Tallinna Ülikool

Digitehnoloogiate instituut

Informaatika

Vabavaralisel kõnetuvastusel põhinev autentimisrakendus

Bakalaureusetöö

Autor: Henrik Romanenkov

Juhendaja: Andrus Rinde

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Sissejuhatus

Inimeses jaoks on kõige tavalisemaks suhtlusviisiks kõne. Läbi kõne väljendame me enda vajadusi ning soove. Läbi küsimuse soovime me saada vastuse millelegi uuele või huvipakkuvale. Olenevalt emotsioonidest võib inimese hääle kõla muuta öeldu tähendust. Lisaks mõjutavad häält mitmed tegurid, nagu inimese sugu, kodune kasvatus, kuid ka hetke emotsioonid. Selle tulemusel on kõnel väga palju omadusi: kõneleja aktsent, tekitatava hääle tugevus ja rääkimise kiirus, hääldus, nasaalsus, heli kõrgus ning muu selline. Lisaks on veel välised tegurid, nagu näiteks taustamüra ja kaja. See kõik teeb kõnetehnoloogia väga huvitavaks, kuid ka keeruliseks valdkonnaks.

Tehnoloogia vaatenurgast on kõnetuvastus väga tähtis nii tänapäeval kui ka tulevikus. Ühiskond on järk-järgult üle minemas tehnikast sõltuvale eluviisile, kus väga palju tehakse meie eest ära. Milleks teha endale ise kohvi, kui masin teeb seda paremini ja kiiremini? Sellised küsimused ümbritsevad meid hetkel ning neid lahendatakse kogu aeg. Nutikad süsteemid pakuvad siinkohal lahendusi, kuid seda ainult juhul, kui need muudavad protsessi kiiremaks ja mugavamaks. Tehnoloogiaid, mille abil selliseid lahendusi luua, on mitmeid, kuid kõige naturaalsemaks viisiks on rääkimine, mille jaoks pole vaja teha muud, kui avada suu. Seetõttu on kõnetuvastus üks parimatest võimalustest, mille abil luua tarku süsteeme. Alustades lihtsatest asjadest, nagu kohvi tegemine või telefonist info otsimine ning lõpetades isesõitvate autode ning tehisintellektidega. Kõik see saab olema ja kohati ka juba on võimalik, kui suudetakse lahendada probleemid, mis hetkel veel esinevad.

Seetõttu pole üllatav, et kõnetuvastusega ning ka baastasemel inimhääle arusaamisega tegelevad suured ettevõtted, nagu Google, Apple, Microsoft, Amazon ja paljud teised. Kõik need firmad tegutsevad ning arendavad lisaks teistele süsteemidele ka kõnetehnoloogiat, kuna see on vältimatu meie arenevas maailmas, kus inimesed muutuvad üha mugavamaks ning soovivad tooteid, mis teeksid nende elu lihtsamaks.

Antud bakalaureusetöö eesmärgiks on tutvustada kõnetehnoloogiat, luua vabavaralistel lahendustel põhinev kõnetuvastust kasutav autentimisrakendus ning anda ülevaade kõnetuvastustehnoloogiast ja selles esinevatest probleemidest.

Töö teema valiti autori enda huvist kõnetuvastustehnoloogia vastu ning soovist luua rakendus. Autor on varem kõnetuvastusega kokku puutunud seminaritööd koostades. Kõnetuvastusel põhineva rakenduse arenguga ning reaalsete lahendustega siiski kokkupuuteid pole.

Eestis on kõnetuvastusega tegeletud juba aastaid ning selle ajaga on tehtud märkimisväärseid edusamme. Siiski on puudus korrektsest materjalist ning õpetusest, kuidas lihtsamat kõnetuvastusel põhinevat rakendust luua.

Töö eesmärgi saavutamiseks antakse kõigepealt lühike ülevaade kõnetuvastuse ajaloost nii Eestis kui ka maailmas, tutvustatakse kasutusvaldkondi, kus see tehnoloogia kasutust leiab ning antakse ülevaade, millistest komponentidest peaks üks kõnetuvastusrakendus koosnema. Peatüki lõpus võrreldakse ning kirjeldatakse loodava rakenduse jaoks kasutatavaid tööriistu. Töö esimeses peatükis oleva kõnetuvastust tutvustava materjali kohta saab pikemalt uurida autori poolt valminud seminaritööst. Töö teises peatükis luuakse kõnetuvastusel põhinev rakendus, kirjeldatakse, kuidas toimus arendus, millistele nõuetele rakendus vastab ning testitakse rakenduse võimekust.

1. Kõnetuvastus ja tarkvararaamistikud

Kõnetuvastus (*speech recognition*) on arvuti abil suulise teksti (kõne) muundamine kirjalikuks (märgijadaks). Rahuldav tase saavutati 1993. aastal (60 sõna minutis, tuvastatavus 90–95 %) (VE, 2006).

Lisaks kõnetuvastusele kasutatakse sama valdkonna kohta ka mõistet „häältuvastus“. Kas see on siiski üks ja sama? Võrreldes eestikeelseid sõnu „kõne“ ja „hääl“, on arusaadav, et need ei tähenda sama asja. Kuidas on aga mainitud terminite tähendusega antud töös uurimise all olevas valdkonnas?

Veebipõhine teatmik AKIT[1] annab otsides vaste ainult mõistele kõnetuvastus. Sellist mõistet nagu häältuvastus nende andmetes pole. Seetõttu kasutatakse antud töös pigem sõna „kõnetuvastus“, kuigi eestikeelsetes materjalides on sama valdkonna kohta kasutatud ka mõistet „häältuvastus.“

1.1 Ajalugu

Inimesed on proovinud luua masinat, mis oskaks rääkida, juba 18. sajandi teisest poolest, kuid siis polnud eesmärgiks mitte luua kõne tuvastav ja arusaav, vaid lihtsalt rääkiv masin. Esimene automaatne kõnetuvastusmasin valmistati aastal 1952, mil kolm Bell Labs'i teadlast ehitasid ühest kõnelejast koosneva süsteemi numbrite tuvastamiseks. 1950ndatel piirdus tehnoloogia üht kõnelejat tundvate süsteemidega, mille sõnavara oli umbes kümme sõna. (Rabiner & Juang, kuupäev puudub)

Raj Reddy oli esimene inimene, kes hakkas 1960ndate lõpul töötama pideva kõnetuvastuse (*continuous speech recognition*) kallal. Eelnevad süsteemid nõudsid kasutajalt pärast igat öeldud sõna pausi tegemist. Reddy poolt kavandatud süsteemi eesmärgiks oli anda häälkäsklusi malemängu käikude jaoks. (Rabiner & Juang, kuupäev puudub)

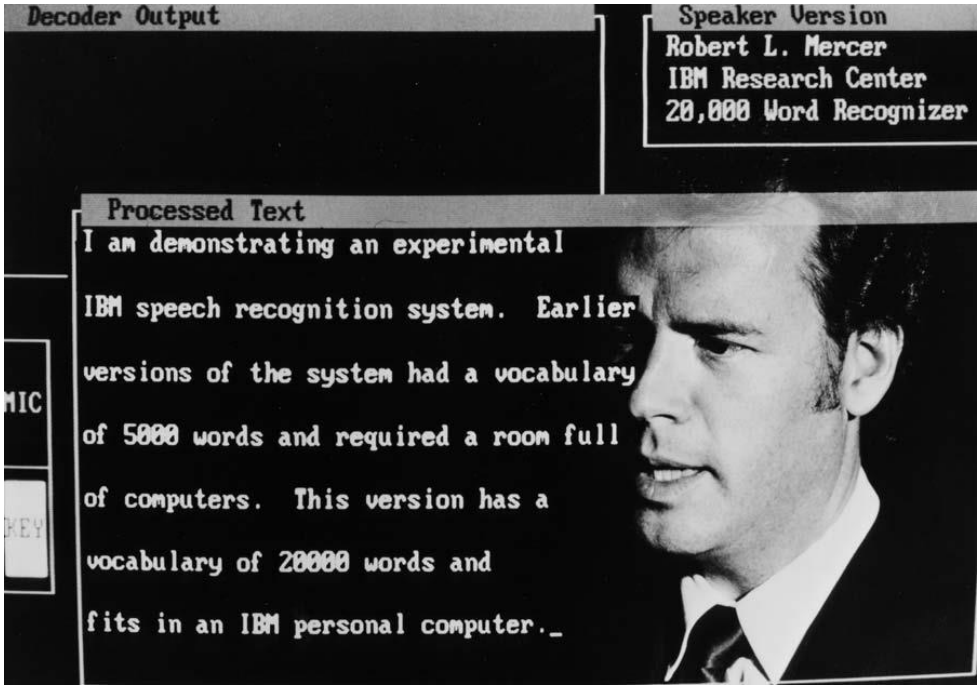
Umbes samal ajal leiutasid Nõukogude teadlased dünaamilise ajadeformatsiooni (*dynamic time warping*, DTW) algoritmi ja kasutasid seda, et luua tuvastaja, mis oli võimeline töötama 200 sõnast koosneva sõnavaraga. Kuigi DTW asendati hiljem uuemate algoritmidega, siis

[1] https://akit.cyber.ee/

signaali kaadriteks jagamise tehnika jätkus. Kõnelejast sõltumatuse saavutamine oli sel ajal teadlaste peamine probleem. (Benesty, Sondhi, & Huang, 2008)

CMUs (Carnegie Mellon Ülikoolis), hakkasid Raj Reddy õpilased kõnetuvastuse jaoks kasutama Markovi peitmudelit (Hidden Markov Model, HMM). HMM'i kasutamine võimaldas arendajatel siduda erinevaid teabeallikaid, nagu akustika, keel ja süntaks, ühtses tõenäosuslikus mudelis. (Lawrence, 2015)

Fred Jelinek'i juhtimise all lõi IBM häälega aktiveeritava kirjutusmasina Tangora, mis suutis 1980ndate keskpaigaks käsitleda 20000 sõnalist sõnavara (Joonis 1). Jelinek'i statistiline lähenemine pani vähem rõhku sellele, kuidas inimese aju töötab ja kõnet mõistab, eelistades kasutada statistilise modelleerimise tehnikaid nagu HMM. Sellega polnud rahul keeleteadlased, kuna nende arvates oli HMM liiga lihtsustatud, et võtta arvesse mitmeid inimkeelte põhiomadusi. Siiski osutus HMM väga kasulikuks ning asendas DTW, saades 1980ndatel domineerivaks kõnetuvastuse algoritmiks. (Rabiner & Juang, kuupäev puudub)



Joonis 1. Tangora kirjutusmasina esitlus (IBM, kuupäev puudub)

1990ndad nägid esmakordselt kaubanduslikult edukate kõnetuvastustehnoloogiate kasutuselevõttu. Kaks varaseimat toodet olid Voiceworks – 1987. aastal Kurzweil Applied Intelligence'i poolt välja antud kõnetuvastusprogramm, millel oli 5000 sõnaline sõnavara, ning Dragon Dictate 30K – kõne põhjal töötav kirjutusmasin, mis anti välja 1990. aastal. AT&T võttis 1992. aastal kasutusse häältuvastamisel kõnede töötlemise teenuse, et suunata

telefonikõnesid inimoperaatorita. Selleks ajaks oli tüüpilise kommertsliku kõnetuvastamissüsteemi sõnavara suurem kui inimese keskmine sõnavara, mis on umbes 5000 sõna. (Rabiner & Juang, kuupäev puudub)

Raj Reddy endine õpilane Xuedong Huang töötas CMU-s välja Sphinx-II süsteemi. Sphinx-II süsteem oli esimene, mis tegi kõnelejast sõltumatut ja suure sõnavaraga kestvat kõnetuvastamist. Kestva kõne käsitlemine koos suure sõnavaraga oli kõnetuvastamise ajaloos märkimisväärne saavutus. Huang liikus 1993. aastal Microsofti ning lõi seal kõnetuvastusrühma. Raj Reddy üliõpilane Kai-Fu Lee liitus Apple'iga, kus 1992. a aitas ta välja töötada kõneliidese prototüübi Apple'i arvutitele, tuntud kui Casper. (Huang, Baker, & Reddy, 2014)

Belgias asuv kõnetuvastuse firma Lernout & Hauspie omandas paljusi ettevõtteid, nende hulgas 1997. a Kurzweil Applied Intelligence ja 2000. a Dragon Systems. L&H kõnetehnoloogiat kasutati Windows XP operatsioonisüsteemis. L&H kõnetehnoloogia ostis ScanSoft, millest sai 2005. a Nuance. Apple litsentseeris algselt Nuance'i tarkvara, et pakkuda kõnetuvastusvõimet oma digitaalsele abistajale Siri. (Wildstrom, 2011)

Google'i esimene jõupingutus kõnetuvastuses tuli 2007. aastal pärast Nuance'i teadlaste palkamist. Esimene toode oli GOOG-411, telefonipõhine kataloogiteenus (Joonis 2). GOOG-411 salvestised andsid väärtuslikke andmeid, mis aitasid Google'il oma tuvastussüsteeme täiustada. Google'i häälotsingut toetatakse nüüd enam kui 30 keeles. (Google, 2010)

Joonis 2 GOOG-411 tööprtosess (Gechlik, 2007)

Eestis on kõnetuvastusega tegeletud suhteliselt vähe. Tallinna Tehnikaülikooli (TTÜ) Küberneetika Instituudi foneetika- ja kõnetehnoloogia laboris tehti esimesed katsetused eestikeelse kõnetuvastusega juba kaheksakümnendate lõpus. Tõsisemalt hakati kõnetuvastusega tegelema alles 2000ndate keskel, millele andis ajendi kahe suure eestikeelse kõne andmebaasi loomine. Need andmebaasid võimaldasid treenida juba suhteliselt hästi

toimivaid akustilisi mudeleid. Põhiliseks kõnetuvastusega seotud uurimisobjektiks on olnud keelemudel. Eesti keele grammatika, põhiliselt keele süntaksist tulenevate erinevate sõnavormide rohkus, teeb sõnapõhise laiahaardelise statistilise keelemudeli loomise keeruliseks. (Alumäe, 2011)

2010. a viis TTÜ Küberneetika Instituut läbi projekti „Kõnetuvastus“, mille eesmärgiks oli täiustada juba eksisteerivat kõnetuvastustehnoloogiat ja olemasolevaid rakendusi. Projekti tagajärjel valmis mitu Android süsteemil põhinevat äppi, näiteks „Kõneleja“. Need rakendused põhinevad dikteerimisel, kus kõnest tekitatakse tekst. Lisaks loodi projekti raames reaalajaline kõnetuvastuse server, mis on mõeldud umbes paarikümnesekundiliste kõnelõikude jaoks. (EKT, kuupäev puudub)

Lisaks on TTÜs valminud veel kõnesalvestuse brauser, kust saab näha, kui hästi arendatud süsteem erinevat kõnet tuvastab. 2011. aastal valmis samas asutuses ka „Veebipõhine kõnetuvastus“, mille abil saab eestikeelset kõnet sisaldavaid helifaile automaatselt transkribeerida. (Romanenkov, 2017)

Miks ei ole eestikeelsed kõnetuvastuse lahendused veel saavutanud sama head kvaliteeti kui inglisekeelsed lahendused? Vastuseks sellele on eesti keele keerukus ning ressursi puudus. Kuigi kõnekorpuse arendamisega tegeles algusaegadel palju inimesi, siis algoritmide ja eesti keele näidete loomisega tegeles TTÜ vanemteadur Tanel Alumäe üksi. Riigil puuduvad vahendid, et viia läbi suuremõõtmelisi arendusi ning erasektorile see valdkond väga huvi ei paku. Eesti keelel on ka liiga vähe kasutajaid, et sellega saaks tegeleda kommertslikul eesmärgil. (Romanenkov, 2017)

Geenius[2] portaali andmetel kogub Google Eesti inimestelt häälenäidiseid, et luua eesti keele tugi oma kõnetuvastuse süsteemile, näiteks läti ja leedu keel on juba toetatud. Kas ja millal me eestikeelset kõnetuvastust näeme, on raske öelda. Töö tegemise hetkel pole tuge veel avalikustatud ning seetõttu ei saa kommenteerida, kui hea see on.

1.2 Kõnetuvastuse tulevik

Mida toob tulevik? Kuigi seda on raske ennustada, on olemas erinevad arvamused, põhinedes tehnoloogia arengul ning sellel, milliste raskustega hetkel kokku puututakse.

[2] https://geenius.ee/eksklusiiv/google-asus-oma-tehismoistusele-eesti-keelt-opetama-et-varsti-telefonidega-emakeeles-vestelda-saaks/

Praegu oleme me digitaalse ekraani ajastus, kus kogu meie elu on nendega mingil moel seotud. Olgu selleks siis nutitelefon meie taskus, ekraan kohvimasina peal või kuvar toidupoes ostetu eest makstes. Tulevikus hakkavad süstemid olema meie jaoks peidetud. See tähendab, et me hakkame üha rohkem kasutama erinevatel žestidel, silmsidel, kõnel ning isegi ajutegevusel põhinevaid lahendusi, eriti kuna tehnoloogid otsivad võimalusi, mis asendaks nutitelefonide kasutajaliidest. (Hamilton, 2017)

Põhiliseks märksõnaks on universaalsus. Kõne ning hääle abil saab juhtida oma kodu, alustades tuledest ja alarmidest ning lõpetades meelelahutuse ja koduseadmete käsutamisega. Üha rohkem autodest omandavad intelligentsuse, mille abil saab inimene mitte lihtsalt raadiokanaleid vahetada, vaid anda sõidukile juhised kuhu minna, ilma et isik peaks auto juhtimisse sekkuma. Töökohad, mis eeldavad suurt mobiilsust, nagu haiglad, laborid ning tööstushooned, kasutavad kõne abil juhitavaid süsteeme. (Tuttle, 2015)

Uuringute põhjal arvatakse, et aastaks 2020 tehakse igas kuus 200 miljardit päringut internetist kasutades häält, mis moodustab kõikidest otsingutest 50%. See tekitab suure vajaduse süsteemide kiireks arenguks, kuna ettevõtted saavad selle arvelt suurt tulu. Kõnetuvastus peab arenema ning lõpuks jääma nii hea tuvastamise ning arusaamise võimekusega, nagu seda on inimene, sest piisab vaid mõnest veast, et kasutaja loobuks süsteemi kasutamisest. (Tuttle, 2015)

Hetkeseis on tunduvalt teine, kuid kiiret arengut on sellegipoolest märgata. Erinevate uuringute põhjal tehakse hetkel kõne põhjal üle miljardi otsingu ühes kuus. 41% inimestest hakkasid kõnetuvastuse abi kasutama viimase kuue kuu jooksul, 60% viimase aasta jooksul ning 11% alustasid rohkem kui kolm aastat tagasi. Siri’t kasutab 19% inimestest igapäevaselt ning Microsoft Cortanal on 133 miljonit kasutajat igas kuus. Kõnetehnoloogiat kasutab igapäevaselt iga teine nutitelefoni omanik. Nende andmete põhjal võib väita, et kõnetuvastus on suur osa meie igapäevaelust juba praegu ning tulevikus veel rohkemgi. (Jeffs, 2018)

Kuna hääl- ja kõnetuvastust kasutavatel süsteemidel puudub silmaga nähtav kasutajaliides, on väga tähtis, et inimese andmed oleksid kaitstud. Kindel arusaam, kuidas andmeid salvestatakse, analüüsitakse ning kasutatakse on tihti probleem, millest avalikult ei räägita. Tehnoloogiaettevõtted, kes arendavad süsteeme, peaksid näitama suuremat ülevaadet ning vastutama andmete turvalisuse eest. (Hamilton, 2017)

Kogu tulevik on suuresti tehnoloogiaettevõtete käes, kelle võimuses on luua vajalikke ning revolutsioonilisi lahendusi, mille abil haarata inimeste tähelepanu ning pakkuda midagi vajalikku. Sellegipoolest peab kõnetuvastustehnoloogia eelnevalt ületama hetkel esinevad probleemid. Kui edasiminekuks vajalikud tooted on saadaval, jääb tarbijatel otsustada nende vajalikkuse üle. Kindel on see, et kõnetuvastus on siin, et jääda ning vaikselt üle võtta meie igapäevased tegemised. (Hamilton, 2017)

1.3 Kõnetuvastust kasutavad valdkonnad

- **Nutiseadmed**

Kõigil kellel on olemas nutitelefon, on võimalus kasutada sinna sisseehitatud kõnetuvastust. Kaks populaarsemat on Google Assistant ning Apple'i Siri. Siri on küll saadaval rohkematel seadmetel, üle 700 miljoni iPhone'i lisaks muudele Apple seadmetele, vastupidiselt Google 400 milionile seadmele, kuid Google'l on hetkel kõige väiksem sõnavigade arv Ameerika inglise keeles. Selleks on 4,9 %, mis teeb Google'i ainukeseks, kellel vigade arv on alla 5 protsendi. Lisaks nutitelefonidele on kõnetuvastus sisseehitatud paljudele nutikelladele ja -kõlaritele, nagu näiteks Amazon'i Echo ja Google Home. (Velde, 2018)

- **Meditsiin**

Meditsiinis on kõnetuvastus kasutuses peamiselt dokumentatsiooniks. Kõnetuvastust kasutatakse patsientide andmete, protseduuride, raviplaanide ja muu sellise koostamiseks, mis võrreldes kirjutamisega säästab meditsiinitöötaja aega ning vähendab haigla paberimajandust. Lisaks parandab see üldist töövoogu, kuna kasutuses olevad süsteemid võimaldavad eristada kasutajaid, nende aktsente ja varasemaid vigu, mille abil saab süsteemi õppida. (News-Medical, 2017)

- **Autondus**

Paljudel autodel on kõnetuvastust võimalik aktiveerida nupu vajutusega, millest antakse juhile teada helisignaaliga. Signaali kostudes on süsteemil nn "kuulamisvahemik", mille käigus see kõnesisendit aktsepteerib. Kõnetuvastuse võimalused varieeruvad automarkide ja mudelite vahel, kuid mõned uuemad mudelid pakuvad fikseeritud käskude asemel naturaalset kõnetuvastamist. (HyClassProject, kuupäev puudub)

- **Sõjandus**

Viimasel kümnendil on tehtud pingutusi kõnetuvastuse testimiseks ja hindamiseks hävituslennukite pardal. Erinevates programmides on edukalt toiminud kõnetuvastusvahendid, mille hulka kuuluvad raadiosageduste määramine, autopiloodisüsteemi juhtimine, suunamispunkti koordinaatide ja relvade vabastamise parameetrite seadistamine ning lennukuva kontrollimine. (HyClassProject, kuupäev puudub)

- **Haridus**

Keelte õppimisel saab kõnetuvastus olla kasulik teise keele omandamisel. Lisaks õige hääldamise õpetamisele aitab see ka inimesel arendada rääkimise soravust. Nägemisepuudega õpilased saavad kasutada kõnetuvastustehnoloogiat sõnade edastamiseks arvutile ja seejärel kuulata, kuidas arvuti neid esitab. Samuti saavad nad kasutada arvutit, juhtides seda oma häälega, ilma et nad peaksid kasutama klaviatuuri ja hiirt. (HyClassProject, kuupäev puudub)

1.4 Kõnetuvastamisel esinevad probleemid

Igal kõnetuvastusega tegeleval süsteemil on hulk tunnuseid, mis peavad olema täidetud, et rakendus saaks oma tööd teha. Väheste tunnuste puhul võib peaaegu iga rakendus saavutada inimesele võrdse täpsuse, aga probleem ilmneb just siis, kui neid on palju. Selle tulemusel hinnatakse igat süsteemi ning selle täpsust erinevalt alljärgnevate tingimuste alusel:

- **Sõnavara suurus** – Veasagedus on võrdelises seoses sõnavara suurusega. Näiteks: numbrituvastuse süsteemis ei ole numbrite „null“ kuni „üheksa“ äratundmine põhimõtteliselt probleem, kuna sõnavara koosneb ainult kümnest elemendist. Süsteemid, mille sõnavara on suurusega 200, 5000 või 100000, võivad sisalda veasagedust kuni vastavalt 3%, 7% ja 45%. Lisaks muudab sõnavara ära tundmise raskeks keeruliste sõnade sisaldus. (Neuro AI, 2013)
- **Kõnelejast sõltuvus vs sõltumatus –** Definitsiooni järgi on kõnelejast sõltuv süsteem mõeldud tuvastama vaid ühe kindla kõneleja juttu ning kõnelejast sõltumatu on mõeldud kõigile. Sõltumatut süsteemi on raske luua, kuna iga süsteemi parameetrid luuakse vastavalt kõneleja(te)le, kellega süsteemi treenitakse, kuid need parameetrid kalduvad olema kõneleja põhised. (Neuro AI, 2013)
- **Isoleeritud, katkendlik või kestev kõne** – Isoleeritud kõne puhul kasutatakse vaid üksikuid sõnu, mille tõttu on kõnet kergem ära tunda. Katkendlikus kõnes kasutatakse

terveid lauseid, milles iga sõna on eraldatud vaikusega. Kestev kõne tähendab kõne naturaalset esitust. Isoleeritud ja katkendliku kõne tuvastamine on üpriski lihtne, kuna sõnadel on kindlad piirid ning need on enamjaolt puhtalt hääldatud. (Neuro AI, 2013)

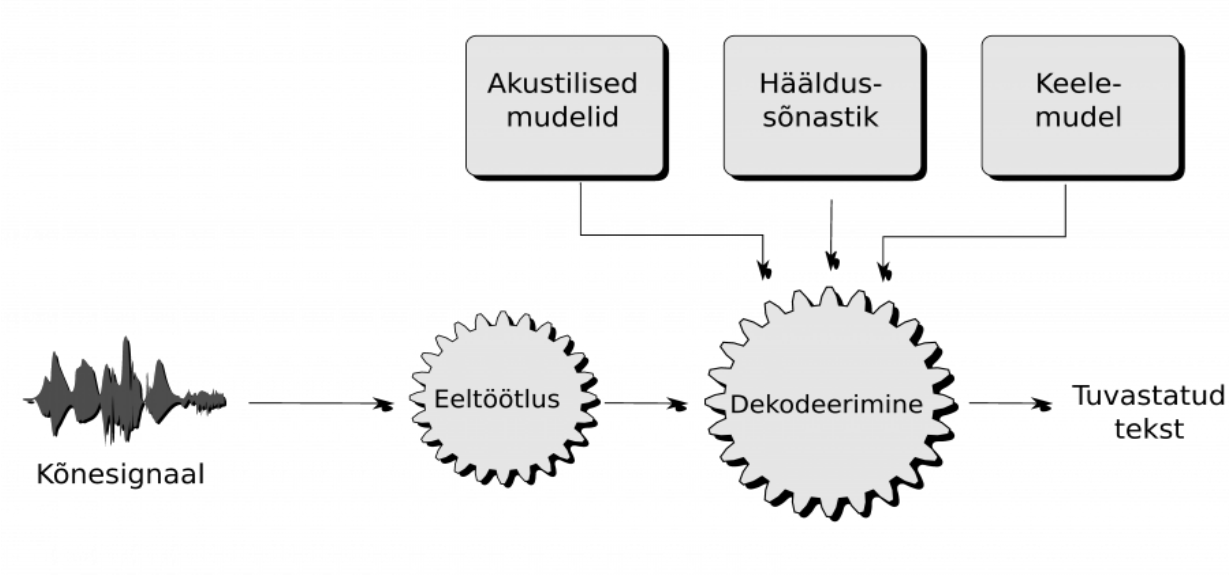
- **Ülesande ja keele piirangud** – Isegi piiritletud sõnavaraga süsteemide puhul erineb tulemus tuvastusele seatud piirangute tõttu. Näiteks võib infopäring loobuda hüpoteesist "Õun on punane.", kuna on olemas ka teisi värve õunu. Lisaks võivad piirangud olla semantilised ning keelduda fraasist "Õun on vihane.", kuna elutul asjal ei saa olla emotsioone või süntaktilised ning lükata tagasi "Vihane õun on.", sest see pole grammatiliselt korrektne. Enamasti esitatakse piirangud grammatikas, mis filtreerib mittesobilikud laused välja, et tuvastaja saaks ainult korrektseid lauseid hinnata. (Neuro AI, 2013)
- **Loetud vs spontaanne kõne** – Süsteeme saab hinnata ka selle põhjal, kui hästi saavad nad aru ettevalmistatud loetud tekstist või spontaansest kõnest. Viimast on tunduvalt raskem ära tunda, kuna see sisaldab kõnekeelest tulenevaid vigu: „ee“ ja „mm“, valed algused, poolikud laused, kogelemine, köhimine ning naer. Selle tulemusel on põhimõtteliselt tegemist piiramata sõnavaraga, kus süsteema peab tundmatute sõnadega arukalt toimetama. (Neuro AI, 2013)
- **Ebasoodsad tingimused** – Kõnetuvastuse suutlikust halvendavad paljud ebasoodsad tingimused. Nendeks on keskkonna helid, näiteks autod ja tehased, akustilised moonutused, näiteks kaja ja ruumi akustika ning erineva kvaliteediga mikrofonid, näiteks lähedalt rääkimise, telefoni ja mitmesuunalise vastuvõtjaga. Lisaks on kõnelejast sõltuvad faktorid, nagu muutlik kõneviis, mille hulka kuuluvad näiteks karjumine, vingumine või vaikselt rääkimine. (Neuro AI, 2013)

1.5 Kõnetuvastuse komponendid

Kõnetuvastus rakenduse toimimiseks on vajalik läbi viia erinevad töötlusi ning omada mitmesuguseid mudeleid, et kõneleja poolt öeldu saaks süsteemi poole väljundiks kõige täpsema vaste (Joonis 3). Sellisteks komponentideks on:

- Eeltöötlus – töödeldakse helisignaal süsteemi jaoks sobivale kujule;
- akustiline mudel – modelleerib häälikuid. Otsustab, mis on kõige tõenäolisem vaste öeldule;

- keelemudel – keele grammatika. Paneb paika, millised sõnad on lubatud ning kuidas neid kombineerida;
- dekodeerimine – leitakse eelnevate punktide abil öeldule õige vaste.



Joonis 3 Kõnetuvastuse protsess (Alumäe, 2011)

1.5.1 Eeltöötlus ja tunnusvektorite arvutamine

Enne parima lahenduse leidmist on vaja esimese sammuna heli salvestada ja digitaliseerida. Pärast seda eeltöödeltakse signaal tuvastuseks sobivale kujule. Signaal lõigatakse lühikesteks lõikudeks, mis on tavaliselt paarikümne millisekundi pikkused ja osaliselt kattuvad, kuid nii pikkad, et signaali spekter ei muutu oluliselt. Järgnevalt viiakse läbi analüüs, et leida spektrist olulised tunnused, mis esitatakse reaalarvude vektorina. Saadud tunnused peaksid võimaldama eristada häälikuid, kuid mitte arvestama ebaoluisi signaali aspekte, nagu taustmüra ja kõneleja emotsioonid. (Alumäe, 2011)

1.5.2 Akustilised mudelid

Kõnetuvastussüsteemi üheks tähtsaimaks osaks on akustilised mudelid. Akustiliste mudelite abil moodustatakse häälikuid. Süsteemi hääldussõnastikus on kirjas kõigi rakenduse poolt tuntavatele sõnadele vastavad häälikujadad. Häälikumudelite ja hääldussõnastiku abil saab luua sõnu ja sõnamudelite abil neist lauseid. Näiteks sõna „meetrit“ häälikujada oleks hääldussõnastikus kujul: „m e e tt r i tt“. (Alumäe, 2011)

Akustiliste mudelite põhiomaduseks on võime öelda mis tahes tunnusvektorite jada kohta, kui tõenäoliselt on see signaal just „minu“ poolt esitatud signaal. Kõnetuvastuses on tähendatud,

et häälikute spekter võib olla tugevalt seotud sellest, millises kontekstis häälik parasjagu on. Seetõttu, modelleeritakse häälikuid nn. kontekstist sõltuvate ühikutega: ühel häälikul on mitu erinevat mudeleid, mida kasutatakse automaatselt vastavalt kõne kontekstist. (Alumäe, 2011)

Akustiliste mudelite kõige kasumlikumad parameetrid arvutatakse suurte kõneandmebaaside põhjal, milles on suur hulk erinevate inimeste kõnet koos selgitustega. Selle info põhjal saab akustilisi mudeleid treenida.Kui andmebaasis on piisav hulk kõnet, peaks treenimise käigus mudelid saama üldistusvõime, ehk sobima ka andmebaasis mitteolevate inimeste kõne tuvastamiseks. (Alumäe, 2011)

1.5.3 Keelemudelid

Akustiliste mudelite abil saame võrrelda süsteemi andmebaasi, kasutaja poolt öelduga. See ei ole aga lause taasesitamiseks piisav. Põhjuseid on siin mitu: inimeste lohakas kõnelemine, sõnalõpud muutuvad dünaamiliselt uue sõna alguseks, sõnade vahel esitatakse kõhklevaid helisi jne. Lisaks on inimeste kõne väga erinev: ühe inimese „o“, häälik, kõlab sarnaselt teise inimese „u“ häälikuga. Isegi, kui häälikude tuvastamine oleks ideaalne, on seda sõnade jadaks keeruline teisendada ilma et teaks midagi keelest, kuna kestva kõne puul puuduvad sõnade vahel pausid, mis aitaksid sõnu piiritleda. (Alumäe, 2011)

Tuvastusmootor suudab tegelikult tuvastada vaid neid sõnu, mis süsteemi keelemudeli sõnastikus esinevad, ning naturaalse keele puhul koosneb sõnastik vaid nendest sõnadest, mis esinevad piisavalt tihti. Ainuüksi sõnade loendamine ei lahenda nn homofoonilisuse probleemi: kas häälikujada /kassaoledvalmis/ vastab lausele „Kas sa oled valmis“ või „Kassa oled valmis“? (Alumäe, 2011)

Selle olukorra lahendamiseks tuleb uurida, kuidas keeles sõnu tavaliselt omavahel kombineeritakse. Näiteks ilmneb, et sõnakolmik „kas sa oled“ on üle 100 korra sagedasem, kui sõnapaar „kassa oled“. Need andmed aitavad arvutil leida igale sõnakombinatsioonile õige vaste. Kuna eesti keeles on erinevaid sõnu väga palju, kasutatakse keelemudelis sõnade asemel morfeeme(näiteks sõna „emaplaadiga“ koosnb kolmest morfeemist „ema“, „plaadi“ ja „-ga“). Morfeemid liidetakse pärast tuvastamist uuesti sõnadeks tagasi. (Alumäe, 2011)

1.5.4 Dekodeerimine

Kõnetuvastuses nimetatakse dekodeerimiseks protsessi, kus treenitud akustilise mudeli ja keelemudeli ning sõnade hääldussõnastiku abil leiab tuvastusmootor sisendlausele kõige

tõenäolisema vaste. Piiratud sõnavaraga tuvastuse puhul on tuvastusprotsess lihtsasti ettekujutatav. Näiteks mobiiltelefonis kasutatava häälvalimisega sarnase rakenduse puhul arvutatakse sisendsignaali vastavus kõigi telefonis olevate nimede akustilise mudeli suhtes, ning seejärel valitakse neist välja kõige tõenäolisem vaste. (Alumäe, 2011)

Loomuliku ja sidusa kõne tuvastuse puhul on olukord raskem, kuid põhimõte on siiski eelnevaga sarnane: antud juhul tuleb kõigi loomuliku keele lausete hulgast valida välja selline, mis on nii akustilise kui ka keelemudeli seisukohast kõige tõenäolisem. Kuna loomulikus keeles on erinevaid võimalikke lauseid praktiliselt lõpmatult, tuleb otsingus kasutada algoritme, mille abil vähem tõenäolisemad harud sellises lausete puus kohe välistada. Selle tegemata jätmisel võtaks dekodeerimine lause pikkusega võrreldes palju rohkem aega. (Alumäe, 2011)

Sellist otsinguruumi piiramist nimetatakse kärpimiseks (*pruning*) ning kärpimise ulatuse abil saab protsessi kiirust parandada. Liigne kärpimine on aga halb, kuna selle tulemusel lõigatakse ära sellised arud, mida mõningal juhul võib hiljem vaja minna. (Alumäe, 2011)

1.6 Ülevaade kõnetuvastuse raamistikest ja API'dest

Antud peatükis tutvustatakse lühidalt populaarsemaid vabavaralisi, mille põhjal luuakse ka valmiv rakendus, kuid ka tasulisi lahendusi. Kõnetuvastusrakendust on võimalik luua ja rakendada raamistike ning API'de abil. Suurettevõtted pakuvad tihti API'sid, kuna selle tulemusel saavad kasutajad ettevõtte poolt arendatud süsteeme enda omasse lihtsalt rakendada. Raamistikud nõuavad rohkem kasutaja enda poolset sekkumist.

1.6.1 Kõnetuvastuse raamistikud

Raamistikud jagunevad tasulisteks ning vabavaralisteks. Enamasti on tasulised süsteemid mõeldud ettevõtetele või asutustele, mis soovivad kasutada kõnetuvastust, kuna pakutakse terviklahendusi, mida tuleb kohandada iga kliendi vajadustele. Lisaks on neil palju lisafunktsioone, mida vabavaralised ei paku või siis pakuvad mingi tasu eest.

Dragon NaturallySpeaking – Nuance Communications'i poolt arendatav ja välja antav kõnetuvastussüsteem, mida on võimalik kasutada arvutis. Süsteem võimaldab dikteerimist, käskluste abil töö tegemist ning teksti kõneks muutmist. Lisaks on võimalik seada ka endale

meelepäraseid käsklusi programmide käsitluseks või mõne funktsionaalsuse jaoks. Peale selle toote pakub ettevõte veel hulganisti erinevaid lahendusi eri valdkondades.

Tazti – Kõne- ja häältuvastust pakkuv ettevõte, mille toote abil saad lihtsalt avada erinevaid faile, esitada muusikat, kasutada rakendusi ning mängida mänge. Süsteemi abil on võimalik seada käsklusi, mille abil saab arvuti teha täpselt seda, mida on vaja. Rakendus töötab ainult Windows'i operatsioonisüsteemil, mille versioon on 7 ja uuem. Apple arvutitel on seda samuti võimalik kasutada, kuid läbi *Desktop*'i visualiseerimise tarkvarade nagu Parallels.

Lisaks tasulistele lahendustele on veel ka vabavaralised raamistikud. Kuigi vabavaralisi lahendusi on mitmeid, on antud töös tutvustatud ainult kahte: CMU Sphinx ja Kaldi (Tabel 1). Põhjus selleks tuleneb sellest, et antud süsteemid on ka hetkel edasiarendamisel, neil on piisavalt funktsionaalusi ning on olemas mõnigad õpetused ja dokumentatsioon. Sphinx süsteemil on olemas eraldi versioon PocketSphinx, mis on nende poolt pakutavatest kõige parema tuvastusega ning mõeldud süsteemidesse rakendamiseks, sellepärast on siin võrdluses kasutatud just seda.

Tabel 1. Vabavaraliste lahenduste võrdlus

	Pocketsphinx	Kaldi
Dokumentatsioon	Olemas põhjalik ning selgesti arusaadavad õpetused, kuidas süsteemi kasutada.	Olemas küll korralik dokumentatsioon, kuid puuduvad selged õpetused. Algajale on keeruline.
Mobiilsete seadmete tugi	On olemas ning lihtne kasutada, kodulehel õpetused.	On võimalik, aga algajale keeruline siduda teiste süsteemidega.
Tuvastuse täpsus	Halvem, kuna kasutab HMM'i, mis iseenesest ei ole halb, aga tänapäeval on olemas paremaid. Sõnavigade arv on suur, mõne allika põhjal umbes 20%.	Parem, kuna kasutuses ka kommertstoodetes kasutuses olevat *Neural Networks* algoritmi, mille sõnavigade hulk jääb alla 10%.
Arenduskeskkond	Linux, Windows, MacOs.	Soovitavalt Linux, võimalik ka Windows'il.

Erinevate artiklite põhjal võib väita, et Kaldi tuvastus on tunduvalt parem võrreldes Sphinx süsteemiga, kuna tuvastuseks kasutatakse *Neural Networks* algoritmi, mitte HMM'i. Siiski soovitatakse algajatel kasutada Sphinx'i, kuna selle võimalused on suuremad ning dokumentatsioon parem. Lisaks pakub Sphinx versiooni PocektSphinx, mis oli ka eelnevas võrdluses ning mida on väga kerge rakendada teistesse süsteemidesse, näiteks veebilehte või mobiilirakendusse.

Kaldi kodulehel on isegi kirjas, et see on pigem mõeldud kõnetuvastusega tegelevatele teadlastele ning süsteemide arendajatele. See ei tähenda, et huvilisele on see süsteem välistatud. Mõlema süsteemi võrdluse aspektist ning materjalide läbi töötamisest võib väita, et selle tundma õppimine võtab kauem aega.

1.6.2 Kõnetuvastuse API'd

Lisaks tasulistele ja vabavaralistele raamistikele on päris palju erinevaid API'si (*Application programming interface, rakendusliides*), mille abil saab olemasoleva kõnetuvastussüsteemi, näiteks Google või Amazoni poolt, enda loodud rakendusse või seadmesse integreerida. Enamus juhtudel on need mingi tasemini tasuta, kuid kui on vajadus lisafunktsioonide või suurema kasutajate hulga järele, siis muutuvad need tasuliseks. Järgnevalt on väike valik kõige populaarsematest kõnetuvastuse API'dest.

Alexa Voice Service – Amazon'i poolt loodud pilveteenus, mis pakub vahendeid, mille abil ühendada Alexa enda poolt loodud tootesse. Nende poolt tuleb kogu tarkvara, mis on vaja olemasolevasse koodi lisada ning selle tulemusel on enda poolt loodud seadmel Alexa tugi, mille abil teha erinevaid päringuid. (Amazon, 2018) Alexa tugi on ehitatud sisse näiteks Huawei Mate 9 nutitelefoni, Ecobee 4 termostaati ning Garmin Speak pardakaamerasse. Lisaks veel erinevatesse valgustuslahendustesse, turvaseadmetesse, pistikutesse ja muudesse tarkadesse süsteemidesse.

Google Cloud Speech-to-text – Google'i pilveteenusel põhinev kõnetuvastuse API. Tänu Google võimekale masinõppe tehnoloogiale, saab tuvastaja hakkama 120 keelega (hetkel veel mitte eesti keelega) ning erinevate kõne tuvastamise moodustega. Piiranguks on tuvastusele kuluv aeg ühes kuus, mis on piiratud 60 minutile. Lubatud ajast üle minnes peab hakkama maksma iga 15 sekundi pealt.

Apple Speech Recognition API – Apple poolt pakutav vabavaraline API, mis lubab iOS operatsioonisüsteemil töötavatel äppidel kasutada sama kõnetuvastuse süsteemi, mis on Siri's.

API pakub tuge nii eelnevalt salvestatud kui ka naturaalsele kõnele, tõlkides selle tekstiks. Puudujäägiks on limiteeritud päringute arv ühes päevas ning ühe audiosalvestuse kogupikkus ei tohi olla rohkem kui üks minut. (Center, 2016)

Lisaks neile on veel olemas ka HTML5 Web Speech API, mille abil on võimalik oma veebilehte integreerida mõne teenusepakkuja kõnetuvastussüsteemi. Hetkel pakuvad seda Google Chrome, Firefox Desktop ning osaliselt ka Safari.

2. Vabavaralisel kõnetuvastusel põhineva autentimisrakenduse loomine

Antud bakalaureusetöö üheks eesmärgiks oli luua kõnetuvastusel töötav rakendus. Selles peatükis selgitatakse arenduseks vajalikke nõudeid, antakse ülevaade vajalikest töövahenditest, kirjeldatakse rakenduse arendusprotsessi ning seda, kuidas rakendus toimib. Lisaks testitakse valmis rakenduse täpsust ning antakse ülevaade soovituslikest edasiarengutest ja lahendust vajavatest probleemidest.

Rakenduse loomiseks vajaliku süsteemi valikuks on seatud tingimused, mille abil on parimat lahendust lihtsam välja selgitada.

Tingimused:

- Rakeduse tüüp on Android'i operatsioonisüsteemil põhinev äpp – raamistik peab töötama Android seadmetel;
- Rakendus luuakse Windows operatsioonisüsteemil – arenduskeskkond ning kõnetuvastuse süsteem toetavad Windows'i.

Sellised tingimused said valitud autori enda soovil ning lähtudes töö eesmärgist, milleks on luua autentimisrakendus. Arendatava rakenduse tüübiks sai valitud äpp, kuna see võiks olla üks viis, kuidas erinevatesse rakendustesse sisse logida. Kuna tegemist on vabavaralise lahendusega ning selliseid süsteeme palju pole, siis turvalisuse huvides on soovitatav kasutada kõnetuvastust koos mõne teise tuvastusviisiga, näiteks näotuvastuse või salasõnaga. Lisaks soovib autor omandada teadmisi äppide loomisel. Äpi arendamise teeb lihtsamaks ka asjaolu, et telefonidesse on ehitatud sisse mikrofon, mis teeb tuvastamise lihtsamaks, seda eriti võrreldes lauaarvutitega.

Arendust viiakse läbi Windows'i süsteemis, kuna just see on loodava rakenduse jaoks kasutatavas arvutis. Lisaks on võimalik kasutada ka Linux'it ning MacOS'i, kuid sel juhul tuleks luua virtuaalne operatsioonisüsteem. Kuna arendatav rakendus on äpp, siis puudub vajadus kasutada teisi operatsioonisüsteeme, mis tuleks luua virtuaalses seadmes.

Nende tingimuste tulemusel sai rakenduse arendamiseks valitud PocketSphinx, millele tehakse Android operatsioonisüsteemil põhinev äpp. Valitud süsteemist saab pikemalt lugeda peatükis 1.6.1.

Enne rakenduse arendamist tuli välja selgitada funktisonaalsused ning teha analüüs, mida on võimalik luua ning mida mitte. Selle põhjal selgusid järgnevad baastingimused:

- Rakenduses on võimalik luua kasutaja;
- Kasutaja saab sisse logida;
- Kõnetuvastust kasutatakse teise astmena sisselogimisel;
- Kasutajal peab olema valik lubamaks aktiveerida mikrofon.

Rakenduse ülesehitus on omane kõige tavalisemale kasutaja tuvastamisele. Rakenduse avades on vaja esmalt luua kasutaja. Kasutajat luues kontrollib süsteem, et väljad oleksid täidetud ning et antud kasutajanimele ei oleks andmebaasis vastet. Turvalisemat lahendust on keerulisem luua ning tavaliselt on ettevõttel parooli tuvastamiseks ning kontrollimiseks ka omad viisid.

Peale edukat kasutaja loomist on võimalik sisse logida ning sealjuures küsitakse varasemalt loodud kasutajanime ning salasõna. Jällegi kontrollib süsteem kasutajanime olemasolu andmebaasis, kuid mitte salasõna. Kui sisestatud väljad on vastavuses, luuakse ühendus kõnetuvastuse süsteemiga ning käivitatakse mikrofon. Mikrofon ootab süsteemi poolt ette antud sõna või väljendit. Kui salasõna esitatakse edukalt, kuvatakse ekraanile teade „Sisenemine õnnestus“ ning hetkel suunatakse kasutaja uuesti avalehele. Reaalses rakenduses peaks siiski kasutajale avanema talle mõeldud sisu.

Kõnetuvastuse poole pealt ei pea kasutaja praegu ise midagi tegema, sest süsteem on juba ise terviklik ning treenitud. Puudub vajadus andmeid treenida nagu seda tehakse näotuvastuses. Antud töös arendatud rakendus kasutab süsteemi poolt kaasatulnud algoritme ning tuvastuskeeleks on inglise keel, millel on olemas väga põhjalik häältussõnastik ehk keelemudel.

2.1 Vahendid rakenduse loomiseks

Rakendusele seatud tingimuste täitmiseks on vaja arenduskeskkonda, mille abil luua arendatavale äpile funktsionaalsused. Lisaks on vaja ka kõnetuvastusega tegelevat rakenduse osa.

Android operatsioonisüsteemile äpi arendamiseks on vaja alla laadida Android Studio[3] - IDE (*integrated development environment*), mis sisaldab kõiki vajaminevaid teeke ning abivahendeid, et enda loodud äpp tööle saada. Kuna autoril puudus varasem kogemus selle süsteemiga, siis sai läbi tehtud kodulehel olev lühike õpetus, mis tutvustab äpi arendamise võimalusi ning kuidas erinevad funktisonaalsused, nagu erinevate lehtede vahel liikumine ning disaini elementidele funktsionaaluse lisamine, toimivad. See on soovitatav ka teistele, kellel puudub kogemus selle programmiga.

Arenduseks saab valida mitme programmeerimiskeele vahel. Ametlikuks keeleks on seatud Java, kuid valikus on veel ka näiteks Kotlin[4] ja C++[5]. Kokkupuude kas Java või mõne muu Android Studio poolt aktsepteeritud keelega tuleb arendamisel kasuks ning säästab aega.

Lisaks arenduskeskkonnale on vajalik ka kõnetuvastuse mootor, milleks töö autor valis PocketSphinx'i. Enne selle kasutust on soovituslik Sphinx'i kodulehelt alla laadida näidisrakendus Android Studiole. See annab võimaluse proovida süsteemi võimekust ning saada ülevaade sellest, mida on võimalik üldse saavutada. Näidisrakenduse koodis on iga funktsioon ning kasutatav muutuja kirjeldatud ning selle abil on ka lihtsam oma variandi loomine ja olemasolevasse äppi süsteemi rakendamine.

2.2 Rakenduse arendus

Selles peatükis kirjeldatakse rakenduse arendamist ning selle käigus tekkinud probleeme. Arenduse esimeseks etapiks on tööle saada äpp, mis võimaldaks luua kasutajat ning sisse logida. Teiseks etapiks on kõnetuvastuse funktsionaalsuse rakendamine täiendava autentimisvahendina. Kolmandaks ning viimaseks faasiks on kõnetuvastuse kohandamine antud rakenduse kontekstis.

Rakendus koosneb kolmest eri vaatest - avaleht, kus saab sisse logida, kasutaja loomise leht ning kõnetuvastust kasutav leht, mis aktiveeritakse pärast sisse logimise nupu vajutamist. Kasutaja andmete hoidmiseks kasutatakse Firebase veebipõhist andmebaasi, mis on mõeldud just mobiilsetele lahendustele ning kuna tegemist on Google enda teenusega, siis Android Studiosse rakendamine on suhteliselt lihtne. Tuleb lisada vajalikud koodiread vastavasse

[3] https://developer.android.com/studio/
[4] https://kotlinlang.org/
[5] http://www.cplusplus.com/

projekti (Joonis 4) ning äpi (Joonis 5) faili, et enda andmebaas tööle saada, kuid nende jaoks on pikemad õpetused olemas Firebase'i kodulehel.

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.2.1'
    }
}

allprojects {
    // ...
    repositories {
        // ...
        maven {
            url "https://maven.google.com"
        }
    }
}
```

Joonis 4. „build.gradle" Projekti varianti minev kood

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:15.0.0'

}

apply plugin: 'com.google.gms.google-services'
```

Joonis 5. "build.gradle" Äpi varianti minev kood

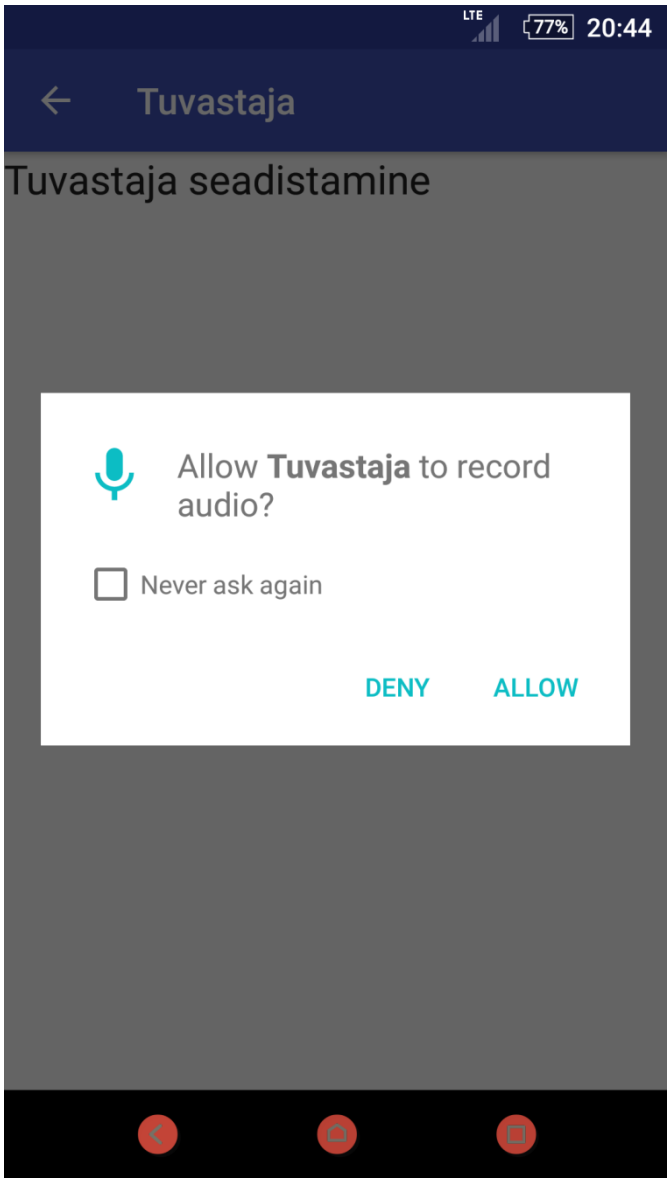
Pärast ühenduse loomist oli vaja luua rakenduse toimimiseks lehed/vaated. Esmalt on vaja avalehte, millel oleks sisselogimiseks vajalikud väljad ning koht, kust külastaja suunata kasutajat looma (Joonis 6). Sisselogimisel kontrollitakse, kas selline kasutajanimi on andmebaasis olemas. Tegevuse õnnestumisel aktiveeritakse kõnetuvastuse osa.

Joonis 6. Äpi avaleht

Järgnevalt sai loodud kasutaja loomise leht (Joonis 7). Lisaks kasutajanimele ning paroolile küsitakse ka kõnetuvastuseks kasutatavat salasõna. Kõik väljad salvestatakse andmebaasi, seda juhul, kui andmebaasist ei leita loodavatele „Kasutajanimi“ ning „Kõnetuvastuse salasõna“ väljadele vasteid. Nende esinemisel kuvab rakendus teate, et selline kasutaja või parool juba eksisteerib. Tegevuse õnnestumisel suunatakse külastaja avalehele ning tal on võimalus sisse logida.

Joonis 7. Kasutaja loomise vaade

Pärast edukat sisselogimist suunatakse kasutaja kõnetuvastuse lehele. Kõnetuvastuse lehel küsitakse esmakordsel sisenemisel luba kasutada telefoni mikrofoni (Joonis 8) ning kui luba ei anta, siis suunatakse kasutaja avalehele. Loa andmisel käivitatakse süsteem ning pärast seadistamist oodab tuvastaja süsteemi koodi kirjapandud salasõna. Rakenduse arendamisel ei jõutud lahenduseni, kus kasutaja poolt salvestatud salasõna oleks ka tema edukal sisselogimisel kõnetuvastussüsteemi poolt aktsepteeritav sõna. Seetõttu on prototüübis kõigil kasutajatel üks süsteemi poolt vastuvõetav sõna. Pikemalt räägitakse rakenduse puudustest töö peatükis 2.4, mis kirjeldab edasiarengu võimalusi. Sellega piirdus esimene etapp ning järgmisena tuli juurde lisada kõnetuvastuse funktsionaalsus.



Joonis 8. Mikrofoni luba

Arenduse esimese faasi mugavamaks tegemisel on kogemuse puudumisel soovituslik tutvuda internetist õppematerjalidega, mida on suurel kogusel saadaval. Kuna töö autoril puudus varasem kogemus äpi loomisel, siis olid need suureks abiks. Paralleele võiks tõmmata veebilehe arendamisega - on erinevad vaated/lehed, millele on lisatud elemendid. Nendeks võivad olla pildid, tekstid, nupud ja muu selline. Igal elemendil on oma unikaalne ID, mille poole saab süsteem pöörduda, et luua sellele funktsionaalsus. Näiteks „Loo kasutaja“ nupule vajutades, suunatakse kasutaja vastavale lehele.

Teise etapina oli vaja siduda kõnetuvastuse funktsionaalsus olemasolevale rakendusele, mis osutus märksa keerulisemaks, kui esimene etapp. Sphinx’i kodulehel on olemas õpetus, kuidas süsteemi rakendada, kui see on väga üldsõnaline.

Süsteemi toimimise seisukohalt on väga oluline, kus teatud failid asuvad, kuid õpetuse järgi toimides kippus tulema süsteemipoolseid veateateid, millest töö autoril vabaneda ei õnnestunud. Selle tulemusel sai otsitud abi internetist, kuid selgus, et teemakohast materjali on väga vähe. Kuigi PocketSphinx'i süsteemi rakendamisest ning tööle saamisest näiteid jagus, siis Android operatsioonisüsteemile mõeldud näidete seast leidis autor ainult ühe sobiva lahenduse[6]. See õpetus õnnestus küll tööle saada, kui siiski esines probleeme. Koodi kompileerimisel esines arenduskeskkonnas süsteemi veateateid, mille tulemusel loobuti ka selle õpetuse kasutamisest. Seejärel ei jäänud muud üle, kui kõnetuvastus ise süsteemi rakendada.

Selle probleemi lahendamiseks toetus autor oma varasematele programmeerimiskogemustele ning PocketSphinx'i Android operatsioonisüsteemile mõeldud näidisrakendusele. Põhiline probleem, mis mõlema õpetuse puhul tekkis, oli see, et kas peab lisama ainult mõne vajaliku faili või terve kausta ning kuhu see olemasoleva projekti kaustas täpselt asetada. Selle probleemi lahenduseks sai vaadeldud näidisrakenduse ülesehitust ning failide ja kaustade paiknemist. Pärast seda said vajalikud failid ja kaustad koos varasemalt mainitud internetis leitud juhendi abiga enda loodud rakendusse lisatud ning kõnetuvastus hakkas tööle (Lisa 1). Kasutaja käest mikrofoni loa küsimine käis paari koodireaga ning selle tulemusel sai rakenduse arendamise teine etapp lõpetatud.

Viimases arendusfaasis oli vaja antud hetkeks tööle saadud rakenduse kõnetuvastuse osa käesoleva töö raames loodava süsteemi jaoks kohandada. Selleks tuli eemaldada mittevajalikud muutujad ja päringud ning alles jätta funktsionaalsus, mis loob ühenduse süsteemiga ja käivitab mikrofoni, mille järel oodatakse sobivat salasõna.

2.3 Rakenduse testimine

Rakenduse testimine edenes suhteliselt kiiresti. Enamus probleeme sai juba koodi kirjutamise käigus lahendatud ning Android Studio poolt näidatud veakoodidele reageerides. Oli vaja testida, et töötaks kasutaja loomine ning sisselogimine. Kasutajat luues salvestatakse andmed andmebaasi. Lisaks kasutajanimele ning salasõnale tekitatakse ka igale kasutajale unikaalne võti, mida saaks turvalisuses aspektist ära kasutada.

[6] https://www.vladmarton.com/pocketsphinx-continuous-speech-recognition-android-tutorial/

Sisselogimisel ning kasutaja loomisel võrreldakse andmebaasis „Kasutajanime“ välja ning kontrollitakse, kas vastava nimega inimene on juba olemas. Kui andmete säilitamine andmebaasi lahenes kiirelt, siis nende kontrollimine võttis juba rohkem aega. Andmebaasi ülesehitus on küll lihtne, kuid andmete küsimiseks on vaja tekitada sündmuse kuulamine, mille käigus peab võrdlema lehel olevat välja andmebaasi sama muutuja alla salvestatud väljadega. Nimetatud probleem sai lahendatud nii, et andmebaasist loetakse kokku kõik kasutajanimed, kus väärtus võrdub külastaja poolt kirjutatuga. Kui andmebaas leiab rohkem kui null väärtust, kuvatakse veateade.

Kõnetuvastuse poole pealt vajas kontrollimist vaid salasõnast arusaamine. Kuna eelneva uuringu põhjal on teada, et antud süsteem on tundlik välistele tingimustele, siis õues ning mürarikkas keskkonnas ei saanud süsteem kohe esimesel korral hääldatust aru. Seetõttu, tuli öeldut mitu korda korrata ning proovida võimalikult selgesti hääldada, et tuvastus toimiks. Kui aga tegemist oli vaiksema kohaga, nagu näiteks siseruumid, kus puudub keskkonna müra, oli tuvastus üpriski hea ning seda isegi vaikselt rääkides või häält moonutades.

Probleeme ilmnes teiste kasutajatega testimisel. Kuna esialgse testimise viis autor läbi enda peal, siis polnud süsteemil teiste häältega kokkupuuteid. Millegipärast ei tahtnud tuvastaja väga teiste kasutajate öeldut aktsepteerida. Selleks tuli proovida mitmeid kordi ning vahel ei piisanud isegi normaalsel helikõrgusel ning selge häälduse rääkimisest. Sellise fenomeni ilmnemiset ei oska autor selgitada, süsteem peaks aktsepteerima mis tahes kõnelejat. Autor märkas, et testimiseks kasutatud seadme mällu salvestati asju, aga mida täpselt ning kas need mõjutasid tuvastuse täpsust, on töö kirjutamise ajal ainult spekulatsioion.

Testimise tulemusena võib öelda, et kasutajal tuleks proovida salasõna hääldada võimalikult selgelt ning kasutada selleks soovitatavalt vaikset keskkonda. Lisaks teeb tuvastuse raksemaks asjaolu, et tegemist on inglise keelega, mis koos halva intonatsiooni ning lohaka kõneviisiga, teeb tuvastuse süsteemile raskeks.

2.4 Võimalused äpi edasiarenguks

Eesti kontekstis oleks esimeseks edasiarenguks muuta inglisekeelne tuvastus eestikeelseks. Selleks on vaja eestikeelseid keelemudelit, mis sisaldaks keelereegleid ja akustilisi mudeleid, mille abil leitakse öeldule parim vaste. Otsimise ning parima lahenduse leidmise protsessi jaoks vajaminevad algoritmid on süsteemis olemas.

Peamiseks probleemiks ning edasiarengu objektiks töö käigus arendatud rakenduse puhul on turvalisus. Sisselogimisel kontrollitakse hetkel ainult kasutajanime olemasolu andmebaasist. See välistab olukorra, et päris igaüks sisse logida saaks. Turvalisuse aspektist võiks igal kasutajal olla isiklik kõnetuvastuse jaoks mõeldud salasõna, mis töö loomise hetkel toimib, kuid seda sõna ei rakendata kõnetuvastuses. Autori arvates oleks see probleem lahendatav, kuid töö kirjutamise hetkel jääb selle funktsionaalsuse lahendamiseks puudu nii ajast kui teadmistest. Igast individuaalsest kasutajast arusaamist on autori teadmisel vabavaraliste lahendustega võimatu lahendada.

Sellegipoolest on olemas selliseid lahendusi, kus süsteem saab aru igast kasutajast. Need rakendused kasutavad mõnevõrra teistsugust tehnoloogiat ning biomeetriat inimkõne tuvastamiseks, mille tulemusel on võimalik kasutajaid eristada. Vabavaralised lahendused kahjuks sellist tuge ei paku.

Lisaks eelnevatele probleemidele on rakenduses mõningaid väiksemaid vigu, mida tuleks lahendada. Kasutajal peaks olema võimalik oma andmeid muuta ning vajadusel oma kasutaja andmebaasist kustutada. Lisaks kasutajanimele ja paroolile peaks kasutaja käest küsima ka tema e-maili, kuhu oleks võimalik vajdusel teavitusi saata või mille abil taastada oma kasutaja, kui sinu andmeid on kuritarvitatud.

Kokkuvõte

Käesoleva bakalaureuse töö eesmärgiks oli tutvustada kõnetuvastuse taga olevat tehnoloogiat ning luua sellel põhinev autentimisrakendus.

Eesmärgi saavutamiseks anti kõigepealt ülevaade kõnetuvastuse toimimise põhimõtetest. Seejärel anti ülevaade erinevates probleemidest, mida kõnetuvastamisel ette tuleb ning millistest komponentidest peab üks süsteem koosnema. Töö esimeses peatükis kirjeldati lisaks veel kõnetuvastuseks kasutatavaid raamistike ja API'si. Töö teises peatükis arendatakse kõnetuvastust kasutav mobiilirakendus. Kirjeldatakse arenduse protsessi, rakenduse testimist ning võimalikke edasiarenguid.

Töö tulemusena valmis kõnetuvastust autentimisena kasutav mobiilirakendus, seega sai töö eesmärk täidetud. Rakenduse näol on pigem tegu prototüübiga, kuna reaalses keskkonnas see väga turvaline ei oleks. Piisab vaid kõrvalolijal salasõna kuulamast ning tal on võimalik rakendusse sisselogida, sest hetkel puuduvad vabavaralised lahendused luua süsteemi, mis aktsepteerib mitut kasutajat, kuid tuvastab igaüht eraldi. Töö oleks hea alus arendajatele, kes sooviksid ning näeksid vajadust seda edasi arendada korralikuks ning toimivaks tuvastuse viisiks. Lisaks oleks antud töö ning rakendus hea õppematerjal inimestele või ka õpilastele, kes sooviksid antud valdkonnas lisateadmisi ning proovida kõnetuvastuse rakendamist mobiilirakendusse.

Käesolevat teemat on võimalik edasi arendada ning luua rakendus, mis tuvastaks üksikut inimest, mitte lihtsalt inimkõnet. Nii saaks ära jätta hetkel kasutusel oleva kaheastmelise tuvastuse, mis teeks inimeste elu veelgi lihtsamaks. Samas jällegi tehnoloogia hetkeolukorras, kus kaheastmeline tuvastamine on üha populaarsem ning enamasti hõlmab see teenusepakkuja poolt kontrollsõnumi saatmist, oleks sõnumi asemel kõnetuvastuse kasutamine väga mugav ja kiire.

Kasutatud kirjandus

Alumäe, T. (2011). Allikas: https://phon.ioc.ee/dokuwiki/doku.php?id=konetuvastus.et

Amazon. (2018). Allikas: https://developer.amazon.com/alexa-voice-service

Center, K. (2016). Allikas: http://www.enlume.com/ios-10-speech-recognition-api-convert-voice-text/

EKT. (kuupäev puudub). *Kõnetuvastus*. Allikas: Eesti Keeletehnoloogia Riiklik Porgramm: https://www.keeletehnoloogia.ee/et/ekt-projektid/konetuvastus

Google. (2010). Allikas: Goodbye to an old friend: 1-800-GOOG-411: https://googleblog.blogspot.com.ee/2010/10/goodbye-to-old-friend-1-800-goog-411.html

Hamilton, A. (2017). Allikas: http://www.thedrum.com/opinion/2017/11/15/listen-up-the-future-voice-technology

HyClassProject. (kuupäev puudub). *Design And Implementation Of Voice Recognition System*. Allikas: http://www.hyclassproject.com/design-and-implementation-of-voice-recognition-system.html

IBM. (kuupäev puudub). Allikas: http://www-03.ibm.com/ibm/history/ibm100/images/icp/F647792J81871S75/us__en_us__ibm100__pioneering_speech__experimental_speech__900x630.jpg

Jeffs, M. (2018). Allikas: https://www.branded3.com/blog/google-voice-search-stats-growth-trends/

Neuro AI. (2013). Allikas: http://www.learnartificialneuralnetworks.com/speechrecognition.html

News-Medical. (2017). *Speech Recognition in Healthcare: a Significant Improvement or Severe Headache?* Allikas: https://www.news-medical.net/whitepaper/20170821/Speech-Recognition-in-Healthcare-a-Significant-Improvement-or-Severe-Headache.aspx

Rabiner, L. (2015). Allikas: http://ethw.org/First-Hand:The_Hidden_Markov_Model

Rabiner, L., & Juang, B. (kuupäev puudub). Allikas: Automatic Speech Recognition – A Brief History of the Technology: http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

Tuttle, T. (2015). Allikas: https://www.recode.net/2015/10/27/11620032/the-future-of-voice-whats-next-after-siri-alexa-and-ok-google

VE. (2006). *Eesti Entsüklopeedia*. Allikas: http://entsyklopeedia.ee/artikkel/k%C3%B5netuvastus

Velde, N. v. (2018). Allikas: https://www.globalme.net/blog/the-present-future-of-speech-recognition#Where_Did_We_Come_From_Where_Will_We_Go

Wikipedia. (kuupäev puudub). Allikas: Speech Recognition: https://en.wikipedia.org/wiki/Speech_recognition

Summary

The aim of this Bachelor's thesis was to introduce the technology behind speech recognition and to create an authentication application based on that.

In order to achieve the goal, an overview of the principles of speech recognition was first given. Subsequently, various problems that occured in speech recognition were examined and the components that a system must consist of were searched. The first chapter also described the frameworks and APIs for speech recognition. In the second chapter of the work a mobile application that uses voice recognition is developed and the development process, application testing and possible developments are described.

As a result of this thesis, a mobile application that uses speech recognition for authentication was developed, therefore the aim was completed. The application is more like a prototype because it would not be very secure in the real environment. It's enough for a bystander to listen to the password and log in to the application, because there are currently no open-source solutions to create a system that accepts multiple users, but identifies each one individually. This work would be a good basis for developers who would see the need to further develop it as a proper and functioning way of detecting the user. In addition, this work and application would be a good educational material for people or students who would like to have additional knowledge in this field and would like to try to apply speech recognition to mobile applications.

It is possible to continue this topic and create an application that identifies an individual, not just any human speech. This would eliminate the two-step identification currently in use, which would make people's lives even easier. Then again, in the current situation where two-step authentication is becoming more popular and, in most cases, involves sending a control message from the service provider, the use of speech recognition instead of a message would be very convenient and quick.

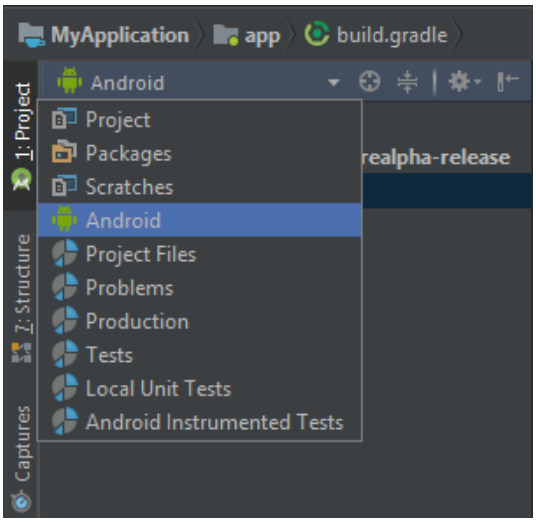
LISAD

Lisa 1. PocketSphinx'i rakendamine Android Studio'sse

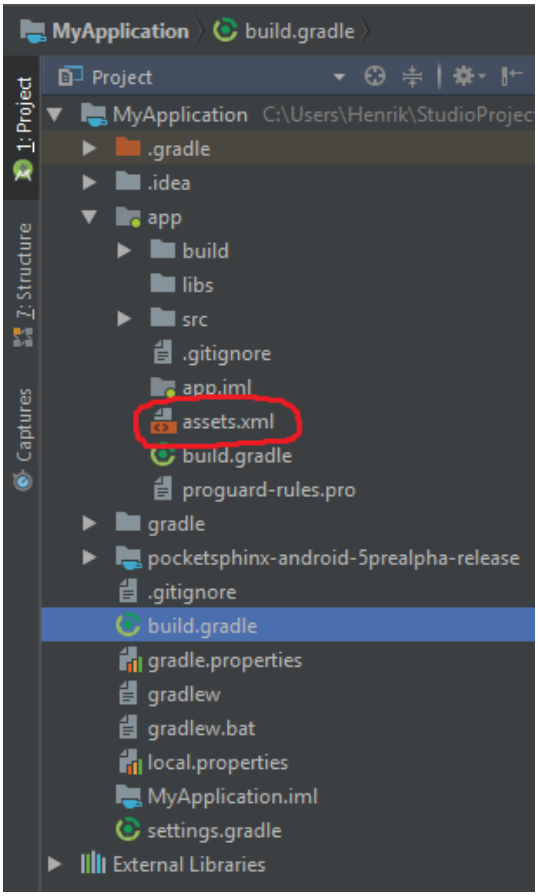
Soovitav oleks alustuseks PocketSphinx'i Android Studio näidisrakenduse repositoorium Github'st[7] alla laadida. See teeb erinevate kaustade ning failide rakendamise ja lisamise lihtsamaks ning kiiremaks. Lisaks ei ole Github'st võimalik üksikuid faile alla laadida, ainult sisu kopeerida.

1. Kui sul on juba Android Studios pooleliolev projekt, millele lisada kõnetuvastuse osa, siis leia enda jaoks sobiv Java fail, kuhu kõnetuvastuse funktsioonid hiljem lisada, aga kui see puudub, siis **loo Android Studio'sse tühi projekt(*Empty Activity*)**. Järgnev on mõlemal juhul sama.
2. **Mine Android Studiosse. Vajuta *File* > *New* > *New module* > *Import Jar/Aar Package* > *Finish*.** Vajaliku faili leiab projektis olevast kaustast nimega „aars" ning sealt fail „pocketsphinx-android-5prealpha-release.aar".
3. **Järgnevalt ava *settings.gradle* ning lisa sinna rida: „**include ':app', ':pocketsphinx-android-5prealpha-release'**"** ning vajuta „*Sync Now*".
4. **Ava *build.gradle(Module: app)* ning lisa sinna „*dependencies*" klassi: „**implementation project(':pocketsphinx-android-5prealpha-release')**"** ja vajuta „*Sync Now*".
5. **Ava *app/manifests/AndroidManifest.xml* ning lisa sinna kaks rida: „**<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <uses-permission android:name="android.permission.RECORD_AUDIO" />**".** Nende abil saab telefoni faile salvestada ning aktiveerida mikrofoni.
6. **Lisa „*assets.xml*" oma projekti.** Mine näidisrakenduses kausta „*models*" ning sealt leiad faili „*assets.xml*". Android Sudio's mine „*Android*" vaatest „*Project*"(Joonis 9) omasse. Nüüd lohista see fail „*app*" kausta kohale ja vajuta „*OK*"(Joonis 10).

[7] https://github.com/cmusphinx/pocketsphinx-android-demo



Joonis 9. Vaate muutmine



Joonis 10. „assets.xml“ lisamine

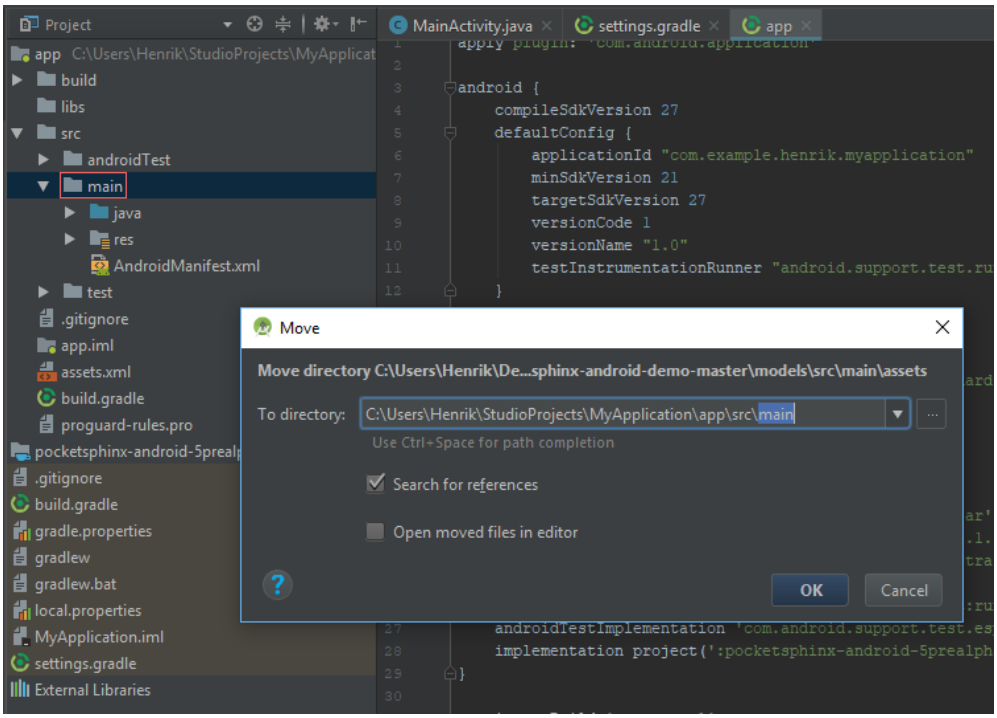
7. **Mine tagasi *build.gradle(Module: app)* ning lisa sinna täiesti lõppu read:**

   „ant.importBuild 'assets.xml'

   preBuild.dependsOn(list, checksum)

clean.dependsOn(clean_assets)**“** ning vajuta „*Sync Now*“. Järgnevalt peaks ilmuma selline veateade: „*C:\Users\Henrik\StudioProjects\MyApplication\app\src\main\assets\sync does not exist.*“ Selle lahenduseks on vaja „*sync*“ kasuta.

8. **Lisame „*Sync*“ kausta koos muu vajalikuga.** Ava Android Studio’s kaust „*app/src/main*“. Mine näidisrakenduses kausta „*models/src/main*“ ning lohista „*assets*“ kaust oma projekti(Joonis 11). Vajuta „Ctrl + F9“ ning kõik peaks olema korrras ning sellega on kõnetuvastus sinu projekti lisatud.



Joonis 11. „assets“ kausta lisamine

9. **Järgnevalt on vaja kõnetuvastuse funktsioone.** Selleks on kõige parem kopeerida näidisrakenduse Java faili sisu. Mine “*app/src/main/edu/cmu/pocketsphinx/demo*“ ning ava Java fail mõnes tekstiredaktoris. Kopeeri selles sisu enda omasse, kas siis „*MainActivity*“ või kui juba on poolik projekt, siis sobivasse Java faili. Selle tulemusel ilmub palju veateateid, kuid sellest pole katki veel midagi.
10. **Vaja on puuduvaid muutujaid.** Mine tagasi kausta „*app/src/main/res/values*“ ning lohista „*strings.xml*“ enda projektis samasse kausta ning tee sama failiga „*main.xml*“ kaustas „*app/src/main/res/layout*“. Enne projekti käivitamisst kustuta Java failis esimene rida, milleks peaks olema kopeerimisel kaasa tulnud „*package*“ ning muuda klassi nimi „*PocketSphinxActivity*“ enda Java faili nimeks(Joonis 12). Viimaks asenda „*SetupTask*“ klassis muutujanimed enda omaga, nagu eelnevad punktis(Joonis 13). Selle tulemusel ei tohik veateateid olla.

```
public class PocketSphinxActivity extends Activity implements
        RecognitionListener {
```

Joonis 12. Klassi nime muutmine

```
private static class SetupTask extends AsyncTask<Void, Void, Exception> {
    WeakReference<PocketSphinxActivity> activityReference;
    SetupTask(PocketSphinxActivity activity) {
        this.activityReference = new WeakReference<>(activity);
    }
```

Joonis 13. Klassi nime muutmine

Käivita rakendus ning oled edukalt rakendanud PocketSphinx'i oma projekti.