VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Bakalauro baigiamasis darbas

Lietuvių kalbos sintezė naudojant Idlak

Atliko:
Tadas Adomaitis parašas

Vadovas:
dr. Margarita Beniušė

Vilnius
2021

Turinys

Sutartinis terminų žodynas

ASR - Automatic Speech Recognition (Automatinis balso atpažinimas)
TTS - Text To Speech (Tekstas į kalbą)
DNN - Deep Neural Network
HMM - Hiden Markov Model
XML - Extensible Markup Language
MLSA - Mel Log Spectrum Approximation
LTS - Letter To Sound
MLPG - Maximum Likelihood Parameter Generation

Santrauka

Šiame darbe aprašoma, kas yra balso sintezė, kaip ji kuriama skirtingoms kalboms, kokia jos svarba šiuolaikiniame pasaulyje bei jos panaudojimas įvairiose technologijų srityse. Taip pat aprašomas lietuviško sintetinio balso vystymas ir naudojimas. Pagrindinis šio darbo elementas yra „Idlak" - visapusiška teksto į balsą sintezės sistema, kuri „Tangle" modulio pagalba gali sujungti parametrinės sintezės metodus kartu su gilių neuroninių tinklų mokymu, tam, kad parašytas tekstas būtų sintezuojamas į audio medžiagą. Darbe išmėginama ši sistema ir aiškinamasis jos veikimo principas bei naujos kalbos kūrimo procesas. Taip pat, pasinaudojus „Idlak" balso sintezės sistema, bandoma lietuviškos abecėlės rašmenimis parašytą tekstą paversti į sintezuotą audio failą.

Summary

Speech synthesis of Lithuanian based on Idlak

This research paper is about voice synthesis, how it is created for different languages, the impact it has in this modern era and how synthetic voice can be used in different fields of technology. It also goes through the usage and creation of a new speech synthesis module, which is used to synthesize Lithuanian language. The main point of this paper is "Idlak" - a full package "text to speech" synthesis system, which uses "Tangle" module to use the newly developed parametric synthesis methods together with deep neural networks to create speech from text input. Lastly we will be also trying to create a text to speech system witch uses text written in Lithuanian language.

Įvadas

0.1. Aktualumas

Siekiant tobulinti žmogaus ir technologijų sąveiką tobulėjo ir sintetinės kalbos technologijos. Šios kalbos atpažinimo ir teksto supratimo srities pažanga leidžia minėtoms sistemoms tapti vis didesne kasdienio gyvenimo dalimi - ji naudojama oro linijų, finansų, medicinos bei kitose svarbiose gyvenimo srityse kaip skambučių bei įmonių pagalbos centrai. Sintetinės kalbos technologija padeda ypač svarbi, nes padeda integruotis neįgaliųjų bendruomenėms, leisdama tiek tekstą įvesti nereikalaujant įvesties rankomis ar nereikalaujant tekstą skaityti. Tokios sistemos visuomenei buvo prieinamos jau prieš beveik 20 metų, pavyzdžiui kompanija „AT&T" 1992 metais pristatė balso atpažinimo sistemą, kurios klaidų lygis buvo mažesnis nei 0,5%, nuo to laiko balso atpažinimo sistemos įsitvirtino beveik visose pagrindinėse gyvenimo srityse. Tačiau, kol kitų kalbų balso atpažinimo ir balso teksto sintezės buvo plačiai naudojamos kelis dešimtmečius, reikšmingas lietuvių kalbos sintezatorių proveržis įvyko tik labai neseniai[14].

Šis darbas yra grindžiamas statistine parametrine kalbos sinteze, kuri buvo sukurta kaip perspektyvi alternatyva dominuojančiam bei pasaulyje, tiek ir lietuvoje plačiai paplitusiam vienetų parinkimo metodui. Ši alternatyva yra tobulinama dėl jos kalbos kūrimo skirtumų bei pritaikymo galimybių[9].

Atliekant šį darbą, prisidedama ne vien prie tolimesnio „Idlak" [17] sistemos vystymo ir lietuvių kalbos vartojimo palengvinimo žmonėms, kurie ateityje norėtų dirbti su „Idlak" sistema ir sintezuoti lietuvišką kalbą, bet ir prie parametrinės sintezės kalbos kūrimo, naudojant gilius neuroninius tinklus, kurių panaudojimas sintetiniam balsui yra dar naujas ir plačiai nenaudojamas.

0.2. Tyrimo objektas ir metodas

Šio tyrimo objektas yra lietuvių kalbos sintezė naudojant „Idlak" sistemą.

0.3. Darbo tikslas

Darbo tikslas - susipažinti su „Idlak" sistema, išmėginti jos galimybes, pabandyti, kaip veikia parametrinė balso sintezė ir panaudoti visa tai lietuviškais rašmenimis parašyto teksto į balsą kūrimui.

0.4. Darbo uždaviniai

Darbo uždaviniai yra:

- Išmokti naudotis ir sintezuoti balsą, naudojantis „Idlak" sistema;
- Sukurti lietuvių kalbos modelį, kurį būtų galima naudoti ir pritaikyti „Idlak" sistemos aplinkai ir funkcijoms;
- Apmokyti „TTS" balso modelį kalbėti lietuvių kalba;

1. Balso sintezė

Kalbos sintezė - tai dirbtinės, suprantamos kalbos kūrimas iš rašytinio teksto, siekiant pagerinti žmogaus ir technologijų sąveiką. Naudojantis šiuo procesu galima gautą tekstinę informaciją iš kompiuterio, telefono ar kito technologinio prietaiso paversti šnekamąja kalba. Kalba yra natūraliausias ir labiausiai paplitęs žmonių komunikavimo būdas, todėl balso sintezės procesas yra ypač svarbus, norint tobulinti ir gerinti žmogaus sąveiką su įvairiais technologiniais įrenginiais.[9]

Dėl šios priežasties kalbos sintezės sritis yra nuolatos vystoma jau daugiau nei 50 metų. Galutinis kalbos tikslas - sukurti sistemas, kurios galėtų imituoti žmogaus savybes suprasti ir generuoti kalbą, kurią būtų galima panaudoti tiek žmonių tarpusavio bendravimui, pavyzdžiui bendravimas skirtingomis kalbomis, tiek žmogaus ir kompiuterio ar kitų prietaisų sąveikai pagerinti. Paaiškėjo, kad būtent kalbos generavimo sritis buvo sunkiausia kalbos technologijų sritis, norint pasiekti kokybiškų rezultatų. Jau daugiau nei 50 metų tyrėjai bando sukurti ir imituoti fizinius kalbos generavimo procesus naudojant analoginius balso ar artikuliacinius žmogaus balso sintezės modelius. Nors buvo sutelkta daug pastangų tiriant kalbos sintezavimą, tačiau daugeliu atvejų generuojama kalba buvo nenatūrali ar tiesiog nepriimtina žmonėms, todėl nebuvo galima jos pritaikyti realioms programų sąsajoms.[21]

Pagrindiniai reikalavimai kokybiškai ir praktiškai naudojamai balso sintezei yra suprantamumas ir priimtinumas, šio rodikliai yra vertinami tiek pasitelkiant žmonių klausytojų grupes, tiek panaudojant duomenis, apdorotus statistiniais metodais. Suprantamumas yra objektyvus parametras, reiškiantis, kokią dalį lingvistinių vienetų gali suprasti klausytojas, tai yra fonemos, skiemenys ar žodžiai. Tuo tarpu priimtinumas yra subjektyvus, šiuo parametru stengiamasi nusakyti, ar sintetinio balso malonu klausytis, kiek jis primena tikrą žmogaus balsą ir ar jis bendrai yra priimtinas klausytojui. Ilgą laiką, kol buvo vystoma kalbos sintezė, didžiausias dėmesys buvo skiriamas balso suprantamumui gerinti, nes balsas, kurio negalima suprasti, yra bevertis, jo negalima pritaikyti realiose situacijose. Tik po kiek laiko, pasiekus pakankamai aukštą suprantamumo lygį, buvo pradėta skirti daugiau resursų kalbos sintezės priimtinumui vystyti.[14]

Aštuntojo dešimtmečio pabaigoje,mokslininkams išsiaiškinus, kaip patikimai iš žmogaus kalbos iškarpyti difonus, buvo sugalvota pabandyti generuoti kalbą sujungiant pagrindinius kalbos vienetus. Atradus ir ištobulinus difonų suskirstymo procesą, buvo gauta pakankamai gerai suprantama kalba, tačiau kalbos priimtinumas buvo prastas ir kalba skambėjo labai nenatūraliai, todėl jos nebuvo galima pritaikyti realiose programose. 1980 metais Yoshinori Sagisaka padarė didelį šuolį balso sintezės procese, kai suprato, kad tam tikru atveju, kai turima tūkstančiai galimų anglų kalbos difonų, galima pažodžiui susieti teisingą difonų seką, taip sukuriant natūraliai skambantį žmogaus balsą. Įgyvendint šį procesą iškilo nauja problema - kaip tiksliai nustatyti, kuris iš tūkstančių difonų turėtų būti naudojamas tam tikroje padėtyje. Tačiau, kaip ir daugelis didelio masto skaičiavimo problemų, ši problema taip pat turėjo sprendimą; per optimalų laiką ir naudojant atitinkamą kiekį resursų buvo ieškomos optimalios difonų išsidėstymo sekos. Kai nauja kalbos sintezės tyrėjų karta išnagrinėjo praktiškai visus vieneto pasirinkimo kalbos vartojimo metodus ir parodė, kad tokiu būdu galima išgauti gerai suprantamą ir kokybišką kalbą, ją buvo galima pradėti naudoti ir įvairiose kompiuterinėse programose. Kai išspręsta natūraliai skambančios kalbos problema, buvo pradėta spręsti kitas su rašytinio teksto skaitymu susijusias problemas, kaip kalbos garsumo nustatymas, kirčiavimas, pauzės ir kiti kalbos aspektai [21].

Kitas pastaruoju metu išpoluliarėjas metodas, kuris skiriasi nuo sujungimo arba vienetų parinkimo sintezės metodų. Tai parametrinės sintezės metodas. Atliekant parametrinę kalbos sintezę, jos išvestis yra paremta galutiniu parametrų skaičiumi. Tuo tarpu sujungimo sintezėje yra nau-

dojamas iš anksto apibrėžtas žodžių rinkinys. Pagrindinė vieneto parinkimo sintezės problema yra harmonikų parinkimas, tuo tarpu parametrinėje sintezėje harmonikų dažnių ilgis yra žinomas, todėl signalo išplėtimas į harmonikas gali būti atliekamas užtikrintai [15].

Jau kurį laiką statistikos parametrų technikos, kaip Hiden Markov Model (HMM), yra aktuali tema. Taikant šią techniką, kalbos sintezatoriai gali sukurti suprantamus ir labai lanksčiai naudojamus balsus. Tačiau šis metodas dar nesugeba aplenkti vienetų parinkimo metodo savo kalbos kokybe, todėl parametrinės sintezės tyrėjai ir toliau bando sukurti modelius, kurie sugebėtų kuo geriau imituoti žmogaus balsą. Sintezatoriai, naudojantys vienetų parinkimo metodą, yra žymiai labiau suprantami ir priimtinesni žmonėms. Šio balso suprantamumas, kaip ir buvo minėta, labiausiai priklauso nuo garsų bazės dydžio - kuo didesnė garso duomenų bazė, tuo kalba tampa suprantamesnė ir kokybiškesnė. Vienetų parinkimo kalbos modeliuose, formuojant naujus sakinius, garso bangos yra tvarkomos tiesiogiai sujungiant garso bangų segmentus. Tuo tarpu šiuo metu generuojama statistinės parametrinės sintezės kalba yra pagrįsta matematiniu modeliu, kuris gali būti apmokomas, kaip atkurti kalbos parametrus, kaip, pavyzdžiui, naudojimui kartu su HMM [9].

Šiuolaikinėje parametrinės teksto į balsą sintezės tyrimų eroje daugiausia dėmesio tyrėjai skiria gilaus mašininio mokymo algoritmams vystyti ir jų algoritmų pritaikymui sintezėje. Taip atsirado nauja galimybė kurti naujus balsus, kai išlaikomi originalūs parametrai ir naudojami nauji matematiniai modeliai, kurie padeda dar geriau atkurti realistišką žmogaus balsą [9].

Nors matematiniai modeliai ir metodai yra labai svarbūs norint išgauti realistišką žmogaus balsą, tačiau, kaip teigia daugelis tyrėjų, diktoriaus pasirinkimas gali taip pat nulemti nemažą dalį balso kokybės sintezės procese. Kadangi balso sintezės darbas yra atkurti diktoriaus įrašytą balsą, todėl reikėtų atkreipti daug dėmesio ne vien į matematinių modelių ar metodų kūrimą, bet ir į diktoriaus pasirinkimą ir jo balso įrašymą [14].

1.1. Lietuvių kalbos balso sintezė

Pasaulyje populiariausiomis kalbomis parašytą tekstą sintezuotu balsu išmokta skaityti jau prieš kelis dešimtmečius, lietuvių kalba skaitomas sintezuotas tekstas atsirado ne ką vėliau, tačiau ilgą laiką lietuvių kalba sintezuoti balsai skambėjo nenatūraliai, todėl nebuvo plačiai naudojami. Jie buvo pritaikyti tik siauram ratui naudotojų - Lietuvos aklųjų ir silpnaregių bendruomenėms. Tačiau paskutiniu metu lietuvių kalbos sintezė iš teksto ėmė sparčiai tobulėti; vien tik 2013-2015 metais pasirodė šeši nauji visiškai sintetiniai balsai [14].

1994 metais lietuvių kalba buvo įtraukta ir į Didžiosios Britanijos „Dolphin" kompanijos „Apollo" sintezatorių greta kitų pasaulio kalbų. Tai buvo pirmasis lietuviško balso sintezatorius akliesiems ir silpnaregiams, naudojant „MS-DOS" sistemą. Ši sistema išsiskyrė tuo, kad buvo pagaminta kaip aparatūrinis įrenginys, kai dauguma lietuviškų sintezatorių buvo kuriami kaip programiniai moduliai [14].

1996 metais VU Filosofijos fakulteto buvo sukurtas sintezatorius „Aistis". Šis sintezatorius naudojo apie 3000-ių kamienų žodyną, kirčiavimo taisykles, bei 480 įvairaus ilgio segmentų balso įrašus, norint sukurta programine įranga išgauti lietuvių kalba kalbantį sintetinį balsą, kuris generuojamas jungiant nemodifikuotus signalo segmentus. 2003 metais Čekijos kompanija „RosaSOFT" į sintezatorių „WinTalker" įtraukė ir tris lietuviškus balsus. Ši sistema veikė „Windows" aplinkoje ir buvo sukurta 2000 metais. 2008 metais buvo sukurtas sintezatorius, kalbantis lietuvišku vyrišku balsu. UAB „Etalinkas" ir „Sakrament" sintezatorius buvo pavadintas Egidijaus vardu. Sintezatorius "SINT.AS", galintis kalbėti dviem, moterišku ir vyrišku, balsais, buvo su-

kurtas Pijaus Kasparaičio kartu su UAB „Algoritmų sistemos". Tai buvo pirmasis sintezatorius, galintis kalbėti lietuvišku moteriškų balsu. Vyriškas ir moteriškas balsai buvo pavadinti Laimos ir Marijaus vardais. Šis sintezatorius nuo kitų skyrėsi tuo, kad jam buvo specialiai parinkti diktoriai ir sintezei buvo panaudotas vienetų parinkimo metodas. 2013–2015 metais Vilniaus universitetas kartu su partneriais vykdė projektą „LIEPA". Vykdant šį projektą buvo sukurti keturi balsai, kurie buvo parinkti tam, kad kuo labiau skirtųsi vienas nuo kito. Balsai buvo sudaryti iš vyriško ir moteriško jaunatviškų balsų, bei vyresnio amžiaus vyriško ir moteriško balsų. Balsai buvo „Aistė", „Edvardas", „Regina" ir „Vladas". Galima paminėti, kad šio darbo praktinėje dalyje ir yra naudojamas šiam projektui sukurtas balsas „Regina". Šiai sintezei taip pat buvo naudojamas vienetų parinkimo metodas ir jam reikalinga kiekvieno diktoriaus didelio dydžio balso duomenų bazės. Taip pat įdomų, kad visi keturi balsai yra naudojami „RoboBraille" paslaugoje, kurioje elektroninio laiško tekstas, nusiųstas į vieną iš tam skirtų elektroninių pašto dėžučių, kaip atsakymą gražiną susintezuotą siuntėjo tekstą, tam tikru, pagal elektroninį paštą pasirinktu balsu. Šios sintezatoriaus suprantamiausi balsai buvo „Regina" bei „Edvardas", todėl šiuo balsus geriausia naudoti norint išgauti suprantamiausią lietuvių kalbos sintetinį balsą[14].

2. Idlak sistema

Balso sintezėje teksto analizavimas, norint nustatyti tarimo ir kalbos struktūrą, dažniausiai vadinamas front-end procesu. Procesas, kuris iš analizuojamo teksto generuoja garso bangas, vadinamas back-end. Hidden Markov modeliu (HMM) - paremtas sintezavimas rodo didesnį efektyvumą sintezuojant kalbą. „HTS" [12] - Hidden Markov modelio kalbos sintezavimo sistema yra dominuojanti atviro kodo sistema, kuri buvo sukurta „Tokuda" grupės „Nagoya" įmonėje daugiau nei prieš dešimtmetį. Kadangi ši sistema vystoma jau ilgą laiką, ji sugeba generuoti vienus geriausių rezultatų balso sintezėje, paremtoje parametriniais modeliais. „HTS" yra back-end sistema, kuri yra priklausoma nuo trečių šalių sistemų, tam, kad būtų galima paruošti pradinę proceso dalį. Nors didesnė dalis resursų buvo naudojama back-end parametrinei balso sintezei tobulinti, palyginti mažai darbo buvo skirta front-end duomenų paruošimui. Tačiau „HTS" turi griežtus licencijos apribojimus, dėl kurių ne visi norintieji gali išbandyti ir dirbti su šia parametrinės sintezės sistema.

„Kaldi" [13] - tai laisvos licencijos automatinio kalbos atpažinimo (ASR) įrankių rinkinys, kuriuo remiantis yra sukurtas ir „Idlak" projektas. „Idlak" yra sistema, skirta kurti visapusišką parametrinę sintezę, naudojant Kaldi sistemą. „Idlak Tangle" yra laisvai licecijuotos, atviro kodo kalbos atpažinimo sistemos TTS plėtinys. „Idlak Tangle" kalbos modeliavimui yra naudojamas naujausias gilių neuronų tinklo (DNN) metodas, „Idlak" XML struktūros paremtas teksto apdorojimas ir naujai išleisto atviro kodo „MLSA" vokoderis (angl. vocoder). Sistema yra sukurta atsižvelgiant į dinamišką dizainą, modernų kodo rašymą, testavimą ir XML failų formato naudojimą pagrindinei sąsajai. Ši sistema siekia supaprastinti parametrinės sintezės procesą tuo pat metu generuojant aukšto lygio rezultatus. „Kaldi" yra paremta didžiąja dalimi komandinių eilučių programų, kurios gali apdoroti grupes įvairių failų rūšių. Idlak veikia panašiu principu, tačiau taip pat įveda naują XML failų formatą. „Idlak" sistemą sudaro daug komponentų; teksto apdorojimo moduliai, teksto apdorojimo duombazės ir balso kūrimo sistema. „Idlak" turi paprastą ir daug leidžiančią licenziją, artimą ryšį su aukščiausio lygio automatinio balso atpažinimo technikomis, nes ji yra dalis "Kaldi" paketo. Taip pat ši sistema yra puikus įrankis kuriant kalbos sistemas nuo pradžios iki galo, generuojant ir sintezuojant parametrinius TTS balsus. Idlak TTS-DNN sistema yra pavadinta „Tangle" [17]. „Idlak" yra ne vienintelė sistema, kuri turi laisvo naudojimo lincecią, į ją panaši sistema vadinasi „Merlin toolkit". „Merlin" [10] taip pat yra įrankių rinkinys galintis generuoti statistinę parametrinę kalbą, naudojantis gilaus neuroninių tinklų (DNN) modeliais. Tačiau kitaip nei „Idlak", „Merlin toolkit" nėra visapusė programa ir turi būti naudojama kartu su kita vartotojo sąsajos sistema, kaip, pavyzdžiui, „Festival" ir jai taip pat reikia pritaikyti vokoderis [10]. Dėl šios priežasties „Idlak" yra patogesnė sistema, norint kuo patogiau modeliuoti kalbą, nesirūpinant pašaliniais įrankiais. „Tangle" sujungia „Idlak" vartotojo sąsają ir „MLSA" vokoderį kartu su DNN modeliavimu, naudojant vienetų ilgio ir akustinius parametrus, duodant pilnai funkcionuojančią visapusę TTS sistemą [17].

Geros kokybės parametrinei TTS sistemai reikia aukštos kokybės vokoderio; tačiau dauguma esamų vokoderių yra arba žemos kokybės, arba dėl licencijavimo apribojimų negali būti naudojami atviro kodo projektuose. Įdiegtos pažangiausias vokoderio technikos į „Kaldi", padeda panaikinti šią spragą ir siūlo pirmąją nemokamą aukštos kokybės parametrinę teksto į kalbą sistemą.

Naudojant leksiką, LTS (angl. letter to sound) taisykles ir tekstą, galima sugeneruoti įrašo scenarijų. Pirmiausia jis turėtų turėti pilną fonetikos aprašymą su daug skirtingų variantų, žodžių ar frazių pradines ir galutines pozicijas. Antra, sakiniai turi būti ne per ilgi ir lengvai skaitomi ir, galiausiai, jie turi būti ta pačia kalba, kaip ir visas balso ir teksto įrašas. Vienas labai svarbus įrašų scenarijaus reikalavimas yra tai, kad jis turi būti visiškai normalizuotas. Pavyzdžiui, visi skaičiai

turi būti užrašyti žodžiais ir didžiosiomis raidėmis. Taip pat yra rekomenduojama, kad visi įrašyti žodžiai būtų taip pat įrašyti ir į leksikoną, net jei ir LTS taisyklės juos gali teisingai numatyti. Labai svarbu visus šiuos įrašus patikrinti rankiniu būtu, kad būtų išvengta klaidų generuojant reikiamus failus. „General American English" pavyzdys gerai atvaizduoja šią normalizaciją:

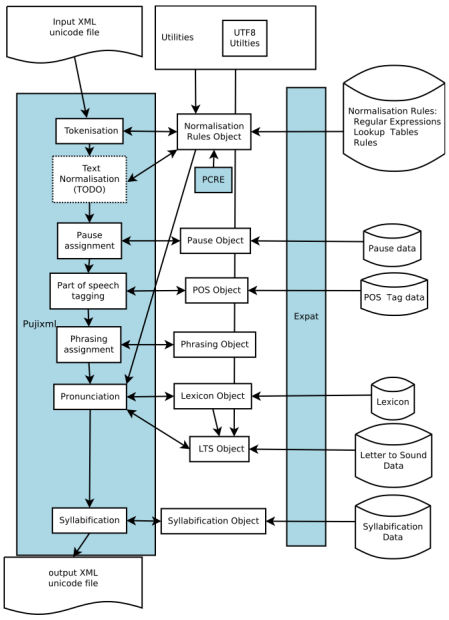
```
1 <fileid id="z0001\_015">
2 To eight thirty P M.
3 < /fileid >
```

Taip pat, kuriant naujus scenarijus, balsus ar kalbas svarbu teisingai identifikuoti įvairius failus ir tekstus, norint, kad visa sistema laikytųsi tų pačių taisyklių ir būtų kuo lengviau suprantama visiems. Taigi, įrašymo scenarijui failo identifikacija yra pateikiama kartu su „fieldid" lauku. Pagal „fieldid" lauką galima identifikuoti, kuriam audio failui prikauso tekstas. „Fieldid" laikosi vienodo modelio: pirmiausia mažoji raidė parodanti šnekančio žmogaus žanrą, kalbėjimo stilių, tikslą ar šaltinio tipą. Pavyzdžiui, raidė *z* yra naudojama fonetikai, *q* reiškia klausimus ir *n* reiškia, kad atitinkamas turinys buvo gautas iš naujienų kanalų. Tada seka keturių simbolių skaičius, kurio pradžia prasideda nuliais ir kuris nurodo tam tikrą šnekėjimo žanrą. Tai gali būti pritaikoma nurodant tam tikras pastraipas, kurios yra naudojamos ar nuorodas į atskirus naujienų straipsnius. Žymėjimo pabaigoje yra trijų skaitmenų eilė, kuri užpildoma nuliais, ji nurodo kiekvieną naudojamą sakinį. Tarkime z0001_001 yra pirmas sakinys nurodytame žanre.

Atrinkti audio įrašus jau pagamintam tekstiniam sakinių failui nėra labai sudėtinga, reikia tik įsitikinti, kad įrašytas audio sakinys atitinka įrašyto tekstinio sakinio „fieldid" pavadinimą. Idealiai audio įrašas yra išgaunamas profesionalioje įrašų studijoje, kurioje yra kuo mažiau pašalinių garsų ir aido. Taip pat įrašant balsą, nereikėtų naudoti garso kompresijos. Kiekvienam šnekančiajam yra priskiriamas triženklis kodas. Kiekvienas audio failas turėtų turėti šį kodą ir po jo esantį pabraukimo ženklą ir „fieldid" numerį. Pavyzdžiui, jeigu šnekančiojo pavadinimas yra „abw", tada audio failas turėtų vadintis abw_z0001_001.wav.

Kartais audio failai ne visiškai sutampa su tekstu, dažniausiai taip nutinka, kai skaitantis diktorius ne iki galo ištaria tam tikrą žodį. Jei vienas žodis sakinyje buvo neteisingai ištartas, jį galima pataisyti kalbėtojui skirtame sakinių scenarijaus faile naudojant „lex XML" žymą. Pavyzdžiui, jeigu kalbantis žmogus neištarė sakinyje paskutinės raidės *t* žodyje „bright", jį galima pataisyti įrašius <lex phon = "b r ay1"> bright </lex> Reikia atkreipti dėmesį, kad tokie sprendimai gali būti praktiškai nepritaikomi kitiems sudėtingesniems tarimo atvejams, todėl, jeigu iškilo sunkesnė problema, kalbėtojui reikia sukurti kalbėtojo leksikos kopiją ir ten pakeisti blogą žodį [1].

Teisingai sutvarkius naujus audio ir tekstinius failus „Idlak" front-end analizuoja ir normalizuoja sukurtą tekstą, tada iš jo sukuria tvarkingą ir pilną fonetinį ir kontekstinį vaizdą, dar žinoma kaip "full labels". Kiekvienas „Idlak" teksto apdorojimo modulis veikia naudojant XML pavidalą. Kiekvienas modulis prideda struktūros į XML failą ir gali priklausyti nuo ankstesnių modulių pridėtų duomenų. Pavyzdžiui, „syllibify" modulis reikalauja kito tarimo modulio sukurto tarimo. Moduliai gali būti sudėti į komandinės eilutės dvejetainių failų eiles, kuriuose naudojamos „Kaldi" stiliaus parinktys. Šiuo metu Idlak sistemoje yra dvi komandinės eilutės programos: „idlaktxp", kuri atlieka teksto apdorojimą, ir „idlakcex", kuri perima išvestį iš „idlaktxp" ir prideda kontekstinių modelių pavadinimus „Kaldi" arba „HTS" formatu. 1 paveikslėlyje parodyti dabartiniai moduliai, kurie sudaro „idlaktxp"[17].



1 pav. idlaktxp diagrama
[17]

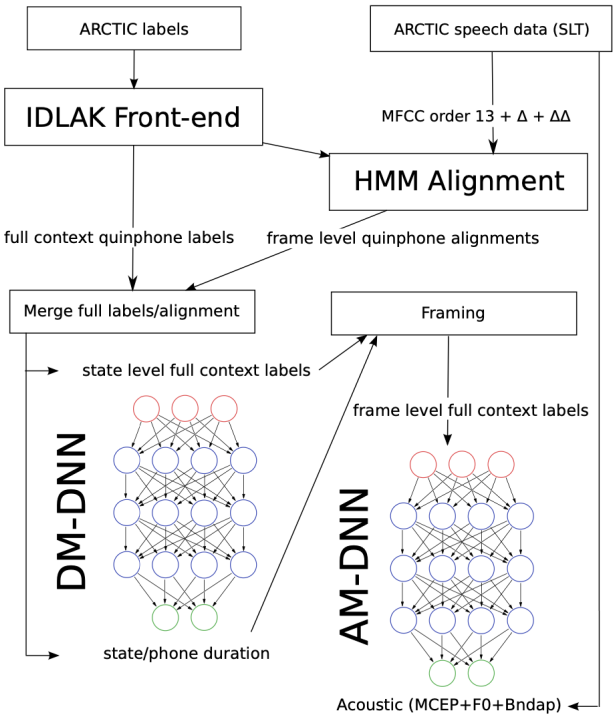
Paruošus audio ir tekstinius failus ir juos apdorojus galima pradėti mokymo procesą. Naudojant „Tangle" yra apmokomi du giliūs neuroniniai tinklai: „Duration Model DNN“ (DM-DNN), kuris pagal įvesties fonetikos žymas nuspės fonemų ir „HMM" ilgį, ir „Acoustic Model DNN“ (AM-DNN), kuris numatys akustinę duomenų seką.

Kadangi „Idlak" sistema yra labiau skirta naudoti, kaip lengva bazinė sistema, „DNN" buvo pritaikytas naudojant palyginti nedideliam skaičiui (3) paslėptų sluoksnių ir mazgų (atitinkamai 100 ir 700, priklausomai nuo akustinio modelio trukmės) kiekviename sluoksnyje. Visuose sluoksniuose yra afininis komponentas, po kurio seka sigmoidinė aktyvavimo funkcija.

Išvesties duomenys (trukmės arba akustiniai) yra normalizuoti taip, kad kiekvieno išvesties komponento vertės būtų nuo 0,01 iki 0,99. Skirtingai nuo kitų būdų (pvz., Zen ar Qian), iš mokymo nėra ištrinami tylūs kadrai, nes buvo nustatyta, kad jų panaikinimas nedarė jokių pokyčių sintezės kokybei. Apmokymo procedūra standartinė - naudojant stochastinį gradientinį nusileidimą, pagrįstą galiniu plitimu. Minimalizavimo kriterijus nustatomas pagal vidutinę kvadratinę paklaidą *(angl. mean square error (MSE))*. Mokymai vykdomi naudojant mokymo rinkinį, o kryžminiam patvirtinimui naudojamas atrinktas pavyzdinis rinkinys.

Priverstinio lyginimo procedūra atliekama visoje duomenų bazėje ir yra naudojama norint visas konteksto žymas sulyginti su akustiniais duomenimis, naudojant standartinius įrankius iš „Kaldi“ rinkinio [17].

Sintezės procedūra veikia taip: „Idlak“ sąsaja paverčia tekstą „pilnomis žymomis“, fonetiniu ir kontekstiniu jo vaizdavimu, tada visos žymos transformuojamos į įvestį, tinkamą DNN, pakeičiant visas teksto detales skaitmeninėmis reikšmėmis. Kiekvienos fonemos ilgis ir kiekvieno HMM trukmės būsena yra prognozuojama naudojant trukmės modelį DNN, o akustinio DNN modelio įvestis yra generuojama sukuriant žymų dalis kiekvienai norimai akustinei daliai. Tai reiškia, kad sintezės metu „pilnos žymos", sukurtos kartu su įvesties tekstu yra sugeneruojamos „Idlak“ ir konvertuojamos į skaitines vertes. Išvesties trukmės yra apskaičiuojamos naudojant DM-DNN. Vėliau šios vertės nuosekliai apdorojamos, kad būsenų trukmės suma būtų lygi fonemos trukmei. Sujungę įvesties žymas ir trukmę galime duomenis panaudoti akustiniam DNN modeliui [17].



2 pav. DNN diagrama
[17]

Kiekybinės fonemos ir „HMM" pozicijos būsenos pridedamos prie įvesties žymų. Tai sukuria akustinių ypatybių sekas su jų dariniais. Šios sekos yra apdorojamos naudojant MLPG algoritmą, kad būtų sukurta sklandi akustinių parametrų seka. Tada šios funkcijos perduodamos į „MLSA" sintezatorių, kad būtų sukurta garso išvestis.

Idlak yra sukurtos sistemos, reikalingos apdoroti ir sukurti naujas kalbas, akcentus ir sutvarkyti audio failus. Tačiau kitas didelis žingsnis sistemos tobulinime - sukurti patogią naudotojų sąsają, kad kuo daugiau žmonių galėtų prisidėti prie duomenų kūrimo ir tobulinimo, kuris dažniausiai priklauso nuo naujų balso įrašų ir leksikos failų tobulinimo [17].

2.1. Hidden Markov Model

Atliekant sintezę, kai naudojamas vieneto pasirinkimo metodas, duomenų bazėje yra saugomi keli kiekvienos fonemos egzemplioriai skirtinguose kontekstuose. Tokią duomenų bazę sukurti užtrunka labai ilgai, be to reikalingas didelis kiekis laisvos atminties diske. Naudojant tokią duomenų bazę atsiranda suvaržymai naudoti tik jau įrašytus garsus. Tai gali būti išsprendžiama naudojant jau minėtą parametrinės sintezės techniką. Naudojant parametrinę sintezę yra reikalaujama mažiau atminties, kurios reikia išsaugoti parametrus, nei saugant atskirtus garso failus, taip pat atsiranda galimybė sugeneruoti daugiau skirtingų garso variacijų.

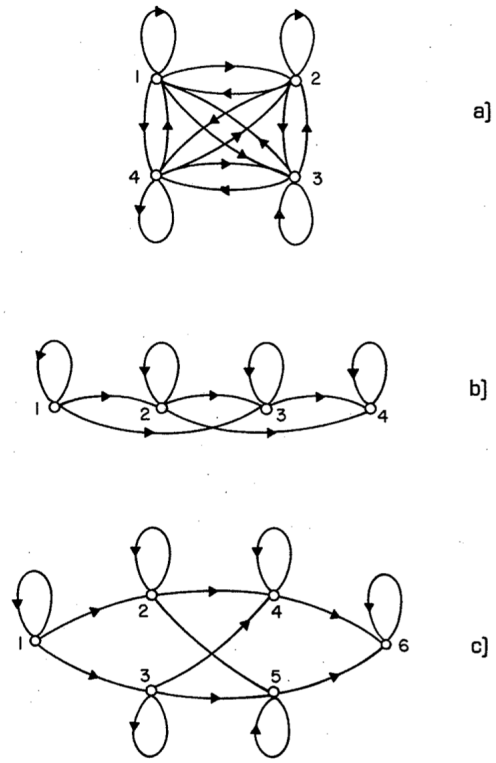
Viena plačiausiai naudojamų parametrinės sintezės technikų yra Hidden Markov modelis. Šis modelis sudarytas iš dviejų fazių - treniravimo fazės ir sintezavimo fazės. Treniravimo fazėje reikia nurodyti, kokias savybes naudojant modelis turėtų būti mokomas. „Mel" dažnio ceptralo funkcijos keoficientai su jų pirmais ir antrais dariniais yra dažniausiai mokymo fazėje naudojami tipai. „Mel" - tai išvestinė klausos skalė, susiejanti suvokiamą dažnį fmel (matuojama mel) ir tikrąjį fizinį dažnį f[22]. Ši dažnio skalė yra tiesinis dažnių intervalas, mažesnis nei 1000Hz ir logaritminis atstumas

didesnis už 1000Hz. Kaip atskaitos taškas naudojamas 1 kHz dažnio aukštis, 40 dB viršijantis girdima klausos garsą, jis apibrėžiamas kaip 1000 mel[11]. Minėtos funkcijos kiekviename žingsnyje sudedamos į vektorinę funkciją. „Baum-Welch" algoritmas yra panaudojamas, kai iš vektorinių funkcijų generuojamos fonemos. Šis modelis dažiniausiai susideda iš fonemos pradžios, vidurio ir pabaigos. Sintezavimo fazė susideda dar iš dviejų etapų; pirmiausia įvertinamas fonemų sekos požymių vektorius, tada naudojamas filtras, kuris konvertuoja vektorines formas į audio signalus.

Garso kokybė naudojant „HMM" nėra tokia gera, kaip naudojant vienetų rinkimo sintezę, tačiau yra galimybė ją pagerinti. Tai įgyvendinama naudojant hidden semi-Markov modelius (HSMM), trajektorinius „HMM" ir stochastinius Markov grafikus [19].

Žemiau esančiame paveikslėlyje (3 pav.) yra trijų skirtingų Markovo modelių iliustracija.

(A) modelis yra vadinamas erodiniu arba neabsorbuojančiu modeliu. Šiame modelyje galima pasiekti bet kurią būseną iš bet kurios kitos būsenos. (B) modelis vadinamas "iš kairės į dešinę", jis naudojamas skirtingo ilgio signalams modeliuoti, todėl yra naudojamas ir balso sintezėje. Šiame modelyje žemesnioji būsena visada būna aukščiau už aukštesniąją būseną. (C) modelis yra lygiagretus ir eina iš kairės į dešinę, šis modelis turi kelis kelius per būsenas [18].



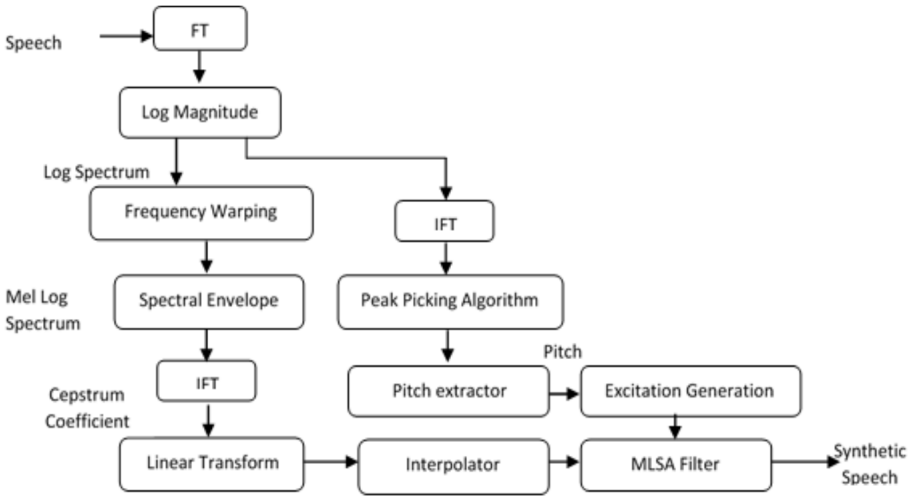
3 pav. trijų skirtingų HMM iliustracija
[18]

2.2. „MLSA" vokoderis

Vokoderis yra priemonė naudojama signalų apdorojimo ir tyrimų srityje, skirta kalbos analizei ir sintezei. Vienas pagrindinių kalbos vokoderio privalumų yra tai, kad jis leidžia sustiprinti, modifikuoti ir į kalbos signalą pervesti atskirtus segmentinius ir viršsegmentinius parametrus. Analizuojami parametrai naudojami balso atpažinimui, kalbos atpažinimui ir balso emocijų atpažinimui. Modifikuotos šių sistemų variacijos yra naudojamos daugelyje aplikacijų, pavyzdžiui, kalbos kūrimo, kalbos gerinimo, kalbančiojo balso modifikacijoms ir balso konvertavimui. Kalbos signalas

turi akustinę ir lingvistinę informaciją. Kalba, dialektas, fonemų tarimas ir kiti su kalbančiuoju susiję parametrai yra priklausomi nuo lingvistinės informacijos. Akustiniai parametrai naudojami vaizduoti ir kurti fizinę žmogaus kalbą.

„MLSA" vokoderis naudojamas cepstrumo sintezei pagal „Mel" skalę. Šis vocoderis yra pranašesnis, nes turi žemą jautrumo koeficientą ir patobulintą garso signalo tvarkymo koeficientą. Garso aukščio parametras (F0) yra gaunamas naudojant aukščio pasirinkimo algoritmą randant aukščiausią dažnį. (4 pav.) pavaizduota „MCEP-MLSA" pagrindu sukurtas vokoderis. Atliekant analizę „MCEP" ir pagrindinis dažnis (F0) gaunamas kas 15 ms. Kaip ir nurodyta, dažnių deformavimo koeficinetas yra paimamas pagal filtro tvarką ir kvantavimo plotį (0,25). Sintezavimo žingsnyje „MLSA" filtas duoda labai tikslias aproksimacijas su trečios eilės modifikuotu Pade aproksimavimu (0,024 (0,2 dB)). „MCEP-MLSA" vokoderis sukuria tos pačios kokybės kalbą sintezuojant tik 60-70 % duomenų, lyginant su paprastais „kepstro" vokoderiais [2].



4 pav. „MLSA" vokoderio diagrama
[2]

3. Darbo aplinka

3.1. Darbo aplinkos specifikacijos

Įrenginio, naudoto „Idlak" sistemos darbui ir kalbos modelio mokymui, specifikacijos:

- CPU: 6-core AMD Ryzen 5 2600X
- GPU: Nvidia GeForce GTX 1060 6 GB
- Desktop: Cinnamon 5.6.7
- Distro: Linux mint 20 Ulyana
- Base: Ubuntu 20.04 focal

3.2. Darbo aplinkos paruošimas

Kadangi „Idlak" sistema yra sukurta naudoti „Linux" aplinkoje, norint pradėti darbą, savo kompiuteryje reikėjo įdiegti „Linux" operacinę sistemą. Darbui atlikti buvo pasirinkta „Linux Mint" sistema, nes ji yra pakankamai paprasta ir nereikalauja daug papildomo darbo, norint pradėti greitai ja naudotis. Kompiuteris, kuris turėjo būti naudojamas darbui atlikti, naudojo „Windows 10" sistemą, todėl buvo nuspręsta įdiegti „Linux Mint", kaip antrą operacinę sistemą įrenginyje. Šiam darbui atlikti reikėjo parsisiųsti „Linux Mint" operacinės sistemos įdiegimo failą, jį įrašyti į „usb" laikmeną ir, naudojant sukurtą „usb" įdiegimo raktą, įdėti į kompiuterį ir suinstaliuoti. Prieš įdiegiant naują operacinę sistemą reikėjo paskirti vietos kietajame diske, kurioje bus įdiegiami ir laikomi reikalingi operacinės sistemos failai. Naudojamas kompiuteris turėjo 2 terabaitų dydžio kietąjį diską, iš kurių apie 150 GB vietos buvo paskirta „Linux Mint" operacinei sistemai. 100 GB vietos buvo paskirta „Linux" operacinės sistemos duomenims saugoti, 30 GB skirta „root" direktorijai, kurioje saugomi visi pagrindiniai sistemos failai, o likusi vieta skirta operacinės sistemos paleidimui. Programinės sistemos įdiegimas nebuvo sudėtingas, o įdiegus ir atnaujinus sistemą buvo galima pradėti iš karto ruoštis darbui su „Idlak" balso sintezės sistema.

3.3. Idlak sistemos paruošimas

Norint pradėti darbą su „Idlak" sistema primiausia reikėjo nusikopijuoti jos failų rinkinį iš „Github" saugyklos [20]. Kai visi „Idlak" failai buvo gauti, norint pradėti sintezės darbus reikėjo patikrinti ir sudiegti visus reikiamus sisteminius priedus. Įėjus į „Idlak" pagrindinę direktoriją, atidaromas „README" failas, kuriame nurodyti žingsniai, kaip pasiruošti darbui. Pirmiausia reikėjo pasirinkti „tools" direktoriją ir sekti tolimesnes instrukcijas. Paleidimo instrukcijos nurodė paleisti reikiamą komandinę eilutę ir pažiūrėti, ar sistemoje netrūksta tam tikrų, „Idlak" sistemai reikalingų, pagalbinių sistemų. Jeigu ne visos sistemos yra randamos Linux aplinkoje, jas reikia papildomai įdiegti rankiniu būdu. Tolimesnis žingsnis - paleisti „make" komandą, kuris įdiegia dar daugiau įvairių įrankių, reikalingų darbui su „Idlak" sistema. Sėkmingai viską įdiegus „tools" direktorijoje, pereinama į „src" direktoriją ir tęsiami pasiruošimo darbai. Pirmiausia „scr" direktorijoje reikėjo paleisti „./configure" komandą, tačiau kilo problemų, nes norint įvykdyti „./configure" komandą reikėjo papildomai sudiegti „python-dev" ir „python3-dev" programas. Kadangi dauguma „Idlak" modulių yra parašyti „python" kalba, šis žingsnis buvo būtinas. Norint naudoti Idlak sistemą, taip

pat reikėjo įdiegti senesnę „GCC" versiją, nes sistemoje egzistuojanti versija, kuri įdiegta su Linux operacine sistema, buvo per nauja ir nepalaikoma kitų sisteminių modulių, kurie buvo naudojami „Idlak" sistemoje. „Configure" komanda reikalinga paruošti programų įdiegimą esamai sistemai, pagal jos specifinius reikalavimus. Ji patikrina, ar sistemoje yra visi reikiami moduliai tam, kad būtų galima toliau įdiegti reikiamas programas, taip pat užtikrina, kad reikalingos programos galėtų naudotis sistemoje jau esančiais moduliais. Dauguma „unix" programų yra parašytos C kalba ir reikalauja C kompiliavimo programos tam darbui atlikti. Taigi, „configure" komanda įsitikina, ar sistema tikrai gali kompiliuoti C kalba parašytą kodą ir suranda, kur sistemoje egzistuoja C kalbos kompiliavimo modulis.

Sėkmingai įvykdžius „configure" komandą, toliau paleista „Makedepend" komanda, kuri suranda visas priklausomybes naudojamas C programų kode ir jas visas patalpina „makefile" faile paprastesniam ir greitesniam make komandos veikimui. Sėkmingai pasibaigus „makedepend" užduočiai, galima paleisti „make" komandą. Ši komanda naudojasi „configure" komandos surastais duomenimis apie esamos operacinės sistemos specifikacijas ir jomis naudojantis gali tvarkingai įdiegti visas reikiamas programas, kurios yra „makefile" direktorijoje.

Baigus pagrindinius sistemos diegimus, papildomai reikėjo įdiegti keletą paketų, norint išmėginti kalbos apmokymą „Idlak" sistema. Tam buvo įdiegta papildomų „python" paketų, kaip „pythonę-lxml" ir „python-pip". „Idlak" naudoja „XML" failų formatą, todėl reikėjo „XML" paketo, bei pip paketo, kuris yra naudingas įrankis, padedantis lengviau įdiegti ir rasti kitus reikalingus paketus sistemos darbui. Taip pat reikėjo įdiegti „numpy" paketą ir „lxml" paketą, kurie buvo reikalingi jau naudojant prieš tai sudiegtą „pip" paketą. „Pip" komanda su kuria įrašomas numpy atrodo taip: pip3 install numpy.

4. Balso sintezės mokymas

4.1. Praktinio darbo tikslas

Praktinio darbo tikslas yra pasinaudojus „Idlak Tangle" sistema sukurti lietuvių kalbos modelį, kuris galėtų įvestą tekstą paversti sintetiniu balsu, šnekančiu lietuviškai.

4.2. Tekstinių duomenų tvarkymas

Prieš pradedant darbą su „Idlak" sistema buvo gauti duomenys lietuvių kalba, kuriuos sutvarkyti ir panaudoti TTS mokymams. Buvo gauta 100 - as audio failų, „excel" dokumentas, su visomis įrašytomis frazėmis, ir .lab failai, kurie turėjo visą informaciją, apie kiekvieno įrašyto žodžio tarimą. Buvo pastebėta, kad „.lab" failai netikami kalbos mokymui su „Idlak" sistema, nes „Idlak" naudoja „XML" failų formatą. Dėl šios priežasties reikėjo perdaryti „.lab" failus. Ši užduotis užėmė daug laiko, nes „.lab" failuose viena eilutė reprezentavo vieną tariamą fonemą, o „Idlak" tarimo failo struktūra kitokia - faile turi būti vieno žodžio pilnas tarimas ir šalia parašytas atitinkamas žodis. Gauti diktoriaus įrašyti audio failai ir tekstiniai failai yra naudojami projekte LIEPA [16].

```
0   406803  pau
406803  1292263 k
1292263 1689777 a
1689777 2736063 ts
2736063 3191851 t
3191851 3744985 a
3744985 4580486 t'
4580486 5464487 Ii
5464487 5772842 b'
5772842 6185647 i
6185647 6612392 n'
6612392 7132394 i
7132394 7472638 n
7472638 7977947 k
7977947 8328007 a
8328007 8744004 j
```

5 pav. „.lab" fonemų failas

```
<?xml version='1.0' encoding='utf8'?>
<lexicon>
<lex pron="aa0" entry="full" default="true">A</lex>
```

6 pav. „lexicon" „XML" failas

Kadangi .lab faile tarp žodžių nebuvo jokių skiriamųjų ženklų, išskyrus skiriamusius „pau" ženklus tarp sakinių, visų (apie 30000) atskirų žodžiu tarimus reikėjo atrinkinėti ranka. Taip pat nebuvo galima labiau automatizuoti šio proceso, nes tas pat žodis gali turėti kelis skirtingus tarimus, taip pat kai kurie žodžių tarimai neturėjo pabaigų, nes dviejų žodžių tarimas buvo sujungtas į viena. Taip pat žodžio ilgis ne visada atitinka tariamų raidžių ilgį ir tarimas ne visada prasideda ar baigiasi ta pačia raide, kaip originalus žodis, todėl nebuvo galima lengvai automatizuoti žodžių tarimo formatavimo proceso. Naudojantis įrašytų sakiniu failu teko žiūrėti, koks turi būti žodžio tarimas. Taip buvo sudarytas ir paruoštas žodžių ir jų tarimų failas, kuris vėliau buvo labai naudingas ir dažnai naudojamas tolimesniems „Idlak" TTS mokymo žingsniams.

4.3. Testinis mokymo bandymas

Norint išsiaiškinti, kaip veikia „Idlak" sintezavimo procesas, pradžioje reikėjo išbandyti ir paleisti, jau paruoštą pavyzdį, kuris naudoja anglų kalbą ir automatiškai suteikia tiek teksto, tiek

audio resursus. Tai buvo atliekama norint apmokyti modelį vėliau naudoti teksto į balsą sintezę. Pavydzdžio direktorija vadinosi „tts_tangle_arctic". „tts" reiškia „text to speech", „tangle", tai balso apmokymo sistema, kurią naudoja „Idlak" sistema tam, kad apjungti įvarius sistemos procesus, kaip DNN ir „MLSA" vokoderis, „arctic" - tai įrašytas ir naudoti paruoštas balso duomenų paketas. Pavyzdį paleisti nebuvo sudėtinga, reikėjo paleisti „run.sh" komandą, kuris pradėjo darbą. Komandinės eilutės kodas automatiškai parsiuntė audio ir tekstinius failus, juos apdorojo ir pradėjo neuroninio tinklo apmokymą. Pavyzdinis apmokymas neužtruko labai ilgai, apie valandą. Apmokymui sėkmingai pasibaigus, buvo galima išbandyti balso sintezę anglų kalba. Norint išbandyti teksto į balsą sintezę reikėjo įvesti komandą: „echo, įrašyti tekstą, kurį norima susintezuoti, | local/synthesis_voice_pitch.sh voices/en/ga/slt_pmdl <išvesties direktorija>". Atlikus šią komandą, išvesties direktorijoje, wav_mlpg aplankale, buvo galima rasti klausymui paruoštą failą, kuriame buvo susintezuotas duotas tekstas.

4.4. Pirmas mokymo bandymas

Pirmas mokymo bandymas buvo atliktas norint išmėginti gautus ir naudoti paruoštus lietuviškus tekstus ir audio įrašus, bei pažiūrėti, kaip reikia pasiruošti ir apmokyti „Idlak Tangle" sistemą su naujais duomenimis. Mokymas buvo atliekamas naudojant anglų kalbos leksiką su lietuviškais duomenimis, tam tikslui buvo perdaromas „tts_tangle_arctic" pavyzdys.

„Idlak" paruošti mokymui su naujais duomenimis nėra labai sudėtinga, reikia tik įdėti audio failus į „rawaudio" direktorijos aplanką, kuriame galima rasti kalbos ir akcento direktorijas. Jų viduje yra „orig" aplankas, kuriame yra audio failai skirti sistemai naudoti. Tekstiniai failai yra įkeliami į „lables" aplanką, kuriame taip pat yra kalbos ir akcento aplankalai, jų viduje ir reikia įdėti paruoštą tekstinį failą, kuris atitinka audio failų eiliškumą.

Darbo pradžioje buvo gauta 100 audio failų, kuriuose buvo po 50 įrašytų sakinių. Taip pat buvo gautas audio įrašams naudojamų tekstinių sakinių sąrašas, iš kurio buvo galima sukurti XML failą su sakiniais, kurių eiliškumas atitinka audio failų eiliškumą. Kadangi vienas audio įrašas atitiko 50 sakinių, todėl tekstinis failas buvo atvaizduotas taip pat.

Tačiau, žiūrint į kitus testinius failus, buvo pastebėta, kad viename audio faile buvo įrašytas tik vienas sakinys ir tekstiniame faile tai pat pagal eiliškumą buvo priskirtas tik vienas sakinys vienam audio failui.

Pirmiausia buvo mėginta atlikti bandymą su audio ir sutvarkytais tekstiniais failais, pagal audio failo formatą, tai yra 50 sakinių vienam failui. Paleidus mokymą, gauta klaida failų apdorojimo procese. Atlikus bandymus sumažinant sakinių kiekį, atitinkantį vieną audio failą tekstiniame faile, buvo pastebėta, kad pakeistame teksto skyriuje klaidos nebeliko. Todėl tapo aišku, kad visus audio ir tekstinius failus reikės suskaidyti po sakinį.

4.4.1. Duomenų pertvarkymas

Kadangi buvo gauta 100 audio failų, kuriuose įrašyta po 50 sakinių, norint sutvarkyti audio failus „Idlak" naudojimui, reikėjo audio failus sukarpyti į atskirus 5000 sakinių. Tam pasirinkta naudoti programa „Audacity". Naudojantis šia nemokama programa kiekviename audio faile galima automatiškai aptikti pauzes tarp sakinių. Tam naudojamas nustatymas „Silence Finder". Taip išsaugoti 5000 skirtingų audio failų. Automatinis failo suskirstymas sakiniais labai padėjo greitai atlikti darbą, tačiau tai vis tiek reikalavo daug laiko, nes kiekvienam iš 100 audio failų reikėjo atskirai aptikti pauzes, tada vieną po kito išsaugoti. Išsaugant naujus sakinius reikėjo patvirtinti visų

50 naujų audio failų saugojimą 100 kartų. Išsaugojus 5000 audio failų, reikėjo juos visus pervadinti tam tikru formatu, kad „Idlak" mokymo procesų failai būtų aptinkami ir teisingai priskiriami tekstiniams ir audio failams. Tai buvo atlikta naudojant Linux terminalo komandą, kuri visų aplankale esančių failų pavadinimus pakeičia į tinkamą audio failų numeraciją, kuri buvo „lt_a0001". Ši numeracija daryta pagal anglų kalbos audio failų pavadinimų pavyzdį.

Pakeitus audio failų pavadinimus reikėjo pakeisti ir tekstiniame faile esančių sakinių paskyrimo pavadinimus, nes pirminiame faile vienam audio failui buvo priskiriama 50 sakinių. Dėl šios priežasties reikėjo tekstinį failą perdaryti ir teisingai kiekvienam audio failui priskirti vieną sakinį tekstiniame faile, pagal atitinkančio tekstą audio failo pavadinimą. „Idlak" sistema prieš pradedant mokyti pakeičia audio failų pavadinimus į kitus pavadinimus, kurie vėliau yra naudojami mokymui. Šiuo pavadinimu reikėjo pavadinti ir visus tekstinio failo sakinius. „Idlak" sistemai ruošiant duomenis audio failai yra pakeičiami tokia tvarka: pirmiausia rašomas failui jau priskirtas pavadinimas „lt_a0001" ir prie jo yra pridedama „_000", kuris nurodo atitinkamą sakinį eilės tvarka. Jeigu sakinių skaičius pasiekia 999, failo pavadinimo pradžios formatas yra padidinamas vienu, kaip pavyzdžiui „lt_a0002_000". Bandant susintezuoti lietuvišką balsą reikėjo labai dažnai keisti didelį kiekį teksto, todėl visas tekstas su reikiamu XML formatavimu buvo įdėtas į „excel" lentelę tam, kad būtų galima greičiau tvarkyti tekstą.

4.4.2. Pirmo mokymo pratęsimas

Tinkamai sutvarkius tiek audio, tiek tekstinius failus, toliau buvo galima apmokyti „Idlak" balso sintezės sistemą. Pakeisti audio ir tekstiniai failai atitiko sintezės sistemos formatavimo reikalavimus ir mokymo procesas galėjo vykti toliau. Tačiau greitai atsirado dar viena problema, kuri neleido pabaigti mokymo proceso. Kadangi pirmas mokymas yra daromas pagal anglų kalbos pavyzdį, o tekstinis failas buvo naudojamas lietuvių kalba, „Idlak" nemokėjo apdoroti lietuviškų rašmenų. Mokymas nutrūkdavo dėl tekste naudojamų lietuviškų raidžių, todėl jos buvo išimtos iš tekstinio failo testavimo tikslais, norint pratęsti procesą ir pažiūrėti, koks sintezavimo rezultatas.

Panaikinus lietuviškus rašmenis iš tekstinio failo, „Idlak" galėjo sėkmingai tęsti darbą. Tačiau, priėjus neuroninio tinklo mokymo dalį, vėl įvyko sistemos klaida, nes mokymas naudojant GPU nepavyko, nes buvo per mažai resursų, reikalingų mokymui. Dėl šios priežasties teko pakoreguoti „dnn" aplankale esantį failą, kuris buvo atsakingas už „CUDA" sistemą ir neuroninio tinklo apmokymą naudojant GPU. Pakeitus failo nustatymus ir paleidžiant sistemą, mokymas buvo automatiškai perduotas CPU darbui, tačiau daugiau problemų neuroninio tinklo proceso metu neiškilo ir CPU ganėtinai gerai ir greitai atliko skaičiavimus, reikalingus neuroninio tinklo mokymui.

Praėjus maždaug 2 valandoms nuo gilaus neuroninio tinklo mokymo proceso dalies pradžios buvo pastebėta, kad „root" sistemos direktorijoje liko labai nedaug vietos ir galiausiai, jai pasibaigus, mokymo procesas sustojo. Ruošiant Linux sistemą nebuvo tikimąsi, kad „root" sistemos direktorija bus naudojama dideliems duomenų kiekiams saugoti, todėl buvo paskirta tik palyginus nedidelė dalis vietos, reikalingos tik sisteminiams moduliams saugoti. Tačiau, neuroninio tinklo darbo metu sistema generavo daug laikinų (tmp) failų, kurie ir buvo saugomi „root" sisteminėje direktorijoje, todėl, norint sėkmingai apmokyti neuroninį tinklą, balso sintezei reikėjo „root" sistemos direktorijos vietą padidinti.

„Root" direktoriją, kuri reikalinga „tmp" failams saugoti neuroninio tinklo mokymo metu, padidinti nebuvo labai paprasta. Šioje direktorijoje buvo daug Linux sistemai reikalingų failų, todėl, įsivėlus klaidai, didinant šią direktoriją galima prarasti visą Linux operacinės sistemos aplinką. Taip pat, norint padidinti „root" direktorijos talpą, negalima naudoti operacinės sistemos iš disko.

Dėl šios priežasties vėl reikėjo naudoti „usb" raktą; įsirašyti Linux Mint sistemos instaliacijos failą ir operacinę sistemą ir paleisti sistemą iš „usb" laikmenos. Taip paleidus Linux sistemą, atidaryta programa „gparted", naudojantis ja atlaisvinta vieta kietajame diske,kuri suteikta reikiamai „root" direktorijai. Papildomai buvo atlaisvinta dar 100 GB, iš viso laisvos vietos „root" direktorijoje buvo apie 120 GB. „Root" sistemos direktorijos talpos didinimas įvyko sėkmingai ir jokie svarbūs sisteminiai failai nebuvo prarasti.

Padidinus „root" direktorijos talpą ir iš naujo paleidus „Idlak" balso sintezės lietuvių kalba mokymo procesą, daugiau problemų neiškilo. Pasibaigus mokymams, buvo pastebėta, kad mokymo metu sukurti laikinieji failai pasibaigus visiems procesams užėmė apie 100 GB vietos.

Pasibaigus „Idlak" mokymo procesui buvo pastebėta, kad pirmas bandymas apmokinti lietuvišką balso įrašą ir tekstą su anglų kalbos fonetika nepasisekė. Treniravimas iš viso užtruko apie 8 valandas, o pasibaigus mokymo procesui, pabandyta gauti balsą, tiek su lietuviškais, tiek su angliškais sakiniais, tačiau sugeneruotas balsas visiškai neatitiko to, kas buvo rašoma abejomis kalbomis. Taip pat gautuose sakinių įrašuose buvo labai daug pašaliniu garsų, tačiau tai galėjo įvykti, nes neuroninis tinklas buvo blogai išmokytas, naudojant skirtingų kalbų duomenis.

4.5. Antras mokymo bandymas

Pradėjus ruoštis mokymui su pilna lietuvių kalbos fonetika buvo pastebėta, kad pirmas bandymas galėjo būti nepavykęs, nes buvo blogai pakeisti audio failų, kurie buvo sukarpyti į 5000 skirtingų sakinių, pavadinimai. Pavadinimai neatitiko tekstinių failų pagal savo eiliškumą, todėl mokymas buvo atliktas blogai. Taip pat mokymas galimai buvo atliktas blogai, nes bandant sistemą mokinti dar kartą, „Idlak" informavo apie naują klaidą, nes paruošti audio failai buvo 32 bitų, o "Idlak" sistema reikalavo, kad audio failai būtų 16 bitų dydžio. Neaišku, kodėl šios klaidos sistema nepastebėjo anksčiau, nors pirmąjį kartą kalbos buvo mokoma su tais pačiais, 32 bitų audio failais. Pastebėjus padarytas klaidas, jos ištaisytos ir mokymas atliktas dar kartą. Paaiškėjo, kad komanda, kuri naudota audio failų pavadinimų pervadinimui, juos pervadino rikiuodama nuo didžiausio iki mažiausio, todėl paskutiniai audio failai tapo pirmais ir negalėjo sutapti su apmokymui skirtu, audio failą atitinkančiu tekstu. Surasta nauja komanda, kuri teisingai pervadina failus pagal norimą failų rikiavimą: „num=1;for i in 'ls *wav | sort -V'; do mv "$i" "lt_a" "$(printf '%04d' $num).$i#*.";((num++));done;" Perdaryti audio failus iš 32 bitų į 16 bitų sudėtinga nebuvo, nes „Audacity"programa turi funkciją, kuri išsaugant failus gali pakeisti jų bitų formatą. Daugiausia laiko užtruko patvirtinti visus 5000 audio failų tam, kad juos išsaugoti, tačiau į pagalbą buvo pasitelkta Linux sistemos automatinio paspaudimo programa „xdotool", kuri padėjo greičiau patvirtinti visų failų išsaugojimą.

Sutvarkius visus failus ir juos dar kartą patikrinus, antrasis apmokymas naudojant lietuviškus duomenis su anglų kalbos taisyklėmis buvo sėkmingas, nes visi failai buvo tvarkingai išdėstyti ir audio failai buvo pakeisti iš 32 bitų į 16 bitų. Baigus mokyti, „Idlak Tangle" sistema sugeneruoto balso kokybė buvo vidutiniška, girdėjosi pašalinių garsų ir balsas skambėjo truputi „robotiškai". Kadangi apmokymas vykdytas naudojant anglų kalbos duomenis, daugelis žodžių buvo tariami netinkamai ir, norint, kad juos sintezatorius tartų tinkamai, reikia rašyti juos kitaip nei yprasta lietuvių kalba, pavyzdžiui, vietoj „ė" rašyti „ee", kad žodis ar raidė būtų sakomi teisingai.

4.6. Lietuvių kalbos diegimas

Lietuvių kalba „Idlak" sistemoje buvo kuriama ne tik pagal anglų kalbos pavyzdį bet ir kitas sistemoje jau sukurtas kalbas. Daugiau informacijos buvo galima gauti iš „Idlak" dokumentacijos puslapio, tačiau daug naudingos informacijos puslapyje nebuvo, todėl reikėjo daugumą dalykų aiškintis ir ieškoti pačiam.

Kuriant naują kalbą, reikia sukurti pagrindinį katalogą su kalbos pavadinimu, taigi, pirmas katalogas buvo pavadintas „lt". Į šią direktoriją buvo nukopijuoti visi failai, esantys anglų kalbos direktorijoje. Viduje sukurto „lt" aplanko turi būti dar vienas aplankas, kuris reiškia akcentą, nes buvo kuriama bendrinė lietuvių kalba. Kitos kalbos antrąjį aplanką yra pavadinusios taip pat, kaip ir pagrindinis aplankas, todėl antras aplankas, kuris anglų kalboje buvo pavadintas „ga" buvo pervadintas į „lt", nurodant, kad bus naudojama lietuvių kalba. Turint visus failus „lt" direktorijoje, juos reikėjo sutvarkyti ir pritaikyti lietuvių kalbai, taip pat kitus aplankus, kurie specifiškai pritaikyti anglų kalbai ir buvo naudoti testinio mokymo tikslams, reikėjo ištrinti. Pagrindiniai failai, kurie buvo reikalingi ir juos reikėjo koreguoti buvo „trules-default.xml", „lexicon-default.xml", „phone-default.xml", „sylmax-default.xml" ir „ccart-default.xml".

Be šių išvardintų failų koregavimo, norint sukurti lietuvių kalbą, reikėjo padaryti ir tam tikrų pakeitimų pačios „Idlak" sistemos failuose bei teisingai sudėti naudojamus lietuvių kalbos failus į reikiamas vietas sintezės mokymo direktorijoje.

4.6.1. lexicon failas

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <lexicon>
3     <lex pron="" entry="" default="">grapheme</lex>
4     <lex pron="a0" entry="full" default="true">a</lex>
5     ..
6 </lexicon>
```

„Lexicon-default.xml" buvo vienas pagrindinių failų, reikalingas sukurti lietuvių kalbos modelį. Į šį failą reikėjo įdėti įvairius žodžius ir jų tarimus. Šis procesas užtruko ilgiausiai. Žodžiams ir jų tarimams buvo panaudoti žodžiai iš gautų, apmokyti skirtų duomenų. Tačiau, sudėjus duomenis į failą, juos taip pat papildomai reikėjo apdoroti. Pirmiausia reikėjo sudėti tinkamas XML žymas, tuomet sudėti akcentus ant tam tikrų, tarime esančių raidžių, kurios yra paskiriamos „sylmax-default.xml" faile. Taigi, reikėjo nuspręsti, kuriuose žodžiuose kokios pasirinktos raidės bus kirčiuojamos. Pavyzdžiui, „rasti'" ir „ra'sti". Šiuo atveju gautas pirminis tarimo tekstas jau turėjo sudėtus tam tikrus požymius ir pagal juos, „excel" raidžių pakeitimo įrankio pagalba, buvo galima lengvai pakeisti - prie tam tikrų raidžių pridėti 0 arba 1 simbolius, kur nulis reiškia trumpai tariamą raidę, o vienetas - ilgai. Sudėjus ženklus ant tam tikrų ilgai ar trumpai tariamų raidžių, buvo galima išrinkti ir ištrinti vienodus žodžius, kurių tarimas taip pat yra vienodas. Kadangi prie tarimo prisidėjo papildomi simboliai, vienetas ir nulis, skirtingų žodžių kiekis padidėjo. Šiam vienodų žodžių atskirimo procesui buvo pasitelkta „excel" programa, kurios pagalba reikalingi sakiniai buvo pakankamai lengvai išrūšiuoti. Išrūšiavus sakinius, liko tokių žodžių, kurie yra vienodi, bet jų tarimas gali būti skirtingas, tuomet iš turimų žodžių reikėjo nurodyti, kuris iš vienodo žodžio tarimų bus naudojamas kaip pagrindinis. Atrinkimas buvo atliktas remiantis bendrine lietuvių kalba, ištariant žodį ir klausantis, kaip jis sakomas. Šį procesą reikėjo atlikti rankiniu būdu, reikėjo pačiam nustatyti, kuris žodio tarimas turi būti naudojamas, kaip pagrindinis.

4.6.2. phone failas

```
1 <phoneset language="" accent="" >
2     <phone name="" ipa="" example-word="" example-pron="" syllabic="" />
3     ...
4 </phoneset>
```

„Phone" failas susideda iš visų fonemų, kurios yra naudojamos lietuviškų žodžių tarime, bei jų panaudojimo žodžiuose. „IPA" - tai kanoninis simbolio atvaizdavimas, kuriam reikėjo susirasti lietuviškų „IPA" simbolių reikšmes. „Example-word" - tai pavyzdinis žodis užrašytas įprastais lietuviškais rašmenimis. „Example-pron" - tai pilnas pavyzdinio žodžio tarimas. „Name" ir „Syllabic" parametrai nebuvo naudojami. Taigi, „Phone" faile buvo nurodytos visos abėcėlės fonemos, naudojamos žodžių tarime, ir pavyzdiniai žodžiai bei jų tarimas, kurie buvo pasirinkti iš leksikono faile esamų žodžių, turinčių reikiamą fonemą.

4.6.3. trules failas

„Trules-default.xml" - tai failas, kuriame yra nurodyti kalbos simboliai ir raidės, taip pat nurodyta, kaip pakeisti netinkamas raides apdorojant tekstinius failus prieš mokymą, kokias raides naudoti ir kaip pakeisti didžiąsias raides į mažąsias. Kaip rašoma „Idlak" dokumentacijoje, galima kopijuoti anglišką šio failo pavyzdį ir atlikti keletą modifikacijų. Taip pat, kuriant šį failą teko žiūrėti ir į kitus, ne anglų kalbą paruoštus failus, nes juose buvo raidės su specialiais simboliais, todėl buvo aiškiau, kaip reikėtų failą pritaikyti naudojant lietuvių kalbos rašmenis.

Skiriamųjų ženklų ir kitų simbolių keisti nereikėjo, nes visi standartiniai simboliai ir skiriamieji ženklai jau buvo aprašyti anglų kalbos faile. Reikėjo pridėti lietuviškas specifines raides prie XML lauko „alpha". Taip pat buvo pridėta specialių lietuviškų raidžių ir prie XML lauko „downcase" simbolių, kurie parodo, kaip didžiąsias raides reikia konvertuoti į mažąsias. Daugiausia eksperimentavimo ir pakeitimų reikėjo XML lauko „convertillegal" simbolių rinkiniui. Ieškant pavyzdžių buvo rasti XML lauko „convertillegal" duomenys rumunų kalbos faile, kurie gerai atitiko lietuvių kalbą, nes tarp specifinių rumuiškų raidžių buvo pridėtos ir lietuviškos. Minėtame faile lietuviškų specifinių raidžių konvertavimas buvo priskirtas į paprastas raides. Gauti reikiamus XML lauko „convertillegal" duomenys, buvo lengvai pritaikyti lietuvių kalbos specifinėms raidėms, kurios konvertuotos iš įprastų į tokias, kurios atitinka „utf8" standarto reikalavimus.

4.6.4. sylmax failas

„Sylmax" failas taip pat buvo paimtas iš anglų kalbos pavyzdinio failo. Kitų kalbų failai buvo padaryti tokiu pačiu principu. Šiame faile reikėjo angliško tarimo fonemas pakeisti į lietuviškas. Šį failą pertvarkyti ir pritaikyti lietuvių kalbai nebuvo sunku, nes buvo naudojamasi anglų kalbos pavyzdžiu; buvo perdarytos XML „syllabic", „onsets" ir „nuclei" laukų reikšmės į lietuvių kalbos tarime naudojamus simbolius.

4.6.5. LTS taisyklių kūrimas

„Idlak" naudoja „cart" medžius tam, kad būtų sukurtos raidžių į garsus taisyklės. Šį failą galima sugeneruoti automatiškai pasinaudojant „Phonetisaurus" paketu, jam pateikus porą reikalingų ir jau paruoštų failų.

```
IDIR=<path to idlak>
cd $IDIR/idlak-misc/cart_lts
./run.sh -l $IDIR/idlak-data/<lng>/<acc>/lexicon-default.xml \
         -s $IDIR/idlak-data/<lng>/<acc>/sylmax-default.xml \
         -p $IDIR/idlak-data/<lng>/<acc>/phone-default.xml \
         -o $IDIR/idlak-data/<lng>/<acc>/ccart-default.xml
```

Bandant sugeneruoti „ccart-default.xml" failo duomenis buvo gauta klaida - lietuvių kalbos specialios raidės nėra tinkamos ir jas reikia pakeisti kitomis, kurios atitinka „utf8" standarto reikalavimus. Taigi, reikėjo visus specifinius lietuvių kalbos simbolius pakeisti. Norint pakeisti simbolius buvo rasta ir pasinaudota „utf8" sertifikatui ekvivalentūs „unicode" simboliai.

„Ą", „Ę", „Į" ir „Ų" nosinėms raidėms buvo paskirtas „Ogonek" simbolis (7 pav.), „Š" ir „Ž" raidėms buvo paskirtas „Caron" simbolis (8 pav.), „Ė" raidei buvo paskirtas „Dot" simbolis (9 pav.), „Ū" raidei buvo paskirtas „Macron" simbolis (10 pav.).

"̨" (U+0328)

7 pav. Ogonek simbolis
[8]

"̌" (U+030C)

8 pav. Caron simbolis
[3]

"̇" (U+0307)

9 pav. Dot simbolis
[6]

"̄" (U+0304)

10 pav. Macron simbolis
[7]

Pakeitus šias raides į raides su išvardintais simboliais, buvo galima tvarkingai sugeneruoti „ccart-default.xml" failo duomenis. Sutvarkius „lexicon" failą ir paleidus programą toliau dirbti, buvo susidurta su dar viena problema, kuri susijusi su reikalingu paketu, naudojamu generuojant „ccart" failą. Apie šio paketo naudojimą nerasta jokios informacijos. Galiausiai išsiaiškinta, kad problema buvo su paketu „wagon", kuris buvo naudojamas „Phonetisaurus" programoje. Rastas „pip" paketas su „wagon" pavadinimu. Įdiegus paketą problema išliko ir pasidomėjus įdiegtu „wagon" paketu, buvo pastebėta, kad tai nėra tas pats paketas, kuris buvo naudojamas programoje generuojant „ccart" failą. Tai buvo kitas paketas, su tokiu pačiu pavadinimu, kuris buvo prieinamas „pip" modului. Rasti reikiamą „wagon" paketą, nebuvo lengva, nes jis yra viena iš „The university of Edinburgh festival speech synthesis system" kalbos paketo dalių.

Radus reikiamą „wagon" paketą reikėjo parsisiųsti „speechstools" aplanką, kuriame buvo visi reikiami failai. Tačiau, norint, kad kodas galėtų naudoti „wagon" paketą, reikėjo programos direktorijos nuorodą pridėti į Linux Mint sistemos „$PATH" direktoriją. Norint paleisti „ccart" turinio generavimo programą, reikėjo į sistemą įdiegti ir „python toolkit", bei „python dev" paketus. Sutvarkius „wagon" paketą ir paleidus „ccart" failo generavimo programą, naudojančią „Phonetisaurus", iškilo dar viena neaiški problema, dėl kurios nepavykdavo generuoti „b.cart failo". Šį failą programa norėdavo generuoti dar kartą, būdavo rašoma, kad faile yra papildomas simbolis. Ši problema išspręsta modifikuojant programos, kuriančios „ccart" failą, kodo dalį. Kode reikėjo pakeisti eilutę „ID=$(basename $f .cart)" į „ID=$(basename “$f” .cart)" ir problema buvo išspręsta. Rasta, kad pridėjus " ženklus prie ir už „$f" kintamojo, ignoruojami neaiškūs simboliai ir programa gali veikti toliau.

Sutvarkius visas problemas buvo galima sėkmingai generuoti „ccart-default.xml" failo turinį.

4.7. Trečias mokymo bandymas

Trečias „Idlak" TTS mokymo bandymas buvo daromas naudojant lietuvių kalbos modelį, sukurtą „Idlak" aplinkoje. Visi mokymo failai buvo jau sutvarkyti ir panaudoti ankstesniuose mokymuose, naudojant anglų kalbą, todėl dėl jų nekilo problemų. Bet pradedant darbą su lietuvių kalbos modeliu teko atlikti pakeitimų „run.sh" faile, kuriame yra visi nustatymai, reikalingi TTS mokymui, bei nurodytos mokymui reikalingų failų vietos ir kitos komandos, kurias reikėjo pakoreguoti, norint sėkmingai apmokyti kalbos modelį.

„SRATE" nustatymas buvo paliktas 48000, nes audio failai buvo sukurti naudojant 48 kHz dažnį. „Spks" parametre buvo taip pat palikta „slt" reikšmė, nes anglų kalba naudojo šį trumpinį ir jis turėjo būti vienas iš egzistuojančių trumpinių „Idlak" sistemoje. „Slt" buvo pavadinti ir audio, ir tekstiniai failai. Tai atlikta pirmo ir anrojo mokymų metu, kai buvo naudojama anglų kalba. Taigi, kad daugiau nereiktų keisti failų pavadinimų, buvo paliktas šis nustatymas. Toliau buvo pakeistas nustatymas „nodev" iš 50 į 500 testams skiriamų duomenų, pagal proporciją: 500 anglų kalbos failų : 50 „nodev" paskirtų failų, taigi 5000 lietuvių kalbos failų : 500 „nodev" paskirtų failų. Šis nustatymas reiškia kiekį mėginių, kuriuos galima skirti praradimui mokymo metu, o visi kiti sakiniai lieka balso modelio treniravimui. Programa buvo pritaikyta naudoti kartu su anglų kalbos pavyzdžiu; kadangi dalis jos buvo skirta audio ir tekstiniams anglų kalbos failams atsiųsti, šią dalį reikėjo panaikinti, nes anglų kalbos failai būtų užrašomi ant paruoštų lietuvių kalbos duomenų. Taip pat tekstinio failo apdorojimo dalis buvo paimta iš kito, rusų kalbos, pavyzdžio, kuris tiko geriau, kai buvo dirbama su lietuvių kalbos duomenimis, paruoštais rankiniu būdu. Atlikus šiuos pakeitimus, buvo galima paleisti „run.sh" programą ir bandyti apmokyti TTS sistemą lietuvių kalbos modeliu.

```
1 #!/bin/bash
2 srate=48000
3 nodev=500
4 #  Audio
5     for f in $spkrawaudio/*_orig/*.wav; do
6         if [ ! -e "$f" ]; then
7             mkdir -p $spkrawaudio
8             cd $spkrawaudio
9             #wget -c -N $url
10            #tar -xjf $arch
11            # get original sample rate
12            for i in $spkrawaudio/cmu_us_${spk}_arctic/orig/*.wav; do
13                orgsrate=`sox --info -r $i`
14                break
15            done
16         #text script
17        if [ ! -e $label_dir/text.xml ]; then
18        mkdir -p $label_dir
19        cd $label_dir
20        wget -c -N $laburl
21    fi
```

Procesas užstrigo antrame balso sintezės apmokymo žingsnyje. Pirmos iškilusios klaidos nebuvo galima suprasti, nes funkcija, kuri iškviesdavo klaidą, buvo paleidžiama naudojant vie-

ną kodo eilutę su jos specifiniais nustatymais. Klaida atrodė taip: „basic_stirng:_pos (which is 18446744079551615) > this->size() (which is 0)". Kad išsiaiškinti problemą, buvo panaudotas „valgrind" Linux paketas. Įrašius „valgrind" eilutės, kuri iššaukdavo klaidą, pradžioje, „valgrind" parodė aiškesnę klaidos išvestį. Tuomet išsiaiškinta, kurioje vietoje ir kokiame faile ieškoti klaidos. Problema buvo tokia: „lexicon" lietuvių kalbos apmokymo faile, buvo tokių eilučių, kurios turėdavo papildomus tarpo ženklus sakinio pradžioje ar pabaigoje, tai iššaukdavo šią klaidą. Ištaisius klaidą, labai greitai atsirado kita problema, kurią reikėjo spręsti. Problemą sukėlė specifinės lietuvių kalbos raidės. Kaip buvo minėta, „trules-default.xml" failas atsakingas už teisingą specifinių simbolių pakeitimą į „utf8" standartui tinkamus skaityti simbolius. Pakeitus minėtus simbolius, „Idlak" sistemos „python_make_lang.py" programa vis tiek rodė klaidą ir neleido jų naudoti. Klaidos tekste buvo rašoma apie „\u364d" simbolių seką, kuri atitinka tam tikrą probleminį simbolį. Tuomet buvo pabandyta lietuvių kalbos simbolius pakeisti į jų skaitines reikšmes, kaip, pavyzdžiui, „š" raidė atitinka

```
(U+0073) - (U+030C) = U+0161.
```

. Dėl šios priežasties buvo bandyta pakeisti simbolius „trules" faile; vietoj specialių lietuviškų raidžių įrašyti jų skaitines reikšmes „convertillegal" failo skiltyje. Programa konvertuodama į nurodytus simbolius interpretavo simbolių seką, ne kaip vieną specialią raidę, bet kaip paprastą simbolių seką, todėl „Idlak" balso apmokymo programa mokino žodžius, kuriuose vietoj specialios raidės buvo simbolių junginys, kaip, pavyzdžiui, „U+0161alta" („šalta").

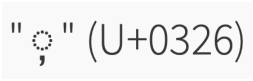
Taigi, pirmą lietuvių kalbos mokymo bandymą teko supaprastinti. „Trules-default.xml" faile specialius lietuvių kalbos simbolius pakeisti paprastomis raidėmis ir taip apmokinti modelį. Pakeitus raides, viskas veikė gerai ir gautas tvarkingas lietuviško balso modelis. Specialiai kurta lietuvių kalba apmokius turimus lietuvių kalbos duomenis, rezultatas buvo kur kas geresnis nei prieš tai atliktuose bandymuose. Be to, nebereikėjo specialiai kartoti raidžių, norint išgauti tam tikrą raidės ilgį ar garsą, nes net parašius tam tikrus žodžius, nenaudojant specialių raidžių, buvo gaunama taisyklinga lietuvių kalba. Žodžiai, kurie buvo naudojami paruoštame „Idlak" lietuvių kalbos leksikone, buvo tariami teisingai ir nenaudojant specialių lietuviškų raidžių. Tačiau, tai galiojo ne visiems žodžiams, leksikone buvę žodžiai buvo generuojami tvarkingai, tačiau nauji žodžiai, kurie nebuvo naudojami apmokant, buvo tariami taip, kaip rašomi. Balso kokybė ir sakinių aiškumas buvo geresni, taip pat, net ir sakiniai, kurie buvo generuojami be specialių simbolių, skambėjo tvarkingai ir teisingai, nors ir nebuvo tariami taip, kaip būtų tikimąsi, bet buvo generuojami taisyklingai, taip, kaip buvo parašytas sakinys.

4.8. Ketvirtas mokymo bandymas

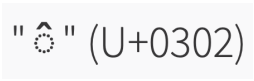
Ketvirtas mokymo bandymas buvo skirtas pažiūrėti, kaip galima panaudoti specifiniu lietuvių kalbos simboliu TTS sintezėje. Prieš tai buvęs lietuvių kalbos mokymas nebuvo apmokytas specifinėmis lietuvių kalbos raidėmis, todėl sintezuojant balsą nebuvo galima pasirinkti, kaip tam tikras žodis bus tariamas, nes skirtingai parašytas žodis be lietuviškų rašmenų tariamas skirtingai, nei lietuviškiems rašmenims esant.

„Trules" faile pakeitus „convertillegal" skiltyje esančius ženklus į lietuvių kalbos ženklų atititikmenis, nebuvo leidžiama apmokyti sintetinio balso lietuvių kalbos modeliu, todėl buvo žiūrima, kaip šis žingsnis yra atliktas kitose kalbų, kurios taip pat turi specialių simbolių, modelius. Buvo naudojamasi rumunų kalbos modeliu, nes šiame kalbos modelyje yra simbolių, panašių į specifinius lietuvių kalbos rašmenis. Buvo pasirinktas simbolis „comma" nosinėms raidėms (11 pav.) ir

„circumflex" simbolis (12 pav.) „č", „ž", „ė" ir „ū" raidėms, nes visos minėtos raidės turi ženklą viršuje. Buvo tikimasi, kad atlikti pakeitimai bus sėkmingi, nes rumunų kalbos modelis šiuos simbolius naudoja „Idlak" sistemos mokymo metu.



11 pav. Comma simbolis
[5]



12 pav. Circumflex simbolis
[4]

Pakeisti šias raides reikėjo ne vien „trules" faile, bet ir kituose lietuvių kalbos modelio failuose. Pirmiausia simboliu, paimtu iš rumunų kalbos modelio specialių simbolių, pakeista viena raidė tam, kad patikrinti, ar galima vykdyti mokymą. Pakeitus vieną raidę atrodė, kad mokymas vyksta sklandžiai, todėl buvo nuspręsta visas raides pakeisti jau aptartais specialiais simboliais. Tačiau, pakeitus visas raides, vėl iškilo ta pati problema - "Idlak" sistema vis tiek negalėjo konvertuoti nurodytų simbolių. Paruoštose kitų kalbų pavyzdžiuose buvo naudojami specialūs simboliai, tai reiškė, kad tikrai yra įmanoma naudoti specialius simbolius, tačiau reikėjo suprasti, kaip galima modifikuoti patį mokymo kodą, kad būtų leista naudoti specialius simbolius. Pagal programos klaidos išvestį buvo rastas failas ir vieta, kurioje įvykdavo konvertavimo klaida. Šis failas buvo „idlak_make_lang.py". Su šiuo failu atlikta daug bandymų, galiausiai buvo rastas sprendimas. Norint naudoti specialius simbolius, šiame faile reikėjo pakeisti eilutę „word= str(attrs[at].upper())" į „word= str(attrs[at].upper().encode('ascii','ignore))". Pakeitus šią eilutę, buvo galima pradėti mokyti balsą, kai tekstas parašytas naudojant ir specialius simbolius. Atlikus šį pakeitimą, buvo galima naudoti ir prieš tai minėtus lietuviškų specifinių simbolių atitikmenis, todėl visos raidės buvo pakeistos iš rumuniškų ženklų atitikmenų atgal į lietuviškų raidžių atitikmenis. Pakeitus raides, reikėjo iš naujo sugeneruoti „ccart-default.xml" failą naudojant „Phonetisaurus" su visais naujais simbolių atitikmenimis. Tuomet pradėtas mokymas jau ir su lietuviškais specifiniais ženklais. Šis mokymo bandymas buvo sėkmingas ir, kaip ir buvo tikimasi, rezultatas šį kartą buvo gautas geriausiais. Žodžiai buvo tariami būtent taip, kaip ir buvo rašoma, taip pat visa bendra lietuvių kalbos sintezė buvo aiškesnė ir rišlesnė. Atliktas bandymas su žodžiais „abėcėlė" ir „abecele". Abu kartus sintezuotas balsas buvo teisingas, pirmą žodį tardamas su „ė" raidėmis balse, o antrą - su „e". Šis bandymas pasitvirtino ir su kitomis raidėmis bei žodžiais. Atliekant žodžių kirčiavimo bandymą pastebėta, kad žodžio tarimas gali pasikeisti priklausomai nuo parašyto žodžio. Pavyzdžiui, vardas „Eglė" buvo tariamas, kaip ir yra tikimasi, o bandant sintezuoti žodį „Egle", sintezuotas balsas jį tarė, kaip kreipinį. Balso sintezė su specialiais lietuviškais simboliais buvo sėkminga, tačiau balso kokybė vis tiek turėjo pašalinių garsų ir dažnai skambėjo truputi „robotiškai", šie pašaliniai garsai labiausiai girdisi tariant „s" raidę. Reikėtų paminėti, kad norint susintezuoti balsą, naudojant lietuviškų raidžių simbolius, reikia naudoti ne paprastas specifines lietuvių kalbos raides, bet jų atitikmenis, į kuriuos buvo specifiniai simboliai buvo keičiami „trules-default.xml" failo „convertillega" skiltyje. Bet kokiu atveju galima sėkmingai susintezuoti balsą, naudojant šiuos simbolius. Jei ateityje būtų kuriama „Idlak" teksto į balsą sintezės sistema su gražiai pritaikyta vartotojo sąsaja, būtų galima leisti vartotojui įvesti paprastas lietuvių kalbos specifines raides, tada jas automatiškai pakeisti į jų atitikmenis ir taip įvestus žodžius sintezuoti.

Išvados ir rekomendacijos

„Idlak" sistema yra geras įrankis, norint lengvai sukurti naują sintetinį balsą, ypač jei ta kalba jau egzistuoja. Tačiau, net ir sukurti naują kalbos modelį nėra labai sudėtinga. Jeigu norima tiesiog išgauti ne visai kokybiškų, bet pakankamai gerų rezultatų, naudojant anglų kalbos modelį galima šią sistemą apmokyti kalbėti bet kokia panašiai skambančia ar panašius simbolius naudojančia kalba. Toks bandymas buvo atliktas šio darbo pirmuosiuose sintezės bandymuose kuriant lietuvių kalbos modelį. Lietuvių kalbos modelio kūrimas šioje sistemoje reikalauja daug laiko, nes šiai sistemai reikia specialiai pritaikyti turimus tiek audio, tiek tekstinius failus. Pritaikius failus reikia išsiaiškinti, kaip sistema apdoroja failus ir kaip jie yra suformatuoti. Reikėtų paminėti, kad nors sistema yra pakankamai aiški, tačiau lengvai prieinamos informacijos apie ją praktiškai nėra. Pavyzdžiui, dokumentacija apie kalbos kūrimą užima tik vieną tinklapio puslapį, kuriame labai abstrakčiai aprašyta tam tikrų failų struktūra. Informacijos apie balso sintezę informacijos nėra daug, todėl aiškinantis specifinius sistemos reikalavimus ir savarankiškai atliekant tam tikrus bandymus sunaudojama daug laiko. „Idlak" sistemos pranašumas tas, kad ji skiriasi nuo daugumos kitų balso sintezės sistemų - norint sukurti naują sintetinį balsą, viską, nuo naujos kalbos sukūrimo ir failų tvarkymo iki neuronų tinklo modelio apmokymo ir teksto į balsą sintezės, galima atlikti įsidiegus vienintelį „Idlak" sistemos paketą ir visus reikiamus įrankius rasti jos viduje. Manau, kad sukurti lietuvių kalbos sintezės modelį pasisekė visai gerai; net ir nenaudojant lietuvių kalbos taisyklių pavyko sukurti pakankamai suprantamą balsą. Atlikus visus reikiamus darbus ir sukūrus lietuvių kalbos modelį sistemoje gavosi labai gerai suprantamas balsas, kuris pakankamai gerai generuoja įvairius lietuvių kalbos žodžius. Nors balsas gavosi suprantamas, tačiau jis nėra labai natūralus, panašūs sintetinio balso rezultatai buvo gauti ir kuriant pavyzdinį sintetinį balsą. Balso nenatūralumo priežastis yra ta, kad ši sistema vis dar yra vystoma. Rašant šį darbą pastebėta, kad sistema buvo nuolat atnaujinama. Peržiūrėjus sistemos kodą, galima pamatyti daug įvairių komentarų, kurie aprašo problemas susijusias su tam tikromis sistemos funkcijomis, kurias reikia išspręsti ar naujas apie funkcijas, kurias dar reikai sukurti. Kadangi „Idlak" sistema yra atviro kodo ir neturi licencijos suvaržymų, manau, kad ji ir toliau tobulės. Atsiradūs detalesnei šios sistemos dokumentacijai, turėtų atsirasti dar daugiau žmonių, kurie pradės naudoti šią sistemą ir taip prisidės prie jos tobulinimo, taip sukurdami labai patogią, naujomis technologijomis paremtą parametrinę balso sintezės sistemą.

Literatūros šaltiniai

[1] David Braude, Matthew Aylett, Caoimhín Laoide-Kemp, Simone Ashby, Kristen Scott, Brian Raghallaigh, Anna Braudo, Alex Brouwer, and Adriana Stan. All together now: The living audio dataset. pages 1521–1525, 09 2019.

[2] Ankita Chadha, Jagannath Nirmal, and Pramod Kachare. A comparative performance of various speech analysis-synthesis techniques. *International Journal of Signal Processing Systems*, 2, 01 2014.

[3] Compart. Combining caron. https://www.compart.com/en/unicode/U+030C.

[4] Compart. Combining circumflex accent. https://www.compart.com/en/unicode/U+0302.

[5] Compart. Combining comma below. https://www.compart.com/en/unicode/U+0326.

[6] Compart. Combining dot above. https://www.compart.com/en/unicode/U+0307.

[7] Compart. Combining macron. https://www.compart.com/en/unicode/U+0304.

[8] Compart. Combining ogonek. https://www.compart.com/en/unicode/U+0328.

[9] Marvin Coto and John Goddard. Speech synthesis based on hidden markov models and deep learning. *Research in Computing Science*, 112:19–28, 12 2016.

[10] The Centre for Speech Technology Research. The merlin toolkit. http://www.cstr.ed.ac.uk/projects/merlin/.

[11] Md Hasan, Mustafa Jamil, Golam Rabbani, and Md. Saifur Rahman. Speaker identification using mel frequency cepstral coe cients. *Proceedings of the 3rd International Conference on Electrical and Computer Engineering (ICECE 2004)*, 12 2004.

[12] HTS. http://hts.sp.nitech.ac.jp.

[13] Kaldi. https://kaldi-asr.org.

[14] Pijus Kasparaitis. Evaluation of lithuanian text-to-speech synthesizers. *Studies About Languages*, 28, 06 2016.

[15] Gražina Korvel, Virginija Šimonytė, and Vytautas Slivinskas. Lithuanian speech synthesis by computer using additive synthesis. *Elektronika ir Elektrotechnika*, 18:77–80, 01 2012.

[16] LIEPA. https://www.raštija.lt/liepa.

[17] Blaise Potard, Matthew Aylett, David Braude, and Petr Motlicek. Idlak tangle: An open source kaldi based parametric speech synthesiser based on dnn. pages 2293–2297, 09 2016.

[18] Lawrence Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 02 1986.

[19] Magdi Rashad, Hazem El-Bakry, Islam il, and Nikos Mastorakis. An overview of text-to-speech synthesis techniques. *International Conference on Communications and Information Technology - Proceedings*, 07 2010.

[20] Idlak source. https://github.com/Idlak/idlak.

[21] P. Taylor. Text-to-speech synthesis. *Text-to-Speech Synthesis*, pages 1–597, 01 2009.

[22] S. Umesh, Leon Cohen, and Douglas Nelson. Frequency warping and the mel scale. *Signal Processing Letters, IEEE*, 9:104 – 107, 04 2002.