Sellistel juhtudel muutuvad kvanditud mudelid vähenenud arvutusnõuete tõttu eriti kasulikuks, sest see võimaldab vähendada riistvaralisi nõudeid, või vastupidi jooksutada mudeleid kiiremini võimekama riistvara peal.

2.4 Paralleliseerimine

Paralleliseerimine on meetod, mida kasutatakse arvutiprogrammide kiirendamiseks, jagades arvutusülesanded mitme protsessi vahel, mis töötavad samaaegselt. See võimaldab töötlemist jagada mitmele arvutusressursile, näiteks protsessoritele või tuumadele, mis kiirendab ülesande täitmist ja suurendab üldist tõhusust.

Whisper mudeli puhul võib paralleelprotsesside kasutamine olla eriti oluline, kui mudelit tuleb reaalajas rakendada. Mudeli võime töödelda suuri andmevooge kiiresti ja tõhusalt võib mõjutada selle kasulikkust mitmetes reaalajas rakendustes, näiteks reaalajas kõnetuvastuses või kõne transkribeerimises.

Samuti aitab protsesside paralleliseerimine vähendada viivitusi, mis on oluline reaalajasüsteemides. Paralleelne arvutus ei pruugi olla alati võimalik või kasulik, kuid Whisper mudeli puhul võib see olla märkimisväärne eelis.

Protsesside paralleliseerimisega aitas leitud Whisperil põhinev C++ raamistik[40][47], mis peaks võimaldama süsteemi paralleliseerimist ja andmetorustike kiirendamist.

2.5 Teemamudel

Töö käigus leiti, et Top2Vec on antud töö kontekstis parem lahendus kui LDA, sest Top2vec kasutab word2vec meetodit, et luua sõnade vektoreid, mis näitavad sõnade semantilisi seoseid. Dokumendi esitamiseks kombineeritakse sõnade vektorid, et luua dokumendi vektor. Dokumendi vektor peegeldab dokumendi sisu ja sõnade semantilist konteksti.

Word2Vec on populaarne meetod sõnade esitamiseks vektorite kujul, kus sõnad esindatakse numbrite jadadena, mida on lihtne matemaatiliselt töödelda. Word2Vec meetod põhineb sõnade semantilistel seostel ja nende konteksti põhisel esinemisel tekstis. Selle tulemusena luuakse sõnade vektorruum, kus sarnased sõnad on lähedal üksteisele ja erinevad sõnad on kaugemal.

Pärast dokumentide esitamist vektorite kujul rakendatakse klasterdamisalgoritmi, et leida sarnased dokumendid. Klasterdamise eesmärk on grupeerida dokumendid, mis jagavad sarnast teemat või sisu. Top2Vec kasutab hierarhilist klasterdamist, mis võimaldab dokumendid kuuluda mitmesse klasterisse. See tähendab, et dokumendid, mis võivad hõlmata mitut teemat, paigutatakse vastavalt klastritesse mis on

üksteisele lähemal. Hierarhiline klasterdamine annab paindlikkuse teemade jaotamisel ning võimaldab mõista teemade keerukust ja seoseid.

Klasterdamise tulemusena tuvastatakse dokumendid, milles esinevad sarnased teemad. Iga teema saab oma võtmesõnad, mis kirjeldavad teema sisu. Need võtmesõnad moodustatakse kõige olulisematest sõnadest, mis on seotud antud teemaga. Võtmesõnad esindavad teemasid ja võimaldavad nende paremat mõistmist.

Top2Vec mudel pakub mitmeid eeliseid võrreldes traditsioonilise LDA-ga, sealhulgas võime paremini töödelda suuri andmekogusid, dokumendipõhine lähenemine, paindlikum klasterdamine ning võime hõlpsalt lisada uusi dokumente juba loodud mudelisse. Sõnade esitamine Word2Vec meetodi abil aitab luua tähendusrikkaid vektoridokumente, mis omakorda võimaldab Top2Vec'il tuvastada teemasid ja klastreid[26][48].

3 SÜSTEEMI KOKKUPANEMINE

Mudelite ja süsteemi testimise jaoks otsustas autor kasutada YouTube'st võetud kanali „vlogbrothers" sisu, millest autori poolt selekteeritud 3 videolõiku.

Kasutusel olid nimetatud kanali poolt järgnevad videolõigud:

- „Do I have ADHD"[49] edaspidi nimetatud VIDEOLÕIK 1
- „The Coming AI Invasion"[50] edaspidi nimetatud VIDEOLÕIK 2
- „AI tells me if AI will write novels"[51] edapidi nimetatud VIDEOLÕIK 3

Videoklipid valiti eeldusega, et kahel neist võiks olla seoseid rohkem AI süsteemidega, erinedes teistest teemadest ja moodustades ühe klastri andmete indekseerimise süsteemis, selle tulemused hiljem analüüsis.

Videolõikude täpsete ehk kontrolltranskriptsioonide saamiseks kasutas autor Whisper "medium.en" mudelit, et saavutada maksimaalne täpsus videolõikude transkribeerimisel, seejärel manuaalselt tekst üle kontrollides samal ajal videolõike kuulates ja analüüsides.

Manuaalselt kontrollitud Whisper mudelit võrreldi reaalajasüsteemi tulemustega iga videolõigu kohta, seda nii kvanditud kui ka kvantimata mudeli puhul, et näha millist mõju see mudeli täpsusele avaldab.

Hindamiskriteeriumiteks võeti valdkonna standardina sõnaveamäär ehk *Word Error Rate* ja selleks kasutatakse vabalt saadaval olevat tööriista „Amberscript"[52].

3.1 Testimise keskkond

Selles peatükis kirjeldatakse ja antakse ülevaade reaalaja mudeli testimise keskkonnast.

Arvuti külge oli kinnitatud Logitech C525 veebikaamera, mille mikrofoni kasutatakse sisendina. Mikrofoni poole kõlariga oli suunatud telefon iPhone 13, mis esitab YouTube'st videolõike. Seadmed on üksteise suhtes vastakuti üksteisest 5 cm kaugusel.

3.2 Teegid ja tööriistad

Enne mudeli kasutamist oli vajalik installeerida vajalikud abistavad tööriistad.

Ffmpeg[53] on tööriist, mis on vajalik audio eeltöötlemise jaoks, see on vajalik, et audio oleks mudelile õiges ja loetavas formaadis. Ühtlasi, kuna Whisper mudel nõuab, et heli diskreetimissagedus oleks 16 kHz siis selle jaoks saabki kasutada ffmpeg tööriista.

SpeechRecognition[54] on teek, mida kasutatakse kõnetuvastuse toestamiseks. Antud töö kontekstis kasutatakse seda teeki audioandmete salvestamiseks, struktureerimiseks ja eeltöötluseks.

PyTorch[46] on masinõppe raamistik, mida kasutatakse antud süsteemis, et rakendada sügava õppe algoritme. See võimaldab kasutada kiirendust läbi graafikatöötlusüksuste (GPU), mis on oluline keerukate masinõppe mudelite, nagu kõnetuvastusmudelite, efektiivseks treenimiseks ja rakendamiseks. PyTorchi tensori arvutuste funktsioonid võimaldavad töötada suuremate andmemahtudega, mida tavaliselt seostatakse kõneandmetega.

CTranslate2[47] on C++ ja Pythoni teek, mis võimaldab tõhusamat järeldusvõimekust Transformer mudelitega ja ühtlasi toetab ka Whisper mudeleid.

3.3 Whisperi mudelid

Whisper mudel tuleb viies erinevas suuruses, see tähendab, et mudeli treenimisel viies erinevas etapis talletati mudeli tulemused, mis tähendab, et väiksemat mudelit on ajaliselt kõige vähem treenitud ja suuremat mudelit kõige rohkem. Suurema mudeli jaoks on riistvaralised nõuded suuremad ja ühtlasi on ka mudeli töö aeglasem, küll aga täpsem. Väiksem mudel töötab suurema mudeliga võrreldes kiiremini, aga mitte nii täpselt[17][39].

Igast mudelist on ka inglise keelne ja mitmekeelne versioon, milest Whisperi autorite endi sõnul peaks olema täpsem inglise keelne mudel, kui mudeli ülesandeks on töödelda inglise keelseid andmeid. Ehk kehtib reegel, kui mudelile teada anda millise keelega tegemist tuleb, siis peab mudel vähem ressurssi kulutama analüüsimisele millise keelega parasjagu tegemist on[17].

Testimise jaoks kasutati videolõike, mis on inglise keeles siis selle puhul peaks .en-lõpuga mudel täpsemaid tulemusi andma hoides ressurssi kokku keele dünaamiliselt

tuvastamise arvelt. Eelneva põhjal otsustas autor väiksema ja ühtlasi teistest mudelistest suhteliselt kiireima mudeli kasuks "tiny.en".

Tabel 3.1 Whisper'i erinevad mudelid[55].

Mudeli suurus	Ingl. k mudel	Parameetrid	GPU VRAM nõuded	Suhteline kiirus
tiny	tiny.en	39 M	~1 GB	~32x
base	base.en	74 M	~1 GB	~16x
small	small.en	244 M	~2 GB	~6x
medium	medium.en	769 M	~5 GB	~2x
large	Puudub	1550 M	~10 GB	1x

3.4 Mudelite kvantimine

Videolõike kasutati, et saada "tiny.en" mudeli baas transkriptsioonid, ilma kvantimiseta ja seejärel peale kvantimist, et selle järgi otsustada millist mudelit on mõistlikum kasutada reaalaja mudeli jaoks. Mudeli valik kaldus väiksemate mudelite poole selle tõttu, et need peavad vähem tööd tegema järelduste tegemiseks, aga siiski jäi küsimus kas kiirem tulemus oleks ka kvaliteetsem tulemus.

Kuna Whisper on eeltreenitud mudel, jääb antud uurimistöö kontekstis mudeli parameetreid mõjutada peale treenimist. Mudelite kvantimise jaoks kasutatakse PyTorch teeki, et kasutada staatilise (PTSQ) ja dünaamilise kvantimise (PTDQ) meetodeid[56][57].

Whisper mudeli kvantimise jaoks tuli kõigepealt importida vajalikud Pythoni moodulid, nagu torch ja torch.quantization. Seejärel sai kasutada funktsiooni torch.quantization.quantize_dynamic(), et dünaamiliselt kvantida mudelit. See funktsioon võimaldas spetsifitseerida, millised kihtide tüübid kvantida (nt *nn.Linear*), ja milliseks andmetüübiks (Int8, int16 jne) kvantida.

Whisper mudeli puhul sai kvantida Whisperi lineaarselt ühendatud kihid, mis muidu kasutaksid täpsuse jaoks 32-bitist ujukomaarvu.
Kvantimist sai teha järgneva käsuga:
torch.quantization.quantize_dynamic(model_fp32,{torch.nn.Linear},type=torch.qint8)
, mis dünaamiliselt teisendas 32-bitised ujupunktarvud ümber 8-bitiseks

täisarvuks[57]. Seega torch teegiga oli võimalik konverteerida lineaarsete kihtide ujupunktarvud väiksemateks täisarvudeks.

3.5 Whisper reaalajas

Olenemata leitud C++ alternatiividele otsustas autor siiski Pythoni kasuks, sest Whisperi Pythoni API-ga on võimalik süsteem kiiremini kokku panna.

Autor proovis ka CTranslate2 teeki, aga piisavalt sorava teksti korral tundus, et puhver aina täitus ja tekst läks kaduma, olenemata kõne lindistamise parameetrite muutmisest. Võimalik, et autor tegi seadistamisel vea, aga otsustas edasi minna isiklikult kirjutatud Pythoni skriptiga, milles kasutati Ffmpeg, PyTorch ja SpeechRecognition teeke.

Antud töö kontekstis kasutati paralleliseerimist Whisperi mudeli jooksutamisel, et parandada kiirust ja töökindlust. Whisperi mudel töötleb heliandmeid, kuid see töötab vaikimisi ainult 30-sekundiliste helisegmentidega. Selleks, et transkriptsiooni oleks võimalik teha varieeruva pikkusega helisegmentidega, pidi heli töötlema osade kaupa.

Paralleeliseerimine töötas järgmiselt: kui helisegment oli salvestatud, lisati see järjestikuste helisegmentide järjekorda. Järgmine helisegment lisati eelnevatele, moodustades järjest pikema helipuhvri. Seejärel töötles Whisper mudel seda pikka helipuhvrit, mis sisaldab kõiki järjestikuseid helisegmente, ning tagastas transkriptsiooni tulemuse. Protsess kordus, kui järgmine helisegment salvestati ja lisati puhvrile. Whisperi mudeli puhul jaotus paralleliseerimine kahte protsessi, millest põhiline oli taustal kuulamine ja pidev kõne salvestus, samal ajal teine kõrvalprotsess salvestusi töödeldes.

1. Taustal kuulamine ja andmete salvestamine

   SpeechRecognition teeki kasutati, et tekitada eraldi paralleelne protsess taustal kuulamise jaoks, mis võttis sisendiks parameetrina liikuva akna pikkuse, ehk antud rakenduses iga 2 sekundi järel edastatakse digitaalse heli andmed töötlemata kujul Whisper'i paralleelselt jooksvale protsessile.

2. Taustal kuulava protsessi jaoks seadistati ka kõnetuvastuse lävend, mis määrab, et teatud heli tasemest allapoole mikrofon ei lindista heli, toimib kui kõrgpääsfilter helile. Eesmärk oli eemaldada liigne müra ja ühtlasi tuvastada hetked reaalajatuvastamise jaoks, kui kõne ei eksisteeri. Dokumentatsioonis soovitatakse kasutada väärtuseid vahemikus 50 kuni 4000, sest ei eksisteeri ühte õiget piiri, mis toimiks erinevate riistvara ja helide kombinatsiooniga. Antud

uurimistöös toimis väärtus 1200, kus mikrofon ei korjanud üles taustamüra, aga endiselt reageeris telefoni kõlarist tulnud videoklippide kõnele[54].

3. Rakendusele anti ka teine parameeter, mis määras kunas lause lugeda lõpetatuks, antud rakenduse puhul toimis väärtusena 3 sekundit. See määratakse eelnevalt mainitud liikuva akna põhjal, mis iga 2 sekundi tagant heli andmeid edastab. Lisaparameetri eesmärk nähtub sellest, et kui heli andmete puhver on tühi ja seda 3 sekundit järjest(ehk miskit pole edastatud heli puudumise tõttu), siis tehakse eeldus, et transkriptsioonis peaks tulema järgmine rida, kuna mõte(või lause) sai läbi.

4. Andmete töötlemine

   Eraldatuna eelnevast protsessist jooksis andmete töötlemine, kus töödeldi puhvrisse talletatud heli andmeid. Kui tuvastati, et heliandmete puhver ei olnud tühi siis see tähendas, et andmed olid saadaval töötlemiseks. Need andmed edastati Whisper mudelile transkribeerimiseks.

Paralleelsuse olulisus seisnes selles, et heli andmeid töödeldi pidevalt ja samal ajal, kui uued heli andmed saabusid. See tagas reaalajas kõnetuvastuse funktsionaalsuse ja võimaldas pidevat teksti transkribeerimist ilma pikema viivituseta.

3.6 Top2vec teemamudel

Kuna teemamudelit on keeruline tekitada ainult üksikute dokumentide puhul, siis ei saadud süsteemi ehitada kolme videolõigu transkriptsiooni abil, millega testiti reaalaja mudelit.

Reaalaja mudelit rakendati „vlogbrothers" YouTube kanalilt vabalt valitud 26 videoklipi peal ja tulemused iga videolõigu kohta talletati teksti kujul. Valitud videomaterjal on nähtav autori tekitatud esitusloendis[58]. Kuna töö käigus selgus, et parima soorituse tegi kvanditud mudel, siis otsustas autor kasutada 26 videolõigu reaalajas transkribeerimiseks „tiny.en" kvanditud mudelit.

Järgmisena laeti dokumendid top2vec mudelisse, koos nende laadimisjärjekorra indeksiga. Treenimisel oli saadaval kolm eri muutujat treenimise kiiruse jaoks, millest antud töös otsustas autor kasutada keskmist, ehk *learn* kiirust. Treenimise tulemusena salvestati mudel, mis oli treenitud reaalaja kõnetuvastuse andmete peal. Peale treenimist sai salvestatud mudeli laadida mälusse, et mudeli seoseid ja järeldusi analüüsima hakata.

Järgmisena uuriti saadud mudelilt teemadega seotud statistikat, et paremini mõista teemade ja dokumentide arvu. Funktsioon „get_num_topics" andis meile ülevaate teemade koguarvust, samas kui funktsioon „get_topic_sizes" tagastas teemade suurused ja numbrid.

```
[337 313 285 271 270]
[0 1 2 3 4]
```

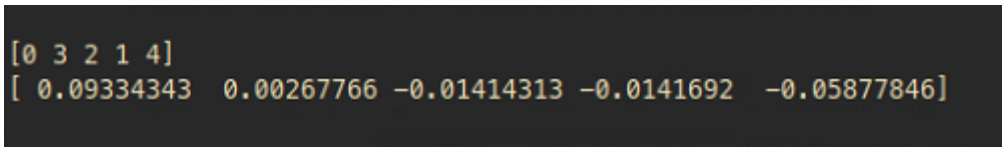
Joonis 3.1 teemade suurused ja teemade arv, indeksitena.

Edasi otsiti konkreetseid teemasid, kasutades funktsiooni „get_topics", mis tagastab kõik leitud ja talletatud sõnad ja millistesse erinevatesse gruppidesse nad kuuluvad. Antud eksperimendi käigus tuvastas top2vec viis teemat.

```
[['like' 'me' 'what' 'my' 'the' 'john' 'about' 'it' 'just' 'we' 'very'
  've' 'don' 'by' 'have' 'now' 'to' 'how' 'if' 'was' 'would' 'so' 're'
  'which' 'as' 'that' 'not' 'be' 'more' 'people' 'of' 'because' 'all'
  'but' 'on' 'one' 'or' 'and' 'in' 'know' 'think' 'been' 'they' 'when'
  'also' 'out' 'there' 'good' 'with' 'is']
 ['would' 'also' 'of' 'we' 'about' 'be' 'just' 'or' 'people' 'like' 'so'
  'if' 'you' 'when' 'was' 'there' 'how' 'which' 'in' 'think' 'at' 'all'
  'they' 'because' 'this' 'and' 'know' 'out' 'who' 'have' 'good' 'do'
  'with' 'my' 'don' 'one' 'as' 'the' 'very' 'john' 'not' 've' 'to' 'for'
  'on' 'can' 'that' 'been' 'is' 'by']
 ['by' 'good' 'how' 'been' 'you' 'not' 'if' 'was' 'the' 'would' 'can'
  'are' 'just' 'do' 'all' 'to' 'on' 'one' 'but' 're' 'that' 'know' 'also'
  'which' 've' 'more' 'they' 'don' 'who' 'at' 'we' 'now' 'very' 'when'
  'john' 'so' 'is' 'about' 'with' 'people' 'get' 'for' 'and' 'this' 'be'
  'there' 'or' 'out' 'in' 'my']
 ['been' 're' 'this' 'how' 'if' 'but' 'there' 'people' 'because' 'just'
  'when' 'with' 've' 'also' 'at' 'not' 'in' 'or' 'we' 'out' 'like' 'by'
  'more' 'very' 'know' 'one' 'john' 'can' 'about' 'was' 'be' 'get' 'don'
  'have' 'now' 'think' 'so' 'what' 'my' 'they' 'it' 'to' 'who' 'would'
  'that' 'you' 'are' 'on' 'of' 'good']
 ['all' 'about' 'this' 'is' 'for' 'don' 'that' 'at' 'me' 're' 'but' 'was'
  'get' 'have' 'good' 'like' 'out' 'one' 've' 'in' 'how' 'also' 'they'
  'know' 'people' 'not' 'when' 'very' 'more' 'there' 'with' 'can' 'if'
  'as' 'who' 'so' 'been' 'my' 'do' 'what' 'the' 'or' 'it' 'we' 'because'
  'by' 'be' 'and' 'on' 'think']]
```

Joonis 3.2 Sõnad mida mudel pidas olulisemateks.

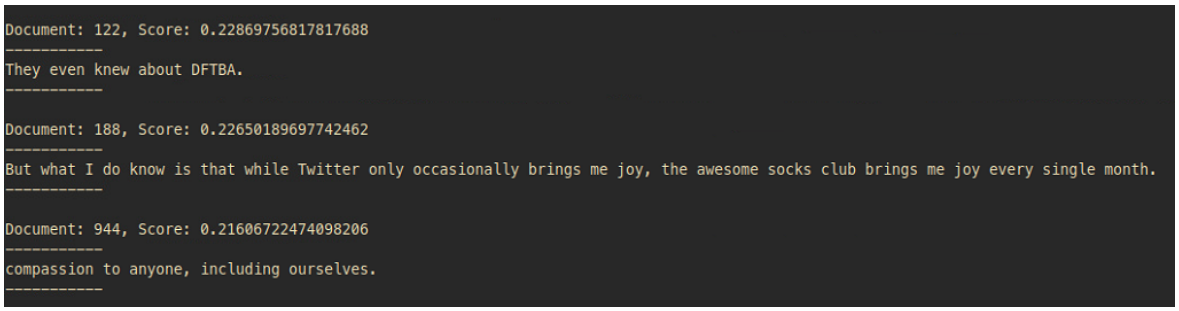
Teemade otsimiseks võeti kasutusele ka võimalus otsida teemasid konkreetsete võtmesõnade alusel, kasutades funktsiooni „search_topics", andes võtmesõnaks "john". Võtmesõna valiti sellepärast, et antud YouTube kanali esindajal on kombeks oma vaatajate poole selle hüüdlausega pöörduda, seega see on turvaline valik teades, et see tekstis eksisteerib. Selle tulemusena saadi teemadega seotud numbrid ja skoorid, mis olid seotud antud võtmesõnaga.



Joonis 3.3 Sõna "john" seonduvus teemadega.

Joonisel 3.3 nähtus, et sõna "john" seos oli kõige tugevam esimese (indeks 0) teemaga. Edasi uurimise jaoks sai kasutada teemat 0, et otsida teisi sarnaseid dokumente, mis seostuksid selle teemaga.

Funktsioon „search_documents_by_topic" võimaldas leida dokumente, mis olid seotud teema numbriga, näiteks teema number 0. Saime vastavate dokumentide teksti, skoorid ja ID-d.



Joonis 3.4 Teema indeksiga 0 seonduvad järgmised dokumendid.

Selgus, et sõna "john" oli teemamudeli poolt tuvastatud kui üks semantiliselt oluline sõna, millel oli antud dokumendi kogumikus kõige tugevam seos ka esimese teemaga. Teades, millise teemaga sõna "john" kõige tugevamat seost omab, sai otsida teema järgi teistest dokumentidest taolisi seoseid.

4 ANALÜÜS

4.1 Whisper mudeli kvantimise tulemused

Whisper mudeli "tiny.en" transkriptsiooni võrdlus kvantimisega ja ilma, rakendades eelpool nimetatud „vlogbrothers“ kanali videolõikudel.

Kvanditud mudelite testimise eesmärk oli leida, kas reaalaja süsteemi rakenduses on mõistlik kasutada kvanditud mudeleid või mitte, kuna oli risk, et väiksemaid mudeleid kasutades läheb informatsioon kaduma ja mudelid muutuvad ebatäpseks.

Tabel 4.1 Kvantimise mõju "tiny.en" mudelite suurustele.

Mudel	Suurus (megabaitides)
tiny.en	151
tiny.en(kvanditud)	101

Tabeli 4.1 tulemuste põhjal nähtus, et kvantimisel oli mõju isegi väiksemale mudelile, aidates vähendada mudeli suurust ja vähendades arvutuste täpsust teatud aritmeetiliste piirideni, mis antud töö puhul tähendas float32 andmetüübi teisendust int8-ks. See teisendus võimaldas neid arve tõhusamalt laadida ja töödelda mälus, mis omakorda tähendas, et kvanditud arvudele pidi protsessor kulutama vähem tsükleid. Küll tasub tõdeda, et protsessorid on erinevate arhitektuuridega ja mõningatel neist on eraldi arvutuslik üksus ka float32 tüüpi arvude kiiremaks töötlemiseks.

4.2 Reaalaja süsteemi analüüs

Mõõtmised Whisper "tiny.en" mudeliga viidi läbi kasutades kolme videoklippi YouTube'ist. Videoklipid valiti mitmekesisuse tagamiseks, et hinnata mudeli toimimist erinevate kõne stiilide, aktsentide ja taustamüra tingimustes. Kõikide videoklippide heli sisu salvestati ja edastati Whisper tuum mudelile transkribeerimiseks. Sõnaveamäära puhul arvestati kolme põhilist kriteeriumit ja loetakse vea alla, kui:

1. Ennustatud tulemustes olid liigsed sõnad, mida tegelikus tekstis ei olnud.

2. Ennustatud tulemustes olid puudulikud sõnad, mis tegelikult tekstis olid olemas

3. Sõna eksisteeris järgnevuses, aga oli vale, kaasaarvatud osaliselt vale(nt. üks täht vale).

4.2.1 Reaalaja transkriptsiooni veamäär

Sõnaveamäära arvutamine oli järgmine: (üleliigsed sõnad + puuduolevad sõnad + valesti tõlgendatud sõnad) / kõik sõnad manuaalselt kontrollitud transkriptsioonis. Selle veamõõtmise meetodi juures olid muidugi mõningad puudujäägid, nagu sõna valeks lugemine kui esines kõigest üks vale täht, aga sõna semantilises mõttes oleks lauses korrektne. Siiski on see metoodika endiselt laialdaselt kasutuses[59].

Tabel 4.2 Reaalaja mudeli tulemused kvantimisel ja ilma.

WER	VIDEOLÕIK 1	VIDEOLÕIK 2	VIDEOLÕIK 3
tiny.en	23.4%/13.5%*	7.5%	4.4%
tiny.en (kvanditud)	3.8%	5.6%	3.9%

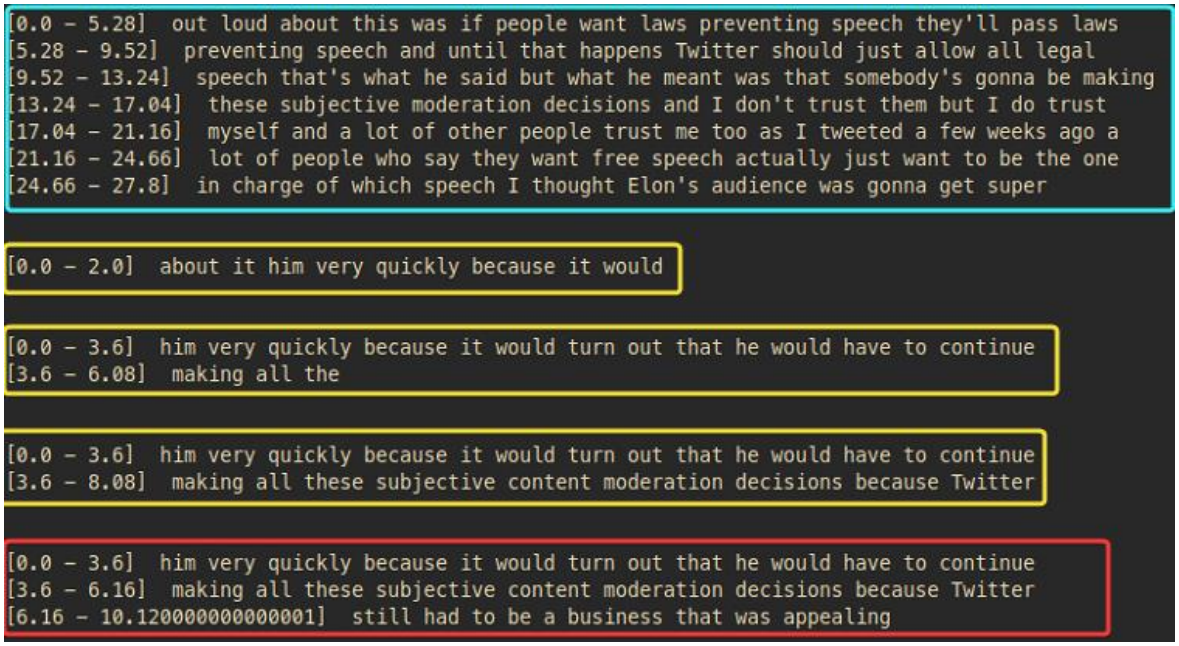
Tabeli 4.2 tulemustest nähtus, et VIDEOLÕIK 1 puhul võis esineda anomaaliaid kvantimata mudeli puhul, sest sõnaveamäär oli mitmekordselt suurem tabelis järgmise kõige halvema tulemuse suhtes. Autor otsustas läbi viia ühe täiendava mõõtmise ja märkida uue mõõtmise tulemuse tabelis tärniga. Teise mõõtmise tulemusena saavutati sõnaveamäär 13.5%, mis oli endiselt teistest kõrgem. Kuna inimene antud videolõigus rääkis võrdlemisi kiiresti ja erilisi pause pidamata, siis on võimalik, et kvanditud mudeli tulemus VIDEOLÕIK 1 puhul oli parem väiksema arvutuskeerukuse tõttu.

Muidugi ei saa siinkohal eirata reaalaja rakenduse muutujaid, mis on antud lahenduses seadistatud kindlatele väärtustele, ehk lauselõpp loetakse peale 2 sekundit vaikust. Seega süsteemil puudub võimekus end dünaamiliselt korrigeerida, siis mudeli jaoks võivad sulanduda lause algused ja lõpud kokku. Sellest tulenevalt on mõningane kadu mudeli täpsuses.

4.2.2 Reaalaja rakenduse töö

Kõne transkribeerimisel on äärmiselt oluline segmentide tuvastamine. Segment on kõne osa, mis sisaldab ühte täielikku mõtet või ideed. Whisper ASR-mudel on võimeline tuvastama kõnesegmente ja määratlema nende algus- ja lõpuajad. See on oluline mitmel põhjusel. See aitab süsteemil mõista, millal üks idee algab ja lõpeb, mis aitab parandada transkriptsiooni loetavust. Teiseks võimaldab see kasutajal näha,

millised kõnesegmendid vastavad transkribeeritud teksti osadele, mis võib olla kasulik näiteks kõne analüüsimisel või kõne sisu jälgimisel.



Joonis 4.1 Whisper transkriptsioonide väljund algus- ja lõpuaegadega.

Joonisel 4.1 on võetud tekstiline väljund, mis saadi reaalaja süsteemiga rakendades seda „Elon Was Right" videoklipile[58]. Tekstiväljundil on Whisper mudeli poolt tagastatud segmentide info, mis asub kasti sees enne teksti, milles on näha ära segmendi alguse ja lõpu aeg.

Sinise kastiga piiritletud väljund määrab ära ühe mõtte, ehk segmendi, mille Whisper mudel tuvastas, ning mis on süsteemi mõistes semantiliselt ühte kuuluv sisu.

Punase kastiga piiritletud väljund määrab ära järgneva mõtte ja kollaste kastidega on näha kuidas üks segment moodustus. Kollaste kastide järgi on näha, et seni kuni üks mõte ei ole lõppenud, siis senikaua lisatakse heliandmete puhvrisse järjestikkuseid segmente, andes võimaluse mudelile täiendada ja isegi korrigeerida sisu pidevalt uueneva konteksti põhjal.

Siit nähtub ka antud reaalaja rakenduse nõrkus, mis on näha kahe segmendi vahelise perioodi järgi, vahemik kus sinine kast lõppeb ja algab kollane. Segmentide vahelt on sõnad puudu ja kollase segmendi algul ka täiesti vale sõna, kuigi see hiljem parandatakse ülejäänud konteksti abil. Sinise kasti ja kollase kasti ühendav lause peaks olema „… *I thought Elon's audience was gonna get super mad at him very quickly*…".

Kuna Whisper mudel on treenitud 30 sekundiliste heli andmete peal ja sarnaselt töötab ka mudeli tuletamise funktsionaalsus, ehk kui heli andmete puhver ületab 30 sekundi piiri, siis kõik mis tuleb peale seda mudel eirab.

Autor arvab, et see võib ka selgitada Tabeli 4.2 VIDEOLÕIK 1 puhul nähtud kõrget sõnaveamäära. Videolõigus on ka keskmise inimese suhtes tegemist väga kiire kõnega, milleks on kogu videolõigu puhul 887 sõna ja seda 4 minuti jooksul, see teeb rääkimise kiiruseks ligikaudu 220 sõna minutis. Erinevate allikate põhjal nähtub, et keskmine inimene räägib umbes 140-170 sõna minutis. On tõenäoline, et kuna rakendus ei ole võimeline dünaamiliselt puhvreid muutma, siis kiiremate kõnelejate puhul võib mudel tekitada rohkem vigu.

Joonise 4.1 juures nimetatud videolõigu puhul oli rakenduse keskmine tuletamise kiirus esimesel mõõtmisel 0.86 sekundit ja teisel mõõtmisel 0.8 sekundit. See on saadud võttes ajadiferentsiaal enne ja pärast transkribeerimise funktsiooni, ilma võtmata arvesse teisi andmete liigutamise, lugemise ja kirjutamise operatsioone. Koos teiste operatsioonidega mõõdeti keskmiseks ühe täieliku tsükli sooritamiseks esimesel testimisel 1.02 sekundit ja teisel testimisel 0.98 sekundit.

Olenemata teatud takistustest ja mõningatest puudujääkidest rakenduse enda suhtes on autor siiski saavutatud tulemustega rahul, sest mudel teeb tööd ligilähedal reaalajale, olenevalt lausete(või segmentide) pikkusest. Viivitusi lisavad juurde vajadused puhverdamise jaoks, seega hetkelisest reaalajast rääkida ei saa, aga arvestades, et maksimaalne viivitus jääb siiski sekundi kuni kahe piiresse peale segmendi lõppu, on tulemus täiesti arvestav.

4.3 Top2vec teemamudeli tulemused

Nagu selgus ka süsteemi arendamise faasis, on teemamudeli jaoks vaja palju rohkem andmeid, kui antud uurimistöö käigus oleks olnud mõistlik tekitada reaalajas transkribeerimisega. Videoklippide arvu tõstmine võimaldas suurema koguse transkribeeritud tekste tekitada ja võimaldas teemamudeli loomist, aga üldiselt treenitakse teemamudeleid vähemalt mõnesaja kuni tuhandete dokumentide peal[26]. Sellest tulenes mudeli erinevate leitud teemade vähesus.

Siiski, ei olnud mudeli loomine asjata, kuna autor tundis videomaterjali sisu ja oli teada, et kanalit juhivad kaks venda, kes suhtlevad üksteisega üleslaaditud videote

vahendusel. Sellest nähtus, et videomaterjali loojatel on kombeks kasutada teatud kindlaid pöördumisi üksteise poole nagu „Hey, Hank!" või „Hey, John!", siis neid võtmesõnu kasutades oli võimalik vähemalt funktsionaalselt ära näidata, kuidas taoline süsteem kasulik võiks olla ja kuidas on võimalik saadud mudelit analüüsida.

Oluline vahe top2vec mudelite ja üldtuntumate metoodikate nagu LDA vahel on just see, et top2vec mudeli puhul ei pea kasutaja mõtlema toorandmete puhastamisele, ega andmete korrigeerimisele. Seda teeb top2vec teek automaatselt, küll on võimalik muuta erinevaid muutujaid, et tekstitöötlemist hõlbustada, aga antud uurimistööst jäi see kõrvale.

Top2vec põhinev lahendus teeb ka äärmiselt lihtsaks mudeli talletamise, et mudeliga oleks võimalik analüüsi või täiendavat treenimist läbi viia. Mudelite taas mälusse lugemine on samuti võimaldatud juba top2vec teegi poolt, mis teeb uute dokumentide mudelisse sisestamise lihtsamaks.

KOKKUVÕTE

Käesoleva töö üheks eesmärgiks oli uurida saadavalolevaid kõnetuvastusmudeleid, valida neist üks ja kasutada reaalajas teksti transkribeerimiseks. Töö teiseks eesmärgiks oli reaalaja süsteemi väljundi teemade jaotus ja indekseerimine. Töö käigus tehti järgmised valikud:

Esmalt valiti Whisper mudel, mis on süvaõppepõhine helituvastussüsteem, treenitud ulatuslikul andmekogumil ja mille puhul on saadavaks tehtud eeltreenitud mudelid.

Seejärel viidi läbi eksperimentaalsed tööd, kus rakendati Whisper mudelit reaalajas heli transkribeerimiseks. Testiti reaalaja rakendust, et hinnata mudeli jõudlust ja täpsust erinevates stsenaariumides, millest väiksema sõnaveamäära saavutas Whisperi tiny.en kvanditud mudel.

Saadud tulemused näitasid, et Whisper mudel suudab edukalt tuvastada ja transkribeerida heliklippe erinevatest allikatest, saavutades kontrollandmestikul sõnaveamäära mis jääb enamik juhtudel 5+-2% piiresse. Arvestades rakenduse protsessidele ja funktsioonidele kuluvat aega, siis reaalaja transkribeerimisele kuluv aeg jääb umbkaudselt ühe sekundi piiresse, mõnevõrra varieerudes olenevalt lause segmendi pikkusest.

Töö käigus rakendati Top2Vec mudelit transkriptsioonide analüüsimiseks ning teemade ja hierarhiate tuvastamiseks. Mudel võimaldas tuvastada olulisi teemasid tekstikorpuses ning seostada sarnase sisuga dokumendid omavahel, mis võimaldas sügavamat arusaamist dokumentide sisust ja nende omavahelistest seostest.

Kokkuvõtvalt näitas antud uurimus, et OpenAI ASR mudel Whisper suudab efektiivselt transkribeerida toorest helisignaali tekstiks reaalajas. Rakendusliku prototüübi väljatöötamine ja testimine näitas, et suure proovimissagedusega heliandmete töötlemine ja nende andmete edastamine mudelile on teostatav ja annab kasutatavaid tulemusi. Märgitakse ära, et sellist mudelit saab rakendada mitmesugustes valdkondades, nagu tarkvaralahendused, mis nõuavad heli tekstiks transkribeerimist, näiteks telefoni. videokõnede või kasvava multimeediasisu transkribeerimine ja indekseerimine.

Edaspidi võiksid uurimistööd keskenduda mitmele parandusele ja laiendusele:

1. Riistvaralised: Eelnevad uurimistööd on tõestanud, et ka nõrgemate kõnetuvastus mudelite tulemused paranevad, kui rakendada ka lähenemist riistvaralise poole pealt. Paremad mikrofonid ja helisüsteemid võivad parandada helisignaali kvaliteeti, mis omakorda võib parandada transkriptsiooni täpsust[22].

2. Ülekande treenimine: Praegu kasutatav mudel on treenitud mitmesugustele keeltele ja aktsentidele. Mudeli täpsem treenimine, eriti eesti keele suhtes, võib parandada transkriptsiooni kvaliteeti.

3. Hübriidsüsteemid: Praegu kasutatav mudel põhineb täielikult tehisintellektil. Tulevased uurimistööd võiksid uurida hübriidsüsteemide loomist, mis lisaks heli töötlemisele võtavad arvesse ka visuaalse info, et oleks liikuvate nägude ja suu liikumise põhjal veel paremini määratleda kõneleja eristamist, see leevendaks mõnevõrra ka riistvara olulisust.

4. Mudeli piirangud: Nagu mainitud, on Whisperi mudel optimeeritud töötama 30-sekundiliste heliklippidega, mis tähendab, et kui skripti puhver täidetakse rohkem kui 30 sekundi jooksul, võib mudel osa heliandmeid kaotada. Tulevased uuringud võiksid keskenduda sellele probleemile, uurides, kuidas on võimalik optimeerida andmeedastust mudelile, et vältida andmete kaotamist ja parandada transkriptsiooni täpsust.

Antud uurimistöö näitas, et ASR mudelid nagu Whisper suudavad pakkuda reaalajas heli tekstiks transkribeerimise lahendusi. Veel on ka teemade modelleerimise valdkonnas toimunud edasiminekuid, mis võimaldavad andmeid analüüsida ilma teadmata, mis nendes andmetes peidus on. See on suur eelis varasemate lahenduste ees, mis eeldasid teemade ette andmist nii arvuliselt, kui ka semantiliselt.

KASUTATUD KIRJANDUSE LOETELU

[1] „Deep Learning". https://www.deeplearningbook.org/ (vaadatud 10. mai 2023).

[2] M. A. Nielsen, „Neural Networks and Deep Learning", 2015, Vaadatud: 9. mai 2023. [Online]. Available at: http://neuralnetworksanddeeplearning.com

[3] „6S191_MIT_DeepLearning_L1.pdf". Vaadatud: 15. mai 2023. [Online]. Available at: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L1.pdf

[4] D. W. Otter, J. R. Medina, ja J. K. Kalita, „A Survey of the Usages of Deep Learning in Natural Language Processing". arXiv, 21. detsember 2019. Vaadatud: 10. mai 2023. [Online]. Available at: http://arxiv.org/abs/1807.10854

[5] „A Beginner's Guide to Neural Networks and Deep Learning", *Pathmind*. http://wiki.pathmind.com/neural-network (vaadatud 12. mai 2023).

[6] „Natural Language Processing (NLP) - A Complete Guide", 11. jaanuar 2023. https://www.deeplearning.ai/resources/natural-language-processing/ (vaadatud 10. mai 2023).

[7] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, ja S. Narayanan, „A Review of Speaker Diarization: Recent Advances with Deep Learning". arXiv, 26. november 2021. Vaadatud: 11. mai 2023. [Online]. Available at: http://arxiv.org/abs/2101.09624

[8] D. Khurana, A. Koli, K. Khatter, ja S. Singh, „Natural language processing: state of the art, current trends and challenges", *Multimed Tools Appl*, kd 82, nr 3, lk 3713–3744, jaan 2023, doi: 10.1007/s11042-022-13428-4.

[9] W. X. Zhao *et al.*, „A Survey of Large Language Models". arXiv, 28. aprill 2023. doi: 10.48550/arXiv.2303.18223.

[10] A. Vaswani *et al.*, „Attention Is All You Need". arXiv, 5. detsember 2017. doi: 10.48550/arXiv.1706.03762.

[11] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, ja I. L. Moreno, „Speaker Diarization with LSTM". arXiv, 23. jaanuar 2022. Vaadatud: 12. mai 2023. [Online]. Available at: http://arxiv.org/abs/1710.10468

[12] „The Full Story of Large Language Models and RLHF", *News, Tutorials, AI Research*, 3. mai 2023. https://www.assemblyai.com/blog/the-full-story-of-large-language-models-and-rlhf/ (vaadatud 8. mai 2023).

[13] O. Russakovsky *et al.*, „ImageNet Large Scale Visual Recognition Challenge". arXiv, 29. jaanuar 2015. Vaadatud: 12. mai 2023. [Online]. Available at: http://arxiv.org/abs/1409.0575

[14] „A Beginner's Guide to LSTMs and Recurrent Neural Networks", *Pathmind*. http://wiki.pathmind.com/lstm (vaadatud 12. mai 2023).

[15] „6S191_MIT_DeepLearning_L2.pdf". Vaadatud: 18. mai 2023. [Online]. Available at: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

[16] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, ja D. Yu, „Recurrent Neural Networks for Language Understanding".

[17] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, ja I. Sutskever, „Robust Speech Recognition via Large-Scale Weak Supervision". arXiv, 6. detsember 2022. doi: 10.48550/arXiv.2212.04356.

[18] „A Beginner's Guide to Attention Mechanisms and Memory Networks", *Pathmind*. http://wiki.pathmind.com/attention-mechanism-memory-network (vaadatud 12. mai 2023).

[19] A. Kumar, „Demystifying Encoder Decoder Architecture & Neural Network", *Data Analytics*, 25. aprill 2023. https://vitalflux.com/encoder-decoder-architecture-neural-network/ (vaadatud 9. mai 2023).

[20] J. Devlin, M.-W. Chang, K. Lee, ja K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv, 24. mai 2019. doi: 10.48550/arXiv.1810.04805.

[21] E. Deruty, „Intuitive understanding of MFCCs", *Medium*, 15. detsember 2022. https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779 (vaadatud 11. mai 2023).

[22] S. Zheng, W. Huang, X. Wang, H. Suo, J. Feng, ja Z. Yan, „A Real-time Speaker Diarization System Based on Spatial Spectrum", *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, juuni 2021, lk 7208–7212. doi: 10.1109/ICASSP39728.2021.9413544.

[23] E. Arisoy, T. N. Sainath, B. Kingsbury, ja B. Ramabhadran, „Deep Neural Network Language Models", *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Montréal, Canada: Association for Computational Linguistics, juuni 2012, lk 20–28. Vaadatud: 18. mai 2023. [Online]. Available at: https://aclanthology.org/W12-2703

[24] P. Kherwa ja P. Bansal, „Topic Modeling: A Comprehensive Review", *ICST Transactions on Scalable Information Systems*, kd 7, lk 159623, juuli 2018, doi: 10.4108/eai.13-7-2018.159623.

[25] D. Spina, J. R. Trippas, L. Cavedon, ja M. Sanderson, „SpeakerLDA: Discovering Topics in Transcribed Multi-Speaker Audio Contents", *Proceedings of the Third Edition Workshop on Speech, Language & Audio in Multimedia - SLAM '15*, Brisbane, Australia: ACM Press, 2015, lk 7–10. doi: 10.1145/2802558.2814649.

[26] D. Angelov, „Top2Vec: Distributed Representations of Topics". arXiv, 19. august 2020. doi: 10.48550/arXiv.2008.09470.

[27] C. B. Asmussen ja C. Møller, „Smart literature review: a practical topic modelling approach to exploratory literature review", *Journal of Big Data*, kd 6, nr 1, lk 93, okt 2019, doi: 10.1186/s40537-019-0255-7.

[28] Q. Wang, Y. Huang, H. Lu, G. Zhao, ja I. L. Moreno, „Highly Efficient Real-Time Streaming and Fully On-Device Speaker Diarization with Multi-Stage Clustering". arXiv, 20. märts 2023. Vaadatud: 12. mai 2023. [Online]. Available at: http://arxiv.org/abs/2210.13690

[29] S. Schneider, A. Baevski, R. Collobert, ja M. Auli, „wav2vec: Unsupervised Pre-training for Speech Recognition". arXiv, 11. september 2019. Vaadatud: 11. mai 2023. [Online]. Available at: http://arxiv.org/abs/1904.05862

[30] J. Lee, T. Kim, J. Park, ja J. Nam, „Raw Waveform-based Audio Classification Using Sample-level CNN Architectures". arXiv, 3. detsember 2017. doi: 10.48550/arXiv.1712.00866.

[31] K. J. Piczak, „karolpiczak/ESC-50". 11. mai 2023. Vaadatud: 11. mai 2023. [Online]. Available at: https://github.com/karolpiczak/ESC-50

[32] J. J. Huang ja J. J. A. Leanos, „AclNet: efficient end-to-end audio classification CNN". arXiv, 15. november 2018. doi: 10.48550/arXiv.1811.06669.

[33] M. S. Hussain ja M. A. Haque, „SwishNet: A Fast Convolutional Neural Network for Speech, Music and Noise Classification and Segmentation". arXiv, 1. detsember 2018. doi: 10.48550/arXiv.1812.00149.

[34] „Send a recognition request with model adaptation | Cloud Speech-to-Text Documentation | Google Cloud". https://cloud.google.com/speech-to-text/docs/adaptation (vaadatud 12. mai 2023).

[35] „Quotas and limits | Cloud Speech-to-Text Documentation | Google Cloud". https://cloud.google.com/speech-to-text/quotas (vaadatud 12. mai 2023).

[36] „Otter.ai - Voice Meeting Notes & Real-time Transcription". https://otter.ai/ (vaadatud 12. mai 2023).

[37] „Conversation analytics", *Symbl.ai Docs*. https://docs.symbl.ai/docs/conversation-analytics (vaadatud 12. mai 2023).

[38] „Real-Time Transcription APIs", *Symbl.ai*. https://symbl.ai/features/transcription/ (vaadatud 12. mai 2023).

[39] „Introducing Whisper". https://openai.com/research/whisper (vaadatud 14. mai 2023).

[40] „ggerganov/whisper.cpp: Port of OpenAI's Whisper model in C/C++". https://github.com/ggerganov/whisper.cpp (vaadatud 21. mai 2023).

[41] „Welcome to DeepSpeech's documentation! — Mozilla DeepSpeech 0.9.3 documentation". https://deepspeech.readthedocs.io/en/r0.9/index.html (vaadatud 21. mai 2023).

[42] R. Řehůřek ja P. Sojka, „Software Framework for Topic Modelling with Large Corpora". Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. University of Malta, Valetta, MT, lk 45–50, mai 2010. Vaadatud: 21. mai 2023. [Online]. Available at: http://is.muni.cz/publication/884893/en

[43] „AMD Ryzen™ 7 5800X Desktop Processors". https://www.amd.com/en/products/cpu/amd-ryzen-7-5800x (vaadatud 28. mai 2023).

[44] „NVIDIA GeForce RTX 3070 Family", *NVIDIA*. https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3070-3070ti/ (vaadatud 28. mai 2023).

[45] J. M. P. Villar, „Efficient speaker diarization and low-latency speaker spotting".

[46] „PyTorch documentation — PyTorch 2.0 documentation". https://pytorch.org/docs/stable/index.html (vaadatud 23. mai 2023).

[47] „Transformers — CTranslate2 3.13.0 documentation". https://opennmt.net/CTranslate2/guides/transformers.html#whisper (vaadatud 23. mai 2023).

[48] R. Egger ja J. Yu, „A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts", *Frontiers in Sociology*, kd 7, 2022, Vaadatud: 31. mai 2023. [Online]. Available at: https://www.frontiersin.org/articles/10.3389/fsoc.2022.886498

[49] *Do I Have ADHD?*, (2017). Vaadatud: 28. mai 2023. [Online Video]. Available at: https://www.youtube.com/watch?v=byh0aAC5vm0

[50] *The Coming AI Invasion*, (1. aprill 2023). Vaadatud: 28. mai 2023. [Online Video]. Available at: https://www.youtube.com/watch?v=EyhnXafetac

[51] *AI tells me if AI will write novels.*, (2022). Vaadatud: 28. mai 2023. [Online Video]. Available at: https://www.youtube.com/watch?v=ZMTConHyTYU

[52] „WER | Calculate the Word Error Rate with our Tool“, *Amberscript*, 19. jaanuar 2021. https://www.amberscript.com/en/wer-tool/ (vaadatud 29. mai 2023).

[53] „FFmpeg“. https://ffmpeg.org/ (vaadatud 22. mai 2023).

[54] A. Zhang (Uberi), „SpeechRecognition: Library for performing speech recognition, with support for several engines and APIs, online and offline.“ Vaadatud: 23. mai 2023. [MacOS :: MacOS X, Microsoft :: Windows, Other OS, POSIX :: Linux]. Available at: https://github.com/Uberi/speech_recognition#readme

[55] „Whisper“. OpenAI, 22. mai 2023. Vaadatud: 22. mai 2023. [Online]. Available at: https://github.com/openai/whisper

[56] „Quantization — PyTorch 2.0 documentation“. https://pytorch.org/docs/stable/quantization.html (vaadatud 22. mai 2023).

[57] S. Gureja, „https://www.scaler.com/topics/pytorch-quantization/“, *Scaler Topics*, 17. jaanuar 2023. https://www.scaler.com/topics/pytorch-quantization/ (vaadatud 22. mai 2023).

[58] „Whisper realtime-ASR - YouTube“. https://www.youtube.com/playlist?list=PLDLPEIOlNflFzANb0KERb8x7z7gnMrchA (vaadatud 31. mai 2023).

[59] F. Chiusano, „Two minutes NLP — Intro to Word Error Rate (WER) for Speech-to-Text“, *NLPlanet*, 3. veebruar 2022. https://medium.com/nlplanet/two-minutes-nlp-intro-to-word-error-rate-wer-for-speech-to-text-fc17a98003ea (vaadatud 30. mai 2023).