Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas

Baigiamasis magistro projektas

Gediminas Norkus

Projekto autorius

Doc. dr. Kastytis Ratkevičius

Vadovas

Kaunas, 2020

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas

Baigiamasis magistro projektas

Valdymo technologijos (6211EX010)

Gediminas Norkus

Projekto autorius

Doc. dr. Kastytis Ratkevičius

Vadovas

Lekt. dr. Vytautas Gargasas

Recenzentas

Kaunas, 2020

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Gediminas Norkus

Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Gedimino Norkaus, baigiamasis projektas tema „Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka)	(parašas)

Norkus, Gediminas. Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas.
Magistro baigiamasis projektas / vadovas doc. dr. Kastytis Ratkevičius; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis – elektronikos inžinerija, krypčių grupė – inžinerijos mokslai.

Reikšminiai žodžiai: Kaldi, TensorFlow, kalbos atpažinimo sistema, akustinis modelis, kalbos modelis, paslėptasis Markovo modelis, neuroninis tinklas.

Kaunas, 2020. 56 p.

Santrauka

Darbo tikslas - ištirti *Kaldi ir TensorFlow* paketų pritaikymo ir funkcijų galimybes automatinio kalbos atpažinimo sistemose. Tyrimams buvo pasirinktas lietuviškų vardų garso įrašų rinkinys *VARDAI_18_3* – 21 diktoriaus balso įrašai su 22 lietuviškais vardais ir 4 daiktavardžiais, kiekvieno diktoriaus ištartais po 20 kartų. Bendras garso įrašų skaičius 10920. 18 diktorių įrašai naudojami apmokymui, o likusių 3 diktorių – testavimui. Vėliau atliktas kryžminis garsyno patikrinimas su užtriukšmintais balso įrašais, (signalo/triukšmo) lygis 5 dB. Baigiamąjame darbe palyginami to paties vardų garsyno rezultatai gauti su *Kaldi* ir *TensorFlow* programiniais paketais. Nagrinėjami *Kaldi* paketo atpažinimo metodai – *monofoninis, trifoninis, LDA+MLLT, SAT, SGMM2, TDNN-pnorm, TDNN-tanh* ir ieškoma, kuris metodas turi mažiausią žodžių atpažinimo paklaidą.

Pasiruošimas tyrimams prasideda nuo aprašomųjų *Kaldi* garsyno failų: *spk2gender.txt, utt2spk.txt, text.txt, wav.scp, corpus.txt,* kurie aprašo diktoriaus lytį, įrašų sąsajas su diktoriais, su tekstine transkripcija, realiu garso įrašu. Taip pat aprašomi užduoties failai: *cmd.sh, path.sh* ir *run.sh,* kuris yra pagrindinis programos paleidimo failas.

TensorFlow paketui paruošiami 6 programiniai failai: *generate_test_vrd.py, generate_train_eval_new.py, deep_speech.py, deep_speech_model.py, decoder.py, get_predictions.py.* Šie failai skirti duomenų paruošimui, kalbos modelio sudarymui, apmokymui, dekodavimui ir rezultatų gavimui.

Atlikus tyrimus su vardų garsynu ir *Kaldi* paketu, pastebėta, kad vardų atpažinimo paklaida tiriant neužtriukšmintus balso įrašus yra labai panaši, atsižvelgiant į skirtingus tyrimo metodus. To pasekoje, vardų garsyno įrašai užtriukšminti 5 dB lygio baltu triukšmu. Geriausias tyrimų rezultatas pasiektas *TDNN-tanh* metodu, atpažinimo tikslumas siekia 98,53 %. Tyrimai su *Tensorflow* paketu remiasi DeepSpeech2 modelio algoritmu ir neuroninių tinklų atpažinimo metodu. Geriausias atpažinimo tikslumas - 87,24 %. Darbo pabaigoje palyginami *Kald*i ir *TensorFlow* programinių paketų atpažinimo rezultatai.

Norkus, Gediminas. Creation and Investigation of Names Recognition System Using Kaldi and TensorFlow Packages. Master's Final Degree Project / supervisor doc. dr. Kastytis Ratkevičius; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field – electronics engineering, study field group – engineering science.

Keywords: Kaldi, TensorFlow, speech recognition system, acoustic model, language model, hidden Markov model , neural network.

Kaunas, 2020. 56 p.

Summary

The aim of this work is to investigate the possibilities of application and functions of *Kaldi* and *TensorFlow* packages in automatic speech recognition systems. The collection of audio recordings of Lithuanian names *VARDAI_18_3* was selected for the research - 21 voice recordings of the narrator with 22 Lithuanian names and 4 nouns, uttered by each narrator 20 times. The total number of audio recordings is 10920. The recordings of 18 narrators are used for training and the remaining 3 narrators for testing. This was followed by a cross-check of the sound with voice recordings, (signal / noise) level 5 dB. The final work compares the results of speech recognition of the same names obtained with the Kaldi and TensorFlow software packages. *Kaldi* package recognition methods are analyzed - monophone, triphonone, *LDA + MLLT, SAT, SGMM2, TDNN-pnorm, TDNN-tanh* and it is researched which method has the lowest word recognition error.

Preparation for the research begins with describing Kaldi descriptive sound files: *spk2gender.txt, utt2spk.txt, text.txt, wav.scp, corpus.txt*, who describe the gender of the narrator, the interfaces of the recordings with the narrators, with textual transcription and real audio recording. The task files are also described: *cmd.sh, path.sh,* and *run.sh,* which is the main program startup file.

6 software files are prepared for the TensorFlow package: *generate_test_vrd.py, generate_train_eval_new.py, deep_speech.py, deep_speech_model.py, decoder.py, get_predictions.py.* These files are used for data preparation, language modeling, training, decoding, and getting recognition results.

Studies with the name recordings and the Kaldi package have shown that the error of name recognition without any adittional noise is very similar given the different research methods. As a result, the sound recordings of the names were injected 5 dB of white noise. The best test result was obtained by *TDNN-tanh* method, the recognition accuracy is 98.53%. The studies with the *Tensorflow* package are based on the DeepSpeech2 model algorithm and the neural network recognition method. The best recognition accuracy is 87,24 %. At the end of the work, recognition results of *Kaldi* and *TensorFlow* are compared.

Turinys

Lentelių sąrašas **8**
Paveikslų sąrašas **9**
Santrumpų ir terminų sąrašas **11**
Įvadas **12**
1. Analitinė dalis **13**
1.1. Automatinis kalbos atpažinimas 13
1.2. Kalbos atpažinimo taikymai 13
1.3. Automatinio kalbos atpažinimo sistema 14
1.3.1. Požymių išgavimas 15
1.3.2. Akustinis modelis 15
1.3.3. Kalbos modelis 16
1.3.4. Kalbos dekodavimas 16
1.4. Fonemų rinkinių rūšys 17
1.5. Paslėptasis Markovo modelis 19
1.6. Gauso mišinių modeliai 20
1.7. Gilieji neuroniniai tinklai 20
1.8. Automatinio kalbos atpažinimo vertinimas 21
2. Metodinė dalis **22**
2.1 Programinės įrangos apžvalga 22
2.1.1 Linux (Ubuntu) operacinė sistema 22
2.1.2 Linux (Ubuntu) operacinės sistemos įdiegimas 22
2.2 Kaldi programinis paketas 23
2.2.1 Papildomų bibliotekų, skirtų Kaldi programiniam paketui, įdiegimas 24
2.2.2 Kaldi programinio paketo įdiegimas[15] 25
2.3 TensorFlow programinis paketas 25
2.3.1 TensorFlow programinio paketo įdiegimas 26
2.4 Spyder (Anaconda 3) programinė aplinka 26
2.5 Vardų garsyno aprašymas 27
2.6 Vardų garsyno paruošimas 27
2.6.1 Failas spk2gender, diktoriaus lyties nustatymas. 27
2.6.2 Failas utt2spk, garso įrašų sąsajos su diktoriais 27
2.6.3 Failas text, garso įrašų sąsajos su tekstine transkripcija. 28
2.6.4 Failas wav.scp, garso įrašų sąsajos su realiu garso įrašu. 28
2.6.5 Failas corpus.txt, aprašantis visas galimas transkripcijas. 28
2.7 Vardų garsyno fonemų aprašomieji failai 29
2.8 Konfigūraciniai failai 30
2.9 Užduoties failai 30
2.10 Garso įrašų užtriukšminimas naudojant „SoX“ įrankį 31
2.11 TensorFlow programinio paketo konfiguraciniai failai 32
3. Tyrimų rezultatai **35**
3.1 Vardų garsyno tyrimo su Kaldi paketu, rezultatai. 35
3.2 Vardų garsyno tyrimo su užtriukšmintais įrašais ir Kaldi paketu, rezultatai 37
3.3 Kryžminis patikrinimas 39

3.4 Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos priklausomybių nuo metodų parametrų tyrimai. 41
3.4.1 *Monofoninis metodas* 41
3.4.2 *Trifoninis metodas* 42
3.4.3 *LDA* metodas 44
3.4.4 *SAT* metodas 46
3.4.5 *SGMM2* metodas 48
3.4.6 *TDNN* metodai 50
3.5 Vardų garsyno tyrimo su TensorFlow paketu, rezultatai. 53
Išvados 54
Literatūros sąrašas 55

Lentelių sąrašas

1 lentelė. ASR sistemos žodžių atpažinimo klaidos 21
2 lentelė. Skirtingų modeliavimo metodų atpažinimo paklaidos 36
3 lentelė. Vardų garsyno atpažinimo klaida procentais 37
4 lentelė. Garsyno Vardai_18_3 atpažinimo klaida procentais, kai triukšmo lygis 5 dB 38
5 lentelė. Garso įrašų rinkinio Vardai_18_3-SNR5 kryžminio patikrinimo atpažinimo klaida procentais 40
6 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant monofoninį metodą 41
7 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (pirmas etapas) 42
8 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (antras etapas) 43
9 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (pirmas etapas) 44
10 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (antras etapas) 45
11 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (pirmas etapas) 46
12 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (antras etapas) 47
13 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (pirmas etapas) 48
14 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (antras etapas), rezultatai 49
15 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (trečias etapas) 49
16 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *pnorm* metodą 50
17 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo parametrų *pnorm_input_dim* ir *pnorm_output_dim*. 51
18 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *tanh* metodą 52
19 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo tikslumo priklausomybės nuo paslėptųjų sluoksnių ir neuronų juose skaičiaus, tyrimo rezultatai 53

Paveikslų sąrašas

1 pav. Automatinio kalbos atpažinimo sistemos struktūra 14
2 pav. *Lexicon1.txt* fonemų rinkinio su dvibalsiais failas 17
3 pav. *Lexicon_graf.txt* fonemų rinkinio su grafemomis failas 18
4 pav. Paslėptuoju Markovo modeliu paremtas fonemų modelis 19
5 pav. Gilusis neuroninis tinklas 20
6 pav. OS grafinė aplinka - operacinės sistemos įdiegimas 22
7 pav. Ubuntu *OS* įrašymo parinktys 23
8 pav. *Kaldi* programinio paketo architektūra 24
9 pav. Terminalo komanda, naudojama Kaldi programinio paketo atsisiuntimui. 25
11 pav. *TensorFlow* programinio paketo aktyvavimo komandos 26
12 pav. *TensorFlow* programinio paketo GPU versijos aktyvavimo komandos 26
13 pav. *Spyder* programinės aplinkos įdiegimo komanda 27
14 pav. *Spk2gender* failo fragmentas. 27
15 pav. *Utt2spk* failo fragmentas. 28
16 pav. *Text* failo fragmentas. 28
17 pav. *Wav.scp* realaus garso failų sąrašo fragmentas. 28
18 pav. *Lexicon.txt* failo fragmentas. 29
19 pav. *Nonsilence_phones.txt.* Fonemų (grafemų) sąrašo fragmentas. 29
20 pav. *Silence_phone.txt.* failas. 29
21 pav. *Optional_silence.txt.* Papildomų tylos fonemų failas. 30
23 pav. *Mfcc.conf* proceso nustatymų failas 30
24 pav. *Cmd.sh* failo nustatymai 30
25 pav. *Path.sh* failo nustatymai 31
26 pav. *Run.sh* pagrindinio failo fragmentas 31
27 pav. „Sox" komandinė eilutė, skirta garso įrašų užtriukšminimui 31
28 pav. G*enerate_test_vrd.py* failo fragmentas. 32
29 pav. *Test_set.csv* failo fragmentas. 32
31 pav. *Deep_speech.py* failo fragmentas. 33
32 pav. D*eep_speech_model.py* failo fragmentas. 33
33 pav. D*ecoder.py* failo fragmentas. 33
34 pav. *Get_predictions.py* failo fragmentas. 34
35 pav. Skirtingų tyrimo metodų rezultatai 37
36 pav. Skirtingų tyrimo metodų su užtriukšmintais garso įrašais, rezultatai 38
37 pav. Skirtingų tyrimo metodų, kryžminio patikrinimo rezultatai 40
38 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant monofoninį metodą, rezultatai 42
39 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (pirmas etapas), rezultatai 43
40 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (antras etapas), rezultatai 44
41 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (pirmas etapas), rezultatai 45
42 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (antras etapas), rezultatai 46

43 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (pirmas etapas), rezultatai 47
44 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (antras etapas), rezultatai 47
45 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (pirmas etapas), rezultatai 48
46 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (antras etapas), rezultatai 49
47 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (trečias etapas), rezultatai 50
48 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *pnorm* metodą, rezultatai 51
49 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo parametrų *pnorm_input_dim* ir *pnorm_output_dim*, rezultatai 51
50 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *tanh* metodą, rezultatai 52
51 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo paslėptų sluoksnių skaičiaus ir neuronų skaičiaus, rezultatai 53

Santrumpų ir terminų sąrašas

Santrumpos:

Kaldi – programinis paketas;

TensorFlow – programinis paketas;

ASR – automatinis kalbos atpažinimas;

MFCC - Melų dažnių skalės kepstriniai koeficientai (angl. Mel Frequency Cepstral Coefficient);

HMM - Paslėptasis Markovo modelis (angl. Hidden Markov Model);

GMM – Gauso mišinio modelis (angl. Gaussian Mixture Model);

SGMM – tarpinis Gauso mišinių modelis (angl. Subspace Gaussian Mixture Model);

DNN – gilusis neuroninis tinklas (angl. Deep Neural Network);

WER – klaidingų žodžių rodiklis (angl. Word Error Rate);

CER – klaidingų simbolių rodiklis (angl. Character Error Rate);

SER – klaidingų sakinių rodiklis (angl. Sentence Error Rate);

LDA – linijinė diskriminantinė analizė (angl. Linear Discriminant Analysis);

MLLT – didžiausios tikimybės linijinė transformacija (angl. Maximum Likelihood Linear Transform);

SAT – prisitaikantis diktoriaus mokymas (angl. Speaker Adaptive Training);

TDNN – laiko uždelsimo neuroninis tinklas (angl. Time Delay Neural Network);

Ubuntu – Unix tipo operacinė sistema;

Sox – garso įrašų užtriukšminimo programinis įrankis.

Įvadas

Šiomis dienomis žmonių gyvenimas be išmaniųjų technologijų atrodytų neįmanomas. Kasdieninės veiklos, operacijos ar veiksmai neapsieina be automatinio valdymo technologijų. Viena iš šių technologijų sričių yra kalbos, balso atpažinimas. Balso atpažinimas atliekamas kompiuteriu ir tam tikrais atvirojo kodo programiniais paketais, pvz. *Kaldi*, *TensorFlow*. Iki šiol kalbos atpažinimas buvo vystomas pagrindinėmis pasaulio kalbomis, tačiau lietuvių kalbos atpažinimas tirtas nedaug.

Darbo tikslas - nustatyti vardų garsyno atpažinimo tikslumą naudojantis Kaldi ir TensorFlow programiniais paketais.

Darbo uždaviniai:

1. atlikti literatūros analizę ir išsiaiškinti kokios programinės įrangos gali būti naudojamos kalbos atpažinimui;
2. išsiaiškinti kaip veikia automatinio kalbos atpažinimo modelis;
3. išsiaiškinti kalbos atpažinimo metodus;
4. nustatyti būdą, kaip vertinti automatinio kalbos atpažinimo modelių efektyvumą;
5. ištirti *Kaldi* ir *Tensorflow* programinių paketų veikimo principą ir jų panaudojimo galimybes;
6. paruošti operacinę sistemą, tinkamą *Kaldi ir TensorFlow* programiniams paketams;
7. paruošti vardų garsyną *Kaldi* ir *TensorFlow* paketų tyrimams;
8. išanalizuoti, kuriuo programiniu paketu ir kalbos atpažinimo metodu gaunama mažiausia garsyno atpažinimo paklaida;
9. pateikti gautus rezultatus ir išvadas.

Darbo objektas ir metodai. Kaldi ir TensorFlow programiniams paketams apmokomas kalbos atpažintuvas, panaudojant 21 diktoriaus lietuviškų vardų garsyną. Kompiuteriui įdiegiama Ubuntu operacinė sistema ir papildomi plėtiniai, reikalingi programinių paketų veikimui. Kalbos atpažinimo bandymai atliekami *monofininiu, trifoniniu, LDA+MLLT, SAT, SGMM* 2, *TDNN-pnorm, TDNN-tanh* metodais.

Temos aktualumas. Šiuo baigiamuoju darbu siekiama ištirti lietuviškų vardų garsyną, naudojantis pažangiais programiniais paketais, o gauti tyrimų rezultatai gali būti panaudoti tolimesniame ASR sistemų projektavime, siekant įgyvendinti lietuvių kalbos atpažinimą sudėtingose ir moderniose kalbos atpažinimo sistemose.

1. Analitinė dalis

Šiame skyriuje yra aprašoma kas yra automatinis kalbos atpažinimas, iš ko sudaryta automatinio kalbos atpažinimo sistema, jos modelis, taikymo galimybės. Taip pat aprašomos fonemų rinkinių rūšys, paslėptasis Markovo modelis, gilieji neuroniniai tinklai, bei automatinio kalbos atpažinimo vertinimas.

1.1. Automatinis kalbos atpažinimas

Automatinis kalbos atpažinimas (angl. *Automatic Speech Recognition,* ASR) – procesas ir su juo susijusi technologija, skirta kalbos signalui paversti jį atitinkančia žodžių seka ar kitomis kalbinėmis esmėmis, naudojant algoritmus, įdiegtus įrenginyje ar kompiuteryje.

ASR tyrimai atliekami jau daugiau nei 60 metų. Pramonė sukūrė platų komercinių produktų asortimentą, kuriame ASR, kaip vartotojo sąsaja tapo vis naudingesnė ir labiau paplitusi. Vartotojams orientuotos programos vis labiau reikalauja, kad ASR atitiktų realaus pasaulio triukšmo ir kitų garsą iškraipančių veiksnių sąlygas. Vis dėlto patikimas ištartų žodžių atpažinimas realistinėje akustinėje aplinkoje vis dar yra iššūkis.

Panašiai, kaip ir bet kurį atpažinimo procesą, kalbos atpažinimą galima išskaidyti į tris pagrindinius etapus: *duomenų įvedimą, požymių išgavimą ir atpažinimą.* Pirmiausia mikrofonu kalbos signalas įvedamas, o tada paverčiamas skaitmeniniu pavidalu. Kadangi kalbos signalą charakterizuoja trys dydžiai: laikas, dažnis ir amplitudė, tai antrame etape fiksuotais laiko momentais imami tam tikro ilgio kalbos signalo kadrai, kuriuose skaičiuojami tam tikrų dažnio juostų energiją charakterizuojantys parametrai. Šie parametrai naudojami trečiame atpažinimo (klasifikacijos) etape, kuriame nustatoma, kokios fonemos ar žodžiai yra kalbos signale [1].

Kalbos atpažinimas yra labai sudėtingas uždavinys. Prireikė trisdešimties metų, kol atsirado pirmosios praktiškai naudojamos sistemos. Uždavinio sudėtingumą nulemia tokios priežastys:

1. Keletą kartų ištarto tam tikro garso akustinė realizacija labai skiriasi, net jei jį ištarė tas pats diktorius ir tame pačiame žodyje;
2. Kalbėjimo greitis gali labai kisti, todėl skiriasi kelių to paties žodžių akustinių realizacijų ilgis. Kintant žodžių ilgiui atskirų garsų ilgis kinta netiesiškai;
3. Garso akustinė realizacija priklauso nuo gretimų garsų.
4. Kalbėjimo sraute nėra aiškių garsų ar žodžių ribų.
5. Kiekvieno žmogaus tartis yra skirtinga, todėl reikalingas arba apmokymas konkrečiam diktoriui, arba sistema kūrimo metu turi būti apmokyta su kuo didesniu diktorių skaičiumi;
6. Jei kuriama atpažinimo sistema remiasi žodžių atpažinimu, žodžių etalonų skaičius gali būti pernelyg didelis;
7. Kalbėjimo sraute gali būti ir nekalbinių fragmentų (pvz., kosulys), kuriuos reikia atskirti ir pašalinti;
8. Praktiniuose taikymuose papildomų problemų sukelia foninis triukšmas.

1.2. Kalbos atpažinimo taikymai

Šiame poskyryje trumpai apžvelgiamos automatinio kalbos atpažinimo taikymo sritys, bei panaudojimo galimybės.
ASR sistemų taikymo sritys:

- orlaiviai ir kosmosas (pvz., kosmoso tyrimai, kosminiai laivai ir kt.);
- automatinis subtitravimas su kalbos atpažinimu;
- automatinis emocijų atpažinimas;
- automatinis vertimas;
- teismo ataskaitos (kalbų rašymas realiu laiku);
- laisvų rankų įranga: kalbos atpažinimo kompiuterinė vartotojo sąsaja;
- namų automatizavimas;
- interaktyvus balso atsakas;
- mobilioji telefonija, įskaitant mobilųjį elektroninį paštą;
- tarimo vertinimas kompiuterinėmis kalbų mokymosi programomis;
- parašymas realiu laiku;
- robotika;
- telematika (pvz., transporto priemonių navigacijos sistemos);
- transkripcija (skaitmeninis kalbėjimas tekstu);
- vaizdo žaidimai;
- virtualus asistentas (pvz., „apple siri“).

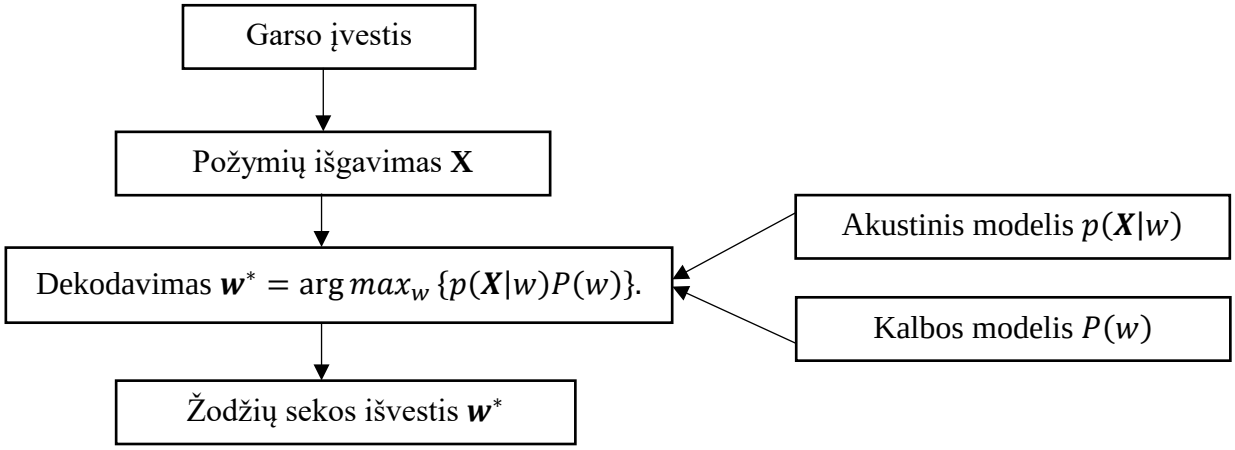
1.3. Automatinio kalbos atpažinimo sistema

ASR modelio tikslas yra surasti labiausiai tikėtiną žodžių seką, atsižvelgiant į duotą kalbos įvestį. Labiausiai tikėtinos žodžių sekos radimo procesas vadinamas *kalbos dekodavimu*. Formaliau, dekodavimo procesas gali būti apibrėžiamas kaip labiausiai tikėtinos žodžių sekos $\boldsymbol{w}^*$ radimas, atsižvelgiant į akustinių savybių **X** rinkinį [2],

$$\boldsymbol{w}^* = \arg max_w \{P(w|\boldsymbol{X})\}. \quad (1)$$

Tiesiogiai suskaičiuoti $P(w|\boldsymbol{X})$ yra sudėtinga, todėl šiuo atveju pritaikoma Bayes' taisyklė ir išraiška tranformuojama į:

$$\boldsymbol{w}^* = \arg max_w \left\{\frac{p(\boldsymbol{X}|w)P(w)}{P(\boldsymbol{X})}\right\} = \arg max_w \{p(\boldsymbol{X}|w)P(w)\}. \quad (2)$$



1 pav. Automatinio kalbos atpažinimo sistemos struktūra

Labiausiai tikėtina žodžių seka nepriklauso nuo akustinių savybių P(**X**) tikimybės ir yra pašalinama paskutiniame (2) lygties žingsnyje. Akustinio modeliavimo užduotis – įvertinti modelio parametrus θ, kad $p(\boldsymbol{X}|w;\theta)$ būtų kuo tikslesni. Ankstesnis P(**w**) yra tai, ką nustatys kalbos modelis. Automatinio kalbos atpažinimo struktūra ir garso įvesties dekodavimas yra parodyti 1 pav. Pirmiausia garso įrašai imami ir apdorojami siekiant gauti funkcijų vektorių **X**, kuris yra dekoderio įvestis. Akustiniai požymiai skaičiuojami 20-30 ms ilgio kadrų sekomis, paimtomis iš garso įvesties.

Kiekvienam kadrui gaunamas požymių vektorių rinkinys. Dekodavimas atliekamas kadras po kadro naudojant „spindulio“ paiešką. Ši paieška išplečia kadrų hipotezę, naudojant informaciją iš ankstesnių kadrų, kai juda į priekį laiku į kitą kadrą. Hipotezė apibrėžiama kaip akustinio modelio išvestis, t. y. tikėtina žodžių seka. Hipotezių tikimybės apskaičiuojamos naudojant akustinį modelį ir kalbos modelį. Kai pasiekiamas paskutinis kadras, visa hipotezė, atitinkanti didžiausią tikimybę, yra tai, kas duos žodžių sekos išvestį **w***.

1.3.1. Požymių išgavimas

Požymių išgavimas yra vienas iš svarbiausių kalbos atpažinimo uždavinių problemų, kuriuo siekiama išspręsti matmenų problemą. Kaip akustinuose duomenyse esanti svarbi informacija gali būti fiksuojama veiksmingiausiu būdu? Vienas iš sprendimų yra apskaičiuoti angl. (*Mel-Frequency Cepstral)* koeficientus (MFCCs). MFCC koeficientai buvo pristatyti S. Davis ir P. Mermelstein [3] 1980-aisiais ir buvo naudojami kalbos atpažinimo užduotims nuo to laiko. Mel-skalė yra dažnių suvokimo skalė. Ja siekiama modeliuoti žmogaus ausies jautrumą ir ji buvo sukurta remiantis eksperimentais, kuriuose savanoriai buvo apklausiami, kaip jie suvokia tam tikrą dažnį, pavyzdžiui, jiems liepta pakoreguoti toną, kad jis atitiktų pusę duotojo tono garso lygio.

Norint apskaičiuoti MFCC koeficientus, garso įrašas suskaidomas į 20-30 ms ilgio kadrus naudojant kadrų funkciją, kurios reikšmė visur yra nulis, išskyrus mažame regione. Diskretizuotas kalbos signalas žymimas $S_i(n)$, kur i reikšmė kinta nuo kadrų skaičiaus ir n reikšmė kinta nuo bandymų skaičiaus. Kalbos signalo periodograma apskaičiuojama atsižvelgiant į angl. (*Discrete Fourier Trasnform)* (DFT) modulio kvadratą,

$$P_i\,(k) = \frac{1}{N}\left|\sum_{n=0}^{N-1} s_i(n)h(n)e^{-j2\pi kn/N}\right| 2 \quad 0 \le k \le N,$$

kur $h(n)$ yra N pavyzdžio analizės kadras [4] ir N yra DFT dydis.

1.3.2. Akustinis modelis

Akustinis modelis (AM) įvertina tikimybę $p(\boldsymbol{X}|\boldsymbol{w};\theta)$. Modelio parametrai θ yra randami atliekant apmokymą. Apmokyme yra tam tikras neapibrėžtumas - laiko suderinimas ištarime nėra žinomas. Paslėptasis Markovo modelis yra dažnas pasirinkimas kalbos atpažinime, dėl sugebėjimo modeliuoti šį neapibrėžtumą tarp akustinių savybių ir atitinkamos transkripcijos.

Norint suprasti natūralią kalbą ir susidoroti su didelės apimties žodynų atpažinimo užduotimis, žodis - nėra tinkamas apmokymo vieneto pasirinkimas. Žodžių, kurie turėtų būti žinomi pagal modelį,

skaičius, sukeltų problemų, kurios būtų per sunkios, kad jas išspręstume. Vietoj to naudojama *fonema* ir apibrėžiama kaip mažiausias kalbos vienetas.

1.3.3. Kalbos modelis

$P(w)$ koeficientas nustatomas pagal kalbos modelį. Kalbos modelyje pateikiama informacija apie žodžių pasikartojimo tikimybę. Ankstesnė tikimybė $P(w)$ žodžių sekai $w = w_1, \dots, w_k$ surandama pagal formulę:

$$P(w) = \prod_{k=1}^{K} P(w_k | w_{k-1}, \dots, w_1).$$

Didelio žodyno atpažinimo atveju žodžio tikimybė dažnai modeliuojama naudojant *n- gramų* kalbos modelį. *N-gramo* modelyje daroma prielaida, kad žodžio atsiradimo tikimybė priklauso nuo prieš tai esančių žodžių skaičiaus. Dideliam žodynui *n* paprastai svyruoja nuo dviejų iki keturių.

$$P(w) = \prod_{k=1}^{K} P(w_k | w_{k-1}, w_{k-2}, \dots w_{k-n+1}).$$

Modelis apmokomas skaičiuojant žodžių sekų įvykius mokymo duomenyse ir surandant tikimybes pagal didžiausią tikimybę.

$$P(w_k | w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2} w_{k-1} w_k)}{C\,(w_{k-2} w_{k-1})},$$

kur $C(w_{k-2} w_{k-1} w_k)$ nurodo bendrą žodžių sekos $w_{k-2} w_{k-1} w_k$ atvejų skaičių duomenyse ir $C\,(w_{k-2} w_{k-1})$ atitinkamai bendrą $w_{k-2} w_{k-1}$ pasikartojimų skaičių.

1.3.4. Kalbos dekodavimas

Kalbos dekodavimo uždavinys yra sprendžiamas pagal lygtį:

$$\boldsymbol{w}^* = \arg\max_{w} \{P(w|\boldsymbol{X})\}.$$

Dekoderis ieško fonemų sekų, atitinkančių žodžius. Fonemos paprastai vaizduojamos kaip trifonai akustiniame modelyje, o žodžių hipotezė yra ribojama žodžių, esančių fonetiniame žodyne. Kalbos modelis patvirtina akustinio modelio žodžių sekos hipotezę. Dekodavimo procese kalbos modelio poveikį reguliuoja *kalbos modelio svoris*, angl. *(Language Model Weight) wlm*. Tada lygtis tampa:

$$w^* = \arg\max_{w} \{P(w|\boldsymbol{X})\} = \arg\max_{w} \{p(\boldsymbol{X}|w) P(w)^{wlm}\}. \quad (3)$$

Rasti atitinkamą žodžių seką, kuri maksimaliai išnaudoja lygtį (3) yra paieškos uždavinys, kurio sprendimas yra dekoderio domenas.

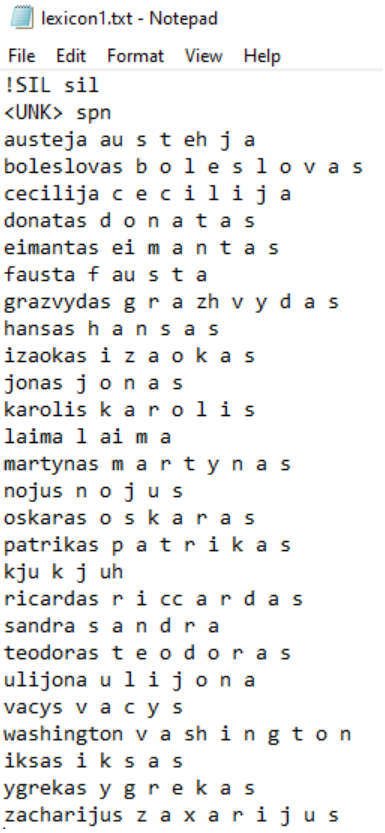
1.4. Fonemų rinkinių rūšys

Fonema – tai mažiausias kalbos vienetas, turintis skiriamąją (distinktyvinę) funkciją. Kiekvieną žodį sudaro fonemų seka. Lietuvių kalba turi 12 balsinių, 10 dvibalsinių ir 45 priebalsines fonemas.

Šio baigiamojo darbo tyrimams buvo panaudoti trys fonemų rinkiniai su *dvibalsiais, grafemomis ir SAMPA_LT.*
Pagal kalbos garsų tarimą, fonemas sudaro: *balsiai, priebalsiai* ir *dvigarsiai.*
Glaudus dviejų garsų junginys yra vadinamas *dvigarsiu*. Dvigarsiai gali būti:

- Grynieji priebalsiniai: ts, t’s’, dz, d’z’, tš, t’š’, dž.
- Grynieji balsiniai: ie, uo, ai, au, ei, ui.
- Mišrieji: al, am, an, ar, el, em, en, er,il, im, in, ir, ul, um, un, ur [5].

Vardų garsynui sudaromas fonemų rinkinio su dvibalsiais failas, kuris pateikiamas 2 pav.



2 pav. *Lexicon1.txt* fonemų rinkinio su dvibalsiais failas

Grafema [6] – rašto sudėtinė dalis, svarbiausias jo vienetas. Jis lygintinas su fonema, kuri yra svarbiausia kalbos fonetinės sistemos dalis. Visas grafemas galima skirstyti į keletą rūšių pagal jų kilmę ir funkciją:

- Grafemos, skirtos žodžiams (leksemoms). Pati gausiausia grafemų grupė. Šios grafemos pagal kilmę gali būti skirstomos į šias rūšis:
- Raidės – fonetiniu pagrindu sudarytos grafemos, turinčios tik fonetinę vertę. Tos rašto sistemos, kur vyrauja raidės, yra vadinamos abėcėlėmis, abdžadais ir abugidomis.

- Silabogramos – irgi fonetiniu pagrindu sudarytos grafemos, žyminčios skiemenis.
- Determinatyvai – grafemos, išreiškiančios tam tikrą idėją, parodančios daikto kategoriją, bet neturinčios fonetinės vertės, t. y., netariamos.
- Mišrios – grafemos, turinčios skirtingos kilmės elementų, kaip kad majų glifai.
- Skaitmenys – grafemos, skirtos išreikšti skaitinėms vertėms. Savo prigimtimi jos visos yra ideogramos.
- Skyrybos ženklai – grafemos, neturinčios tarimo, bet galinčios parodyti kalbos intonaciją.

Raidinėse rašto sistemose neretai naudojami grafemų junginiai, kurie išreiškia vieną garsą. Tokie junginiai yra vadinami *digrafais* (*sh, ch, ll* ar *ff* anglų kalboje, *ch* lietuvių kalboje), trigrafais (*sch* vokiečių kalboje), tetragrafais (*tsch* vokiečių kalboje), pentagrafais (*tszch* vokiškame žodyje *Nietszche*) ir pan. Bet lygiai taip pat viena grafema gali išreikšti garsų junginį: pvz., grafema *x* anglų kalboje išreiškia garsų junginį *ks*.Vardų garsynui sudaromas fonemų rinkinio su grafemomis failas, kuris pateikiamas 3 pav.

lexicon_graf.txt - Notepad

File Edit Format View Help

```
!SIL sil
<UNK> spn
austeja a u s t eh j a
boleslovas b o l e s l o v a s
cecilija c e c i l i j a
donatas d o n a t a s
eimantas e i m a n t a s
fausta f a u s t a
grazvydas g r a zh v y d a s
hansas h a n s a s
izaokas i z a o k a s
jonas j o n a s
karolis k a r o l i s
laima l a i m a
martynas m a r t y n a s
nojus n o j u s
oskaras o s k a r a s
patrikas p a t r i k a s
kju k j uh
ricardas r i cc a r d a s
sandra s a n d r a
teodoras t e o d o r a s
ulijona u l i j o n a
vacys v a c y s
washington v a sh i n g t o n
iksas i k s a s
ygrekas y g r e k a s
zacharijus z a x a r i j u s
```

3 pav. *Lexicon_graf.txt* fonemų rinkinio su grafemomis failas

SAMPA (*Speech Assessment Methods Phonetic Alphabet*) yra mašininiu būdu nuskaitoma fonetinė abėcėlė, skirta tarptautinio fonetinio alfabeto (angl. *International Phonetic Alphabet (IPA)*) simbolius paversti į ASCII kodavimą. SAMPA transkripcija sukurta taip, kad ją būtų galima analizuoti. Kaip ir įprasto IPA atveju, SAMPA simbolių eilutėje nereikia tarpo tarp vienas po kito einančių simbolių.

1987–89 m. - tarptautinės fonetikų grupės, pirmiausia pritaikė šią abėcėlę Europos Bendrijų kalboms: danų, olandų, anglų, prancūzų, vokiečių ir italų. Vėliau norvegų ir švedų kalboms (1992 m.), graikų, portugalų ir ispanų kalboms (1993 m.). Vykdant projektą BABEL, SAMPA išplėstas ir Bulgarijos, Estijos, Vengrijos, Lenkijos, Rumunijos ir iš esmės visoms europos kalboms.[7]

Pagal A. Raškinio, G. Raškinio ir A. Kazlauskienės straipsnį [8], lietuviškas SAMPA turi atitikti tam tikrus kriterijus:

- *Fonetiškai skaidomas.* Lietuviški priebalsiai turi būti minkšti arba kieti, balsiai – ilgi arba trumpi. Kirčio ženklai: kairinis, dešininis ir riestinis.
- *Įskaitomas*. SAMPA paremtos transkripcijos turi būti lengvai įskaitomos žmogui. SAMPA kodai turi būti panašūs į lietuvių kalboje naudojamus abėcėlės simbolius: a, ą, b, c, č, d, e, ę, ė, f, g, h, i, į, y, j, k, l, m, n, o, p, r, s, š, t, u, ū, ų, v, z, ž.
- *Suderinamas su IPA.* Lietuviški tekstai turi būti iššifruoti naudojant tradicinius IPA simbolius.
- *Vienareikšmiškas.* Didelis dėmesys skiriamas aprašant dvibalsius (ui, ie, uo, ai, au, ei,), mišriuosius dvigarsius (al, am, an, ar, el, em, en, er, il, im, in, ir, ul, um, un, ur) ir sutaptinius priebalsius (c, č, dz, dž).
- *Nepriklausomas nuo vienetų.* Fonetinė informacija, atitinkanti garso segmentą, turi būti lokaliai izoliuota ir užkoduota.

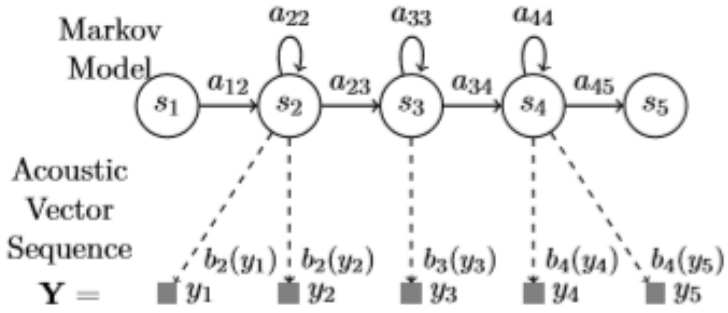
SAMPA sistemoje, vietoj lietuviškų kirčio ženklų yra naudojami simboliai, pvz. pučiamieji *š* ir *ž* yra užkoduojami simboliais *sh* ir *zh*, o *č* raidė užkoduojama simboliu *ch*. Priebalsių minkštumas žymimas po fonetinio vieneto rašant „1“, pvz., m' – m1.

1.5. Paslėptasis Markovo modelis

Paslėptasis Markovo modelis (angl. Hidden Markov Model) yra statistinis modelis. Kalbos atpažinime paslėptasis Markovo modelis suteikia statistinį žodžių garsų atvaizdavimą. Jis susideda iš būsenų grandinės. HMM dabartinė būsena yra paslėpta ir galima pastebėti tik kiekvienos būsenos išeigą. Kiekviena HMM būsena atitinka vieną kadrą akustinėje įvestyje.

Modelio parametrai, įvertinami akustiniame apmokyme, $\theta = [\{a_{ij}\},\{b_j()\}]$, kur $\{a_{ij}\}$ atitinka perėjimo tikimybes ir $\{b_j()\}$ – išvesties stebėjimo paskirstymus. Perėjimo tikimybė a_{ij} yra tikimybė, kad perėjimas iš būsenos *i* į būseną *j*. Svarbus HMM bruožas yra savarankiškai besikartojantis a_{ii}, todėl HMM gali modeliuoti skirtingų ilgių fonemas. Atliekant perėjimą ir įeinant į naują HMM etapą, sugeneruojamas funkcijos vektorius naudojant paskirstymo, susijusio su tos konkrečios būsenos HMM.

Pirmoji HMM būsena ir paskutinė būsena vadinamos nespinduliuojančiomis būsenomis. 4 pav. pavaizduota *s1* yra įėjimo būsena, o *s5* - išėjimo būsena. Jos naudojamos kaip įėjimas ir išėjimas iš modelio ir supaprastina HMM sujungimą, bei fonemų modelius, kad sudarytų žodžius.



4 pav. Paslėptuoju Markovo modeliu paremtas fonemų modelis

1.6. Gauso mišinių modeliai

Gauso mišinys yra dažnas pasirinkimas sprendžiant HMM uždavinį ir ieškant išėjimo reikšmių. Kalbėtojo, akcento ir lyčių skirtumai kalbos duomenyse paprastai sukuria kelis režimus. Gauso mišinio modeliai gali apibūdinti tokį daugiarūšiškumą, todėl GMM laikomi galinga kalbos atpažinimo priemone [9]. Kalbėjimo atpažinimas, pagrįstas paslėpto markovo modelio-Gauso mišinio modelio (HMM-GMM) sistema, paprastai apima visiškai atskiro GMM mokymą kiekvienoje HMM būsenoje.

1.7. Gilieji neuroniniai tinklai

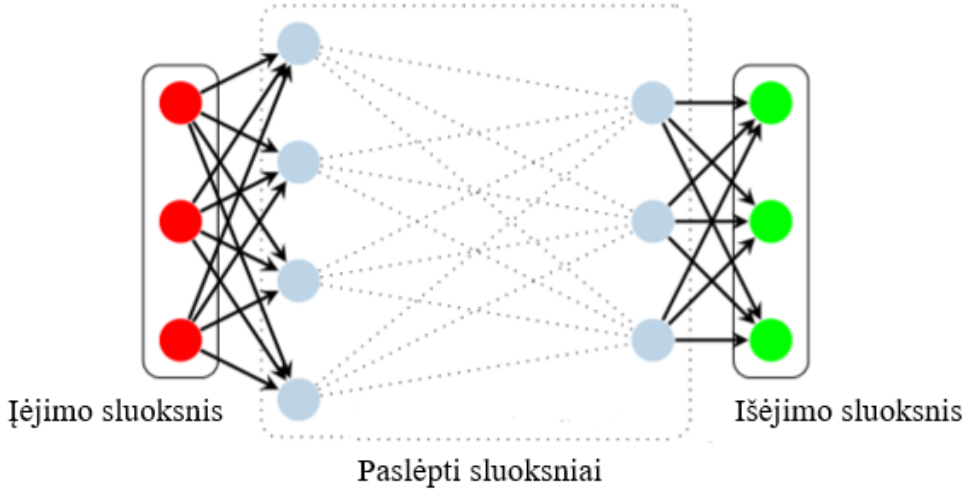
Gauso mišinių alternatyva yra gilieji neuroniniai tinklai, angl. (DNN) [10]. Gilusis neuroninis tinklas yra į priekį orientuotas, dirbtinis neuroninis tinklas, kuriame yra daugiau nei vienas paslėptas sluoksnis tarp įvesties sluoksnio ir išvesties sluoksnio, kaip parodyta 5 paveiksle. Mazgai išilgai kelio turi svorius, pritvirtintus prie jų, o kiekvieno mazgo išvestis apskaičiuojama pagal *aktyvinimo funkciją* a_{ut}. Paprastai mazgo įėjimas, esantis DNN sluoksnyje, apskaičiuojamas iš žemiau esančio sluoksnio pagal formulę,

$$x_j = b_j + \sum_i y_i w_{ij}$$

, kur b_j yra j paklaida, i yra vienetų indeksas, esantis žemesniame sluoksnyje ir w_{ij} yra jungties svoris iš reikšmės j į reikšmę i žemesniame sluoksnyje. Išėjimas į aukštesnį sluoksnį apskaičiuojamas pagal formulę,

$$y_j = a_{ut}(x_j).$$

Paslėpti sluoksniai priverčia gilųjį neuroninį tinklą modeliuoti netiesinius, sudėtingus ryšius duomenyse. Daugiaklasiai klasifikacijai, išvesties vienetas j konvertuoja savo bendrą įvestį x_j į tikimybę, naudojant „softmax“ funkciją.



5 pav. Gilusis neuroninis tinklas

Apmokymo tikslas yra optimizuoti *objektyvią funkciją* ir atnaujinti vidinių mazgų svorius, remiantis tam tikra informacija, perduodama modeliui. Vienas svarbus parametras apmokymo procese yra *mokymosi vertė*. Kuo didesnė mokymosi vertė, tuo greitesnis, bet mažiau tikslus tampa mokymas.

1.8. Automatinio kalbos atpažinimo vertinimas

Automatinio kalbos atpažinimo modelių efektyvumui palyginti naudojamas matavimas, vadinamas klaidingų žodžių rodikliu, angl. *Word Error Rate (WER).* Šis matavimas atliekamas žodžių lygmenyje, naudojantis *Levenšteino atstumu* tarp žodžių, kad būtų surastas atitikmuo ir rezultatas.

Levenšteino atstumas yra minimalus *atpažinimų, ištrynimų* ir *įterpimų* skaičius [11]. Paprastoje metrikoje visos šios operacijos vertinamos vienodai ir jų reikšmė lygi vienetui. Toliau pateikiamas žodžių palyginimas tarp nuorodos eilutės ir galimos ASR išvesties eilutės pavyzdys, 1 lentelė. Klaidos žymimos raidėmis S, D, I, o teisingai atpažinti žodžiai raide C.

1 lentelė. ASR sistemos žodžių atpažinimo klaidos

Žodis	Austeja	Boleslovas	Cecilija		Eimantas	Fausta	Grazvydas	Hansas	Izaokas
Išėjimas	Austeya	***	Cecilija	ai	Eimatas	Fausta	Grazvydas	Hansas	Izaokas
Klaida	S	D	C	I	S	C	C	C	C

Atlikus palyginimą, klaidingų žodžių rodiklis (WER) apskaičiuojamas pagal formulę:

$$WER = \frac{S_{vis} + D_{vis} + I_{vis}}{N}$$

, kur

S_{vis} yra visų neteisingai atpažintų žodžių skaičius,
D_{vis} yra visų ištrintų žodžių skaičius,
I_{vis} yra visų įterptų žodžių skaičius,
C_{vis} yra visų teisingai atpažintų žodžių skaičius,
N yra visų ištartų žodžių skaičius $(N = S_{vis} + D_{vis} + I_{vis})$.

Taigi, duotam pavyzdžiui, kur $S_{vis} = 2, D_{vis} = 1, I_{vis} = 1, C_{vis} = 5$, WER lygus:

$$WER = \frac{2 + 1 + 1}{2 + 1 + 5} = \frac{4}{8} = 0.5$$

Klaidingų žodžių rodiklis (WER) gal ir nėra optimalus matavimo vienetas matuojant ASR sistemos kokybę, tačiau, jo pakanka modelių našumui palyginti ir bendram tikslumui įvertinti.

2. Metodinė dalis

2.1 Programinės įrangos apžvalga

Šiame skyriuje aprašomos programinės įrangos, naudotos vardų garsyno tyrimams atlikti.

2.1.1 Linux (Ubuntu) operacinė sistema

Linux – laisvos (atviro kodo) operacinės sistemos branduolio (kernel) pavadinimas. Dažnai taip sutrumpintai vadinama ir bendrai visa Unix tipo operacinė sistema naudojanti Linux branduolį (kuris buvo išleistas pirmą kartą 1991 m. spalį), sukurtą Linus Torvalds. Yra daugybė GNU/Linux variantų, dar vadinamų platinamaisiais paketais (angl. distribution). Populiariausias iš jų dabar yra Ubuntu.

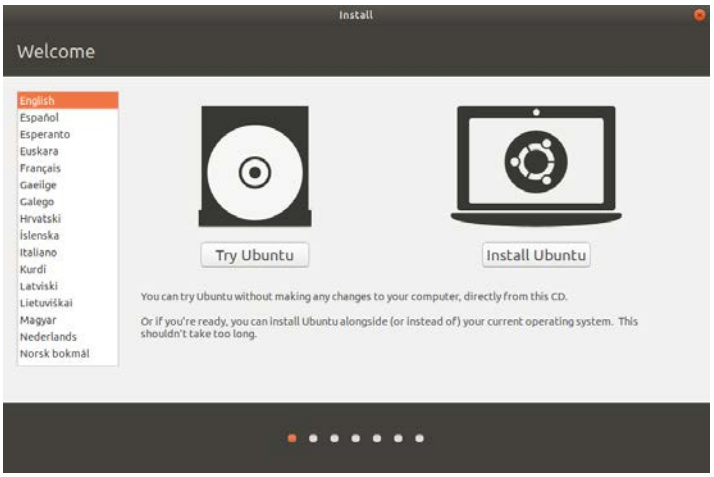
Ubuntu pasižymi tokiomis savybėmis:

- Daugiaprocesė (multitasking). Operacinė sistema gali vykdyti kelias programas, užduotis ar procesus vienu metu, paskirstydama kompiuterio resursus (pvz. centrinio procesoriaus).
- Lanksti – sistemą galima konfigūruoti pagal savo poreikius.
- Saugi – apsaugo kompiuterį nuo nepageidaujamų kenkėjiškų programų.

Pastaroji dabar naudojama serveriuose, asmeniniuose kompiuteriuose ir superkompiuteriuose, taip pat – mobiliuose telefonuose ir kitose buitinėse sistemose.

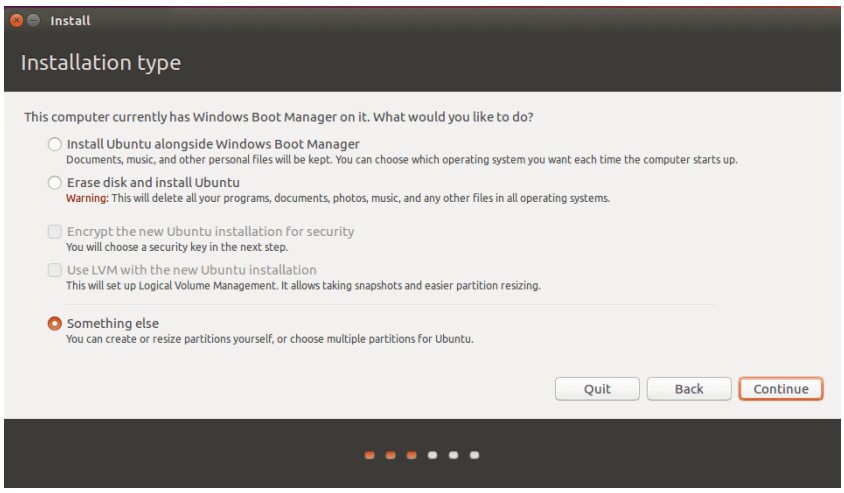
2.1.2 Linux (Ubuntu) operacinės sistemos įdiegimas

Ubuntu OS diegimas vykdomas iš savaime pasikraunančios kompaktinės plokštelės arba prijungiamos USB laikmenos, atminties kortelės ir t.t. Prieš diegimo procesą užkraunama iš esmės visiškai funkcionuojanti operacinė sistema. Vėliau, įsitikinus, kad viskas veikia, diegimas paleidžiamas tiesiog darbo lauke pasirinkus diegimo piktogramą. Nuo *Dapper Drake* (6.06) versijos visas diegimo procesas vyksta visiškai funkcionuojančioje operacinės sistemos grafinėje aplinkoje 6 pav.



6 pav. OS grafinė aplinka - operacinės sistemos

Ubuntu grafinės aplinkos lange galima pasirinkti ar instaliuoti OS atskirai nuo esamos operacinės sistemos, pvz. Microsoft Windows ar suformatuoti kietąjį diską, ištrinant senus failus ir dokumentus ir padaryti *Ubuntu* pagrindine kompiuterio operacine sistema, 7 pav.



7 pav. Ubuntu *OS* įrašymo parinktys

Operacinės sistemos įrašymui buvo pasirinkta laisva kietojo disko dalis, kurioje atskirai nuo esamos *Windows* OS įrašyta *Ubuntu* OS.

2.2 Kaldi programinis paketas

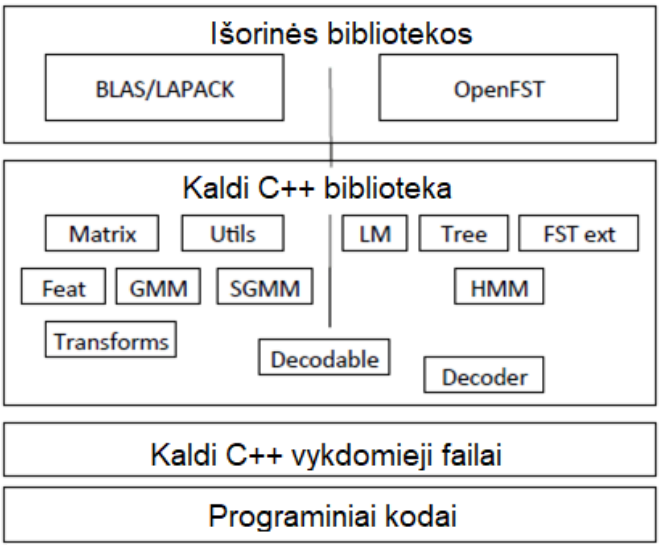
Kaldi gyvavimo pradžia įvyko 2009 m. Johno Hopkinso universiteto seminare, kurio pavadinimas buvo „Mažos vystymosi išlaidos, Aukštos kokybės kalbų atpažinimas naujoms kalboms ir Domenams“ [12]. Pagrindinis šio projekto dėmesys buvo sutelktas į angl. *Subspace Gaussian Mixture Model* (SGMM) modeliavimą ir kai kuriuos leksikos mokymosi tyrimus. Ten buvo pradėta kurti programinė įranga, kuri dabar vadinama *Kaldi*, tačiau tuo metu sukurtas paketas vis dar priklausė nuo HTK. 2010 m. vasarą buvo parašyta didelė dalis programinio kodo, tačiau vis dar nebuvo pilnai veikiančios darbinės sistemos. Kodas buvo išleistas 2011 m. gegužės 14 d. ir pristatytas visuomenei 2011 m. Prahoje.

Kaldi yra atvirojo kodo kalbos atpažinimo įrankių rinkinys, parašytas C ++ kalba, skirtas kalbos atpažinimui ir signalų apdorojimui, laisvai prieinamas naudojant *Apache License v2.0.*[13] *Kaldi* tikslas yra turėti modernų ir lankstų kodą, kurį būtų lengva suprasti, modifikuoti ir išplėsti.

Kaldi specifiniai reikalavimai: baigtinių būsenų keitiklio (FST) pagrįsta sistema, platus linijinės algebros palaikymas ir neribota licencija paskatino programinio paketo plėtrą. Svarbios šio programinio paketo savybės:

- Integracija su baigtinių būsenų keitikliais (angl. Finite State Transducers),
- Platus linijinės algebros palaikymas,
- Išplečiamas dizainas,
- Atvira licencija,
- Kruopštus testavimas.

Kaldi architektūrą sudaro bibliotekų ir mokymo kodai. Šie kodai atlieka bibliotekos funkcijas per komandinės eilutės programas. *Kaldi* C++ biblioteka remiasi *OpenFST* biblioteka ir naudoja optimizuotas linijinės algebros bibliotekas, tokias kaip *BLAS* ir *LAPACK*. Susijusios funkcijos paprastai sugrupuojamos į vieną vardų sritį C++ kode, kuris atitinka vieną katalogą failų sistemoje. Katalogų pavyzdžiai pateikiami 8 paveiksle.



8 pav. *Kaldi* programinio paketo architektūra

- Požymių išgavimas: požymių išgavimu ir signalo skaitymo kodu siekiama sukurti standartines MFCC ir PLP funkcijas.
- Akustinis modeliavimas: palaiko įprastus modelius (Gauso mišinio modelius (GMM)), tarpinius Gauso mišinio modelius (SGMM) ir taip pat galimi naujų rūšių modeliai.
- Kalbos modeliavimas: naudojama sistema, paremta FST pagrindu.
- Dekoderiai: sistema turi keletą dekoderių, nuo paprastų iki gerai optimizuotų.

Kaldi naudoja vykdomuosius failus, kurie įkelia duomenis iš failų ir paprastai rezultatus vėl saugo failuose. Kaip alternatyva, vienos *Kaldi* programos išvestis gali būti įtraukta į kitą komandą naudojant sistemos „vamzdžius“, kurie padeda sujungti atskirus įrankius į vieną grandinę. Be to, šis programinis paketas suteikia naudingų standartizuotų programinių kodų ruošinių, kurie apima *Kaldi* vykdomuosius failus arba pridėda naujų funkcijų. Kodų ruošiniai yra *utils* ir *steps* kataloguose ir pastarieji naudojami daugelyje apmokymo kodų.

2.2.1 Papildomų bibliotekų, skirtų Kaldi programiniam paketui, įdiegimas

Sėkmingai įdiegus *Ubuntu* operacinę sistemą, pirmiausia parsisiunčiamos ir įrašomos papildomos bibliotekos, skirtos *Kaldi* paketo visapusiškam veikimui. Neįdiegus šių plėtinių, rodomos kompiliavimo klaidos, dėl kurių programinis paketas negali funkcionuoti.
Toliau pateikiamas papildomų plėtinių sąrašas [14]:

- *awk* - programavimo kalba, naudojama failams ir duomenų srautams ieškoti ir apdoroti,
- *bash* - *Unix shell* ir kodų programavimo kalba,
- *make* - automatiškai sukuria vykdomas programas ir bibliotekas iš pirminio kodo,
- *perl* - dinaminė programavimo kalba, puikiai tinkanti tekstiniams failams apdoroti,
- *atlas* - skaičiavimų automatizavimas ir optimizavimas tiesinės algebros srityje,
- *autoconf* - automatinė programinės įrangos kompiliacija skirtingoms operacinėms sistemoms,
- *automake* - nešiojamų *Makefile* failų kūrimas,
- *git* - paskirstyta versijų kontrolės sistema,
- *libtool* - statinių ir dinaminių bibliotekų kūrimas,
- *svn* - peržiūros kontrolės sistema, reikalinga *Kaldi* atsisiuntimui ir įdiegimui,

- *wget* - duomenų perdavimas naudojant *HTTP, HTTPS ir FTP* protokolus,
- *zlib* – duomenų suspaudimas.

Papildomų plėtinių įdiegimas atliekamas per *Ubuntu* programinių eilučių terminalą. Tam skirta komandinė eilutė – *sudo apt-get* arba *sudo apt-get install.* Norint įdiegti *atlas* plėtinį, komandinė eilutė atrodytų taip: *sudo apt-get install atlas.*

2.2.2 Kaldi programinio paketo įdiegimas[15]

Kaldi programinis paketas atsisiunčiamas iš GitHub.com įvedus *git* komandą *Ubuntu* operacinės sistemos terminale:

```
git clone https://github.com/kaldi-asr/kaldi.git kaldi --origin upstream
cd kaldi
```

9 pav. Terminalo komanda, naudojama Kaldi programinio paketo atsisiuntimui.

Jei norime pamatyti *Kaldi* paketo atnaujinimus, klaidų pataisymus, vedame komandą:

```
git pull
```

Toliau terminale pasirenkamas aplankas /src ir įvykdomas programos kompiliavimas, 10 pav. pateiktos komandinės eilutės:

```
./configure --shared
make depend -j 8
make -j 8
```

10 pav. *Kaldi* programinio paketo kompiliavimo kodas

Kaldi paketas paruoštas darbui, toliau galime apžvelgti, kas sudaro programinio paketo direktoriją. Pagrindiniame kataloge yra šie aplankai:

- *egs* – programinių kodų pavyzdžiai, leidžiantys greitai sukurti ASR sistemas,
- *misc* - papildomi įrankiai, nereikalingi *Kaldi* funkcionavimui,
- *src* - *Kaldi* programinis kodas,
- *tools*- naudingi komponentai ir išoriniai įrankiai,
- *windows*- įrankiai, skirti *Kaldi* paleisti per Windows platformą.

Akivaizdu, kad svarbiausias katalogas yra *egs*, kuriame ir plėtojame savo ASR sistemą.

2.3 TensorFlow programinis paketas

TensorFlow [16] yra atvirojo kodo platforma, skirta kompiuteriniam mokymuisi. Jis turi išsamią, lanksčią įrankių, bibliotekų ir bendruomenės išteklių ekosistemą, leidžiančią tyrėjams įsigilinti į mašininį mokymasi, o kūrėjams - lengvai kurti ir diegti palaikomas programas.

TensorFlow sukūrė tyrėjai ir inžinieriai, dirbantys „Google Brain“ komandoje „Google“ mašinų intelekto tyrimų organizacijoje, kad galėtų atlikti mašininio mokymosi ir giliųjų neuroninių tinklų tyrimus. Sistema yra pakankamai bendra, kad ją būtų galima pritaikyti ir daugelyje kitų sričių.

TensorFlow teikia stabilias „Python“ ir „C ++“ programų sąsajas, taip pat negarantuojamą atgalinį suderinamumą su kitomis kalbomis.

TensorFlow yra antros kartos „Google Brain“ sistema. 1.0.0 versija buvo išleista 2017 m. Vasario 11 d. Nors nuorodinis diegimas vykdomas atskirais įrenginiais, *TensorFlow* gali vienu metu veikti su keliais procesoriais ir vaizdo plokštėmis (su pasirenkamais CUDA ir SYCL plėtiniais, skirtais bendrosios paskirties skaičiavimui, grafikos apdirbimo įrenginiuose). Šį programinį paketą galima naudoti 64 bitų *Linux, MacOS, Windows* ir mobiliųjų kompiuterių platformose, įskaitant *Android* ir *iOS*.

2.3.1 TensorFlow programinio paketo įdiegimas

Yra keletas metodų, kuriuos galima naudoti *TensorFlow* programinės įrangos įdiegimui, pavyzdžiui, naudojant *pip* komandą [17]. *TensorFlow* diegimas naudojant *conda* programinius paketus suteikia daugybę pranašumų, įskaitant visą paketų valdymo sistemą, platesnį platformos palaikymą, supaprastintą GPU naudojimo patirtį ir geresnį procesoriaus našumą. Šiuos paketus galima rasti *Anaconda* repozitorijoje, o juos įdiegti paprasta, tiesiog įrašius *conda install tensorflow* arba *conda install tensorflow-gpu* komandinės eilutės sąsajoje.

Vienas pagrindinių *TensorFlow* diegimo naudojant *conda*, o ne *pip* komandą pranašumų yra *conda* paketų valdymo sistema. Įdiegus *TensorFlow* naudojant *conda*, įdiegiamos visos būtinos bei suderinamos programinių paketų priklausomybės. Viskas atliekama automatiškai ir vartotojams nereikia įdiegti jokios papildomos programinės įrangos, naudojant sistemos paketų tvarkytuves ar kitas priemones. Be to, bet kurį iš 1400 profesionaliai sukurtų paketų, esančių *Anaconda* saugykloje galima įdiegti kartu su *TensorFlow*, norint išvystyti visavertę mokslo duomenų aplinką. Šie paketai yra įdiegiami izoliuotoje *conda* aplinkoje, kurios turinys nedaro įtakos kitoms aplinkoms.

Po sėkmingo *Anaconda* programinės aplinkos įdiegimo, kuris aprašomas 2.4 skyrelyje, *TensorFlow* įrašomas sukuriant naują *conda* aplinką ir programinių eilučių terminale įvedus šias komandas:

```
conda create -n tensorflow_env tensorflow
 conda activate tensorflow_env
```

11 pav. *TensorFlow* programinio paketo aktyvavimo komandos

Vaizdo plokštės (GPU) versija aktyvuojama šiomis komandomis:

```
conda create -n tensorflow_gpuenv tensorflow-gpu
 conda activate tensorflow_gpuenvF
```

12 pav. *TensorFlow* programinio paketo GPU versijos aktyvavimo komandos

2.4 Spyder (Anaconda 3) programinė aplinka

Spyder yra mokslinė aplinka, parašyta Python programavimo kalba, sukurta mokslininkų, inžinierių ir duomenų analitikų [18]. *Spyder* gali pasigirti unikalia redagavimo, analizės, derinimo ir profiliavimo funkcijų kombinacija, bei duomenų tyrimu, interaktyviu vykdymu ir vaizdžiomis mokslinio paketo vizualizavimo galimybėmis.

Be daugybės integruotų funkcijų, galimybes dar galima išplėsti naudojant papildinių sistemą ir API (angl. Application Programming Interface). Be to, *Spyder* taip pat gali būti naudojama kaip „PyQt5“ plėtinių biblioteka, leidžianti kūrėjams remtis jos funkcionalumu ir įterpti jos komponentus, pavyzdžiui, interaktyviąją konsolę, į savo „PyQt“ programinę įrangą.

Spyder programinė aplinka įdiegiama per *Ubuntu* programinių eilučių terminalą, įvedus komandą:

```
sudo apt-get install spyder3
```

13 pav. *Spyder* programinės aplinkos įdiegimo komanda

2.5 Vardų garsyno aprašymas

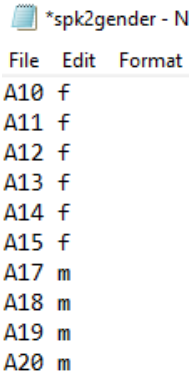
Šiam baigiamąjam darbui pasirinktas vardų garsynas. Lietuviškų vardų garsyno įrašų rinkinį *VARDAI_18_3* sudaro dvidešimt vieno diktoriaus balso įrašai: 18 – apmokymui ir 3 – testavimui.

Kiekvieno garso failo pavadinimas yra sudarytas iš diktoriaus numerio, tariamo vardo pirmos raidės, bei dviejų skaitmenų iteracijos – pvz. 0A01, 0B10 ir t.t. Kiekvienas vardas ištariamas 20 kartų. 21 diktoriaus, lietuviškų vardų pavadinimų garso įrašų rinkinys sudarytas iš 10920 skirtingų balso įrašų.

2.6 Vardų garsyno paruošimas

Kaldi programiniam paketui sudaromi 5 tekstiniai failai [19], kurie aprašo garsyno diktorius, bei transkripcijas atitinkamiems garso įrašams.

2.6.1 Failas spk2gender, diktoriaus lyties nustatymas. Failą *spk2gender* sudaro *diktoriaus identifikatorius* ir *lyties indikatorius*. 14 pav. pateikiamas informacinis failas aprašantis diktoriaus lytį:



14 pav. *Spk2gender* failo fragmentas.

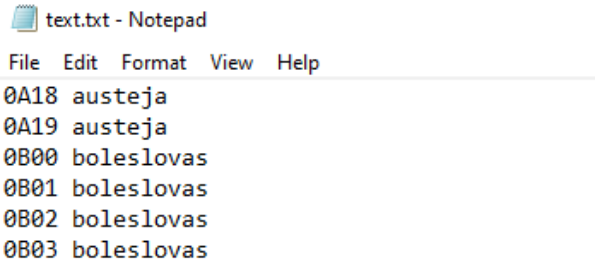
Diktoriaus identifikatorius sudarytas iš diktoriaus numerio, pvz. A10, A15. Lyties indikatorius sudarytas iš (m – vyras, f - moteris).

2.6.2 Failas utt2spk, garso įrašų sąsajos su diktoriais. Failą *utt2spk* sudaro *garso įrašo identifikatorius* ir *diktoriaus identifikatorius*. 15 pav. pateikiamas informacinis failas aprašantis garso įrašų sąsajas su diktoriais:

15 pav. *Utt2spk* failo fragmentas.

Garso įrašų identifikatorius sudarytas iš diktoriaus identifikatorius, tariamo vardo pirmosios raidės (pvz. A – Austeja, B – Boleslovas), garso įrašo numeris nuo 00 iki 19.

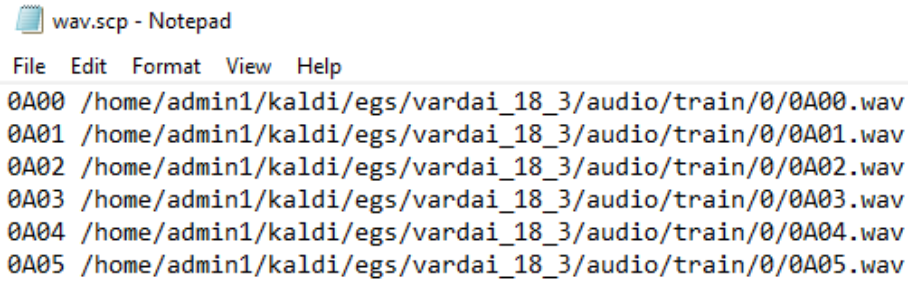
2.6.3 Failas text, garso įrašų sąsajos su tekstine transkripcija. Failą *text* sudaro *garso įrašo identifikatorius* ir *įrašo transkripcija*. 16 pav. pateikiamas informacinis failas aprašantis garso įrašų sąsajas su tekstine informacija:



16 pav. *Text* failo fragmentas.

Garso įrašų sąsaja su tekstine informacija aprašoma taip: pirmas skaičius, pvz. - 0 yra diktoriaus numeris, toliau vardo pirmoji raidė, pvz. A – austeja, B - boleslovas ir įrašo numeris – nuo 00 iki 19, bei visas ištariamas vardas – austeja.

2.6.4 Failas wav.scp, garso įrašų sąsajos su realiu garso įrašu. Failą *wav.scp* sudaro *garso įrašo identifikatorius* ir *garsinio failo pavadinimas su katalogo vardu*. 17 pav. pateikiamas informacinis failas aprašantis garso įrašų sąsajas su realiu garso įrašu:



17 pav. *Wav.scp* realaus garso failų sąrašo fragmentas.

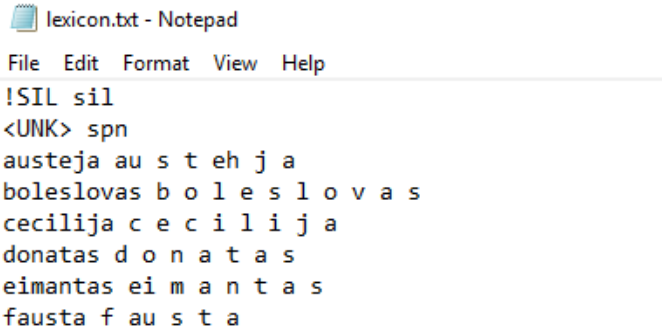
2.6.5 Failas corpus.txt, aprašantis visas galimas transkripcijas. Šiame faile aprašomas visų apmokymui ir testavimui paruoštų garso įrašų transkripcijų sąrašas, 5720 tekstinių eilučių: 220 eilučių su įrašu „*austeja*“, 220 eilučių su įrašu „*boleslovas*“ ir t.t.

Failai *spk2gender, utt2spk, text, wav.scp* aprašomi testavimo ir apmokymo duomenims atskirai. Aprašomieji failai kuriami pagal tai, kokie diktoriai priklauso testavimo ir apmokymo dalims. Testavimo garsyną aprašantys failai laikomi projekto *vardai_18_3* kataloge *data* /test, apmokymų garsyną aprašantys failai laikomi *data* /*train* kataloge.

2.7 Vardų garsyno fonemų aprašomieji failai

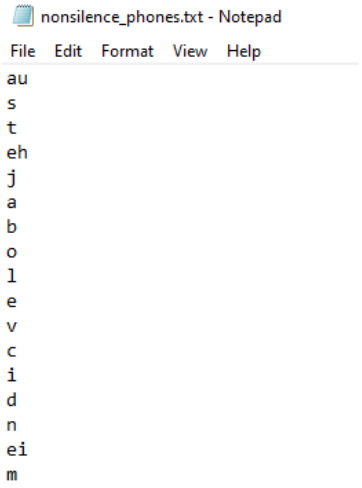
Pateikiami vardų garsyno fonemas aprašantys failai: *lexicon.txt, nonsilence_phones.txt, silence_phones.txt, optional_silence.txt.*

Lexicon.txt. Vardų garsyno tyrimams atlikti panaudotos grafemos, kurios yra mažiausi rašto kalbos vienetai, atitinkantys garsinės kalbos vienetus – fonemas, tačiau dvibalsiai *ie* ir *uo* sudaro atskiras grafemas. 18 pav. pateikiamas fonetines transkripcijas aprašantis failas.



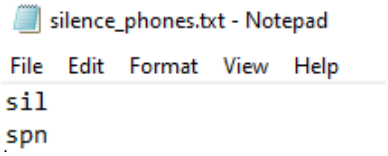
18 pav. *Lexicon.txt* failo fragmentas.

Nonsilence_phone.txt. Vardų fonemų sąrašo, kuriame nėra tylos fonemos failas, pateikiamas 19 pav.



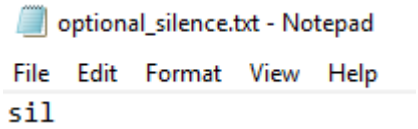
19 pav. *Nonsilence_phones.txt.* Fonemų (grafemų) sąrašo fragmentas.

Silence_phones.txt. Tylos fonemų sąrašo failas, pateikiamas 20 pav.



20 pav. *Silence_phone.txt.* failas.

Optional_silence.txt. Papildomų tylos fonemų sąrašo failas, pateikiamas 21 pav.



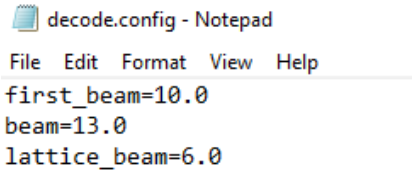
21 pav. *Optional_silence.txt.* Papildomų tylos fonemų failas.

Visi pastarieji failai, aprašantys fonemas, laikomi projekto *vardai_18_3* aplanke data/local/dict.

2.8 Konfigūraciniai failai

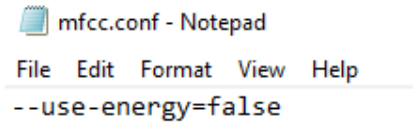
Sukuriami du papildomi failai, *decode.config* ir *mfcc.conf* kurie atlieka konfiguracijos pokyčius dekodavimo procese, bei požymių išgavimo procese. Šie failai laikomi: *kaldi/egs/vardai_18_3/conf* kataloge:

***Decode.config*.** Dekodavimo proceso nustatymai, pateikiami 22 pav.



22 pav. *Decode.config* dekodavimo nustatymų failas

***Mfcc.conf*.** *MFCC* metodo požymių išgavimo proceso nustatymai, pateikiami 23 pav.

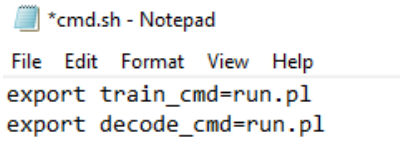


23 pav. *Mfcc.conf* proceso nustatymų failas

2.9 Užduoties failai

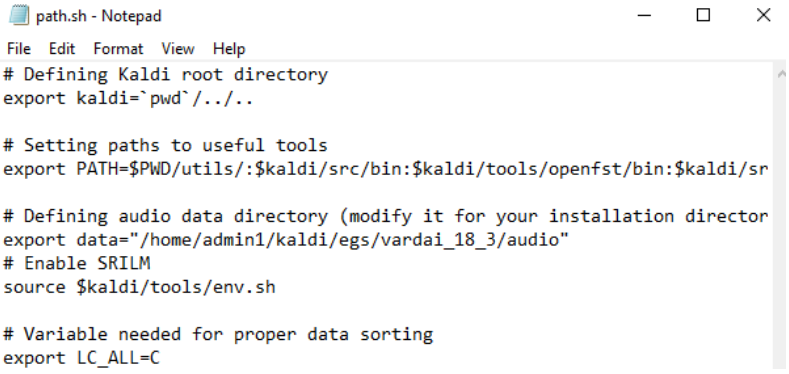
Galutiniam sistemos paleidimui reikalingi dar trys papildomi failai: *cmd.sh, path.sh, run.sh.* Šie failai talpinami *kaldi/egs/vardai_18_3* projekto kataloge.

Cmd.sh. Failas leidžia pasirinkti ar sistema veiks vietiniame kompiuteryje ar išoriniame serveryje. 24 pav. pateikiamas *cmd.sh* failas, kurio nustatymai, pagal nutylėjimą skirti vietinio kompiuterio resursams naudoti.



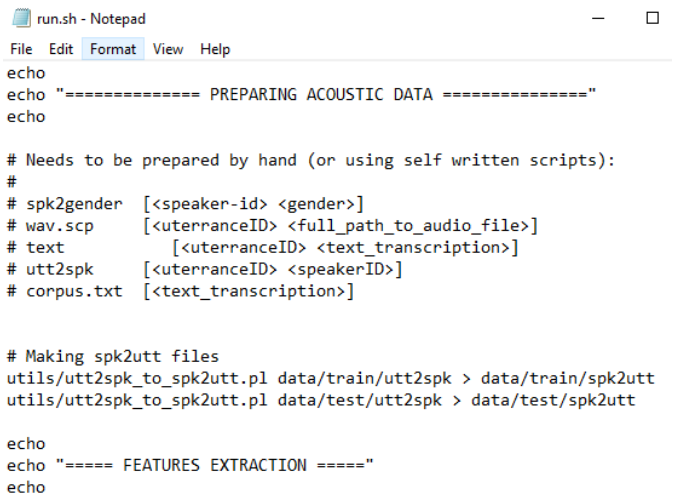
24 pav. *Cmd.sh* failo nustatymai

Path.sh. Failas, aprašantis nuorodas programiniams plėtiniams, nurodantis Kaldi paketo pagrindinę direktoriją, garsyno įrašų direktoriją ir t.t. 25 pav. pateikiamas *path.sh* failas.



25 pav. *Path.sh* failo nustatymai

Run.sh. Tai yra pagrindinis programos paleidimo failas, kuriame aprašomos nuorodos į kitus *shell* (.sh) tipo failus. Taip pat atliekamas akustinių duomenų paruošimas, požymių išgavimas, kalbos duomenų paruošimas, kalbos modelio sukūrimas, apmokymas, rikiavimas, dekodavimas, bei aprašomi skirtingi tyrimo metodai, pvz. monofoninis metodas, trifoninis metodas, SGMM metodas, LDA, MMLT, SAT ir DNN metodai. 26 pav. pateikiamas *run.sh* failo fragmentas.



26 pav. *Run.sh* pagrindinio failo fragmentas

2.10 Garso įrašų užtriukšminimas naudojant „SoX“ įrankį

SoX yra kelių platformų („Windows“, „Linux“, „MacOS X“ ir kt.) komandų eilutės įrankis, galintis konvertuoti įvairius kompiuterio garso failų formatus į kitus formatus[20]. *SoX* skaito ir rašo garso failus populiariausiais formatais ir pasirinktinai gali jiems pritaikyti efektus. Jis gali sujungti kelis įvesties šaltinius, sintetinti garsą ir daugelyje sistemų veikti kaip bendrosios paskirties garso grotuvas arba kelių takelių garso įrašymo įrenginys. Jis taip pat turi ribotas galimybes padalinti įvestį į kelis išvesties failus.

Pateikiama *Sox* komandinė eilutė, skirta garso įrašų užtriukšminimui, ,,baltu triukšmu” 5dB lygyje, pavaizduota 27 paveiksle.

```
C:\Users\GedasPC\Desktop\Magistr>sox.exe E:\Noise\audio\train\0\0A00.wav noise.wav synth whitenoise vol 0.1   && sox -m
E:\Noise\audio\train\0\0A00.wav noise.wav E:\Noise\SNR5\train\0\0A00.wav
```

27 pav. ,,Sox” komandinė eilutė, skirta garso įrašų užtriukšminimui

2.11 TensorFlow programinio paketo konfiguraciniai failai

Vardų garsyno tyrimui atlikti, reikia sudaryti 6 programinius failus: *generate_test_vrd.py, generate_train_eval_new.py, deep_speech.py, deep_speech_model.py, decoder.py, get_predictions.py.* Šie failai skirti garsyno duomenų paruošimui, kalbos modelio sudarymui, apmokymui, dekodavimui ir rezultatų gavimui.

Generate_test_vrd.py. Šis failas skirtas vardų garsyno testavimo duomenų paruošimui. Pirmiausia nustatoma direktorija iš kurios imami garso įrašų failai, o toliau atliekamas garso įrašų atpažinimas pagal pirmąją vardo raidę ir atitinkamai priskiriamas vardas, garso įrašo pavadinimas, jo dydis, bei transkripcija, *wav_filename, wav_filesize, transcript*. 28 pav. pateikiamas *generate_test_vrd.py* failas.

```
if __name__ == "__main__":
  print('austeja ')
  dataset = []
  pattern="*.wav"
  for (path,sub_dirs,files) in os.walk('vardai_18_3', topdown=True):
#    print(path)
#    print(sub_dirs)
#    print(files)
    for name in files:
            if fnmatch.fnmatch(name, pattern):
                file = os.path.join(path, name)
                print(name)
                bytes = os.path.getsize(file)
                transcript = str(re.search(r'\D', name).group())
                transcript = transcript.replace("A","Austeja")
                transcript = transcript.replace("B","Boleslovas")
                transcript = transcript.replace("C","Cecilija")

                dataset.append((file, bytes, transcript))
# Test set.
data_frame = pandas.DataFrame(data=dataset,columns=["wav_filename", "wav_filesize", "transcript"])
```

28 pav. G*enerate_test_vrd.py* failo fragmentas.

Paleidus failą, sugeneruojamas duomenų sąrašas atskirame *test_set.csv* faile. 29 pav. pateikiamas šio failo fragmentas.

	A	B	C	D	E
1	wav_filename,wav_filesize,transcript				
2	vardai_18_3\audio\test\16\16A00.wav,38744,Austeja				
3	vardai_18_3\audio\test\16\16A01.wav,39944,Austeja				
4	vardai_18_3\audio\test\16\16A02.wav,41144,Austeja				
5	vardai_18_3\audio\test\16\16A03.wav,48344,Austeja				
6	vardai_18_3\audio\test\16\16A04.wav,39944,Austeja				
7	vardai_18_3\audio\test\16\16A05.wav,39944,Austeja				
8	vardai_18_3\audio\test\16\16A06.wav,41144,Austeja				
9	vardai_18_3\audio\test\16\16A07.wav,41144,Austeja				
10	vardai_18_3\audio\test\16\16A08.wav,41144,Austeja				
11	vardai_18_3\audio\test\16\16A09.wav,43544,Austeja				
12	vardai_18_3\audio\test\16\16A10.wav,42344,Austeja				
13	vardai_18_3\audio\test\16\16A11.wav,39944,Austeja				
14	vardai_18_3\audio\test\16\16A12.wav,43544,Austeja				
15	vardai_18_3\audio\test\16\16A13.wav,44744,Austeja				
16	vardai_18_3\audio\test\16\16A14.wav,44744,Austeja				
17	vardai_18_3\audio\test\16\16A15.wav,45944,Austeja				
18	vardai_18_3\audio\test\16\16A16.wav,43544,Austeja				
19	vardai_18_3\audio\test\16\16A17.wav,43544,Austeja				
20	vardai_18_3\audio\test\16\16A18.wav,45944,Austeja				
21	vardai_18_3\audio\test\16\16A19.wav,44744,Austeja				
22	vardai_18_3\audio\test\16\16B00.wav,44744,Boleslovas				

29 pav. *Test_set.csv* failo fragmentas.

Generate_train_eval_new.py Šis failas skirtas garsyno apmokymo duomenų nustatymui. Galima keisti procentinę duomenų dalį patenkančią į apmokymą. 30 pav. pateikiamas šio failo fragmentas.

```
22 if __name__ == "__main__":
23   print('austeja')
24   dataset = []
25   randomize=True
26   train_set_percent=0.7
27   pattern="*.wav"
28   for (path,sub_dirs,files) in os.walk('SNR5', topdown=True):
29     for name in files:
30             if fnmatch.fnmatch(name, pattern):
31                 file = os.path.join(path, name)
32 #                print(file)
33                 bytes = os.path.getsize(file)
34                 transcript = int(re.search(r'\d', name).group())
35                 transcript = numbers[transcript]
36 #                print(transcript)
37                 dataset.append((file, bytes, transcript))
```

30 pav. G*enerate_train_eval_new.py* failo fragmentas

Deep_speech.py. Tai yra pagrindinis projekto direktorijos failas, kuriuo prieinama prie DeepSpeech modelio apmokymo ir įvertinimo. 31 pav. pateikiamas *deep_speech.py* failo fragmentas.

```
def evaluate_model(estimator, speech_labels, entries, input_fn_eval):

    # Get predictions
    predictions = estimator.predict(input_fn=input_fn_eval)
```

31 pav. *Deep_speech.py* failo fragmentas.

Deep_speech_model.py. DeepSpeech modelio struktūrinis failas, kuriame aprašomi galimi neuroninių tinklų tipai, paslėpti sluoksniai ir t.t. 32 pav. pateikiamas *deep_speech_model.py* failo fragmentas.

```
23 # Supported rnn cells.
24 SUPPORTED_RNNS = {
25     "lstm": tf.nn.rnn_cell.BasicLSTMCell,
26     "rnn": tf.nn.rnn_cell.RNNCell,
27     "gru": tf.nn.rnn_cell.GRUCell,
```

32 pav. D*eep_speech_model.py* failo fragmentas.

Decoder.py. Dekodavimo proceso failas, kuriame apibūdinamas indeksų konvertavimas į žodžius, klaidingų žodžių rodiklio (WER) apskaičiavimas, geriausio žodžių atpažinimo spėjimas ir t.t. 33 pav. pateikiamas *decoder.py* failo fragmentas.

```
42   def convert_to_string(self, sequence):
43       """Convert a sequence of indexes into corresponding stri
44       return ''.join([self.int_to_char[i] for i in sequence])
45
46   def wer(self, decode, target):
47       """Computes the Word Error Rate (WER).
```

33 pav. D*ecoder.py* failo fragmentas.

Get_predictions.py. Modelio testavimo ir rezultatų išvedimo failas. Įvykdžius šį programinį failą, apibūdinamas modelio efektyvumas, atsižvelgiant į klaidingų žodžių rodiklį, kurio reikšmė atvaizduojama programinio terminalo lange.

34 pav. pateikiamas *get_predictions.py* failo fragmentas.

```
def test_model(estimator, speech_labels, entries, input_fn_eval):
    """Test the model performance using WER anc CER as metrics.
```

34 pav. *Get_predictions.py* failo fragmentas.

3. Tyrimų rezultatai

Tyrimams atlikti buvo panaudotas lietuviškų vardų, dvidešimt vieno diktoriaus garsynas. Garsyno atpažinimui panaudoti du programiniai paketai – *Kaldi* ir *TensorFlow*, o duomenų atpažinimo tikslumui palyginti buvo naudojami skirtingi akustinio modeliavimo metodai. Tolimesni rezultatai bus pateikiami kiekvienam programiniui paketui atskirai.

3.1 Vardų garsyno tyrimo su Kaldi paketu, rezultatai.

Pirmiausia atliktas monofoninio metodo tyrimas.

Apmokymui skirti 18 diktorių garso įrašai talpinami *kaldi/egs/vardai_18_3/audio/train* direktorijoje, o testavimui skirti 3 diktorių garso įrašai, esantys *kaldi/egs/vardai_18_3/audio/test* direktorijoje.

Rezultatams gauti linux operacinės sistemos terminale paleidžiamas *run.sh* failas. Gauti tokie rezultatai:

%WER 0.26 [4 / 1560, 4 ins, 0 del, 0 sub]
%SER 0.26 [4 / 1560]
exp/mono/decode/wer_11
%WER 0.26 [4 / 1560, 4 ins, 0 del, 0 sub]
%SER 0.26 [4 / 1560]
exp/mono/decode/wer_12
%WER 0.19 [3 / 1560, 3 ins, 0 del, 0 sub]
%SER 0.19 [3 / 1560]
exp/mono/decode/wer_13
%WER 0.06 [1 / 1560, 1 ins, 0 del, 0 sub]
%SER 0.06 [1 / 1560]
exp/mono/decode/wer_14
%WER 0.06 [1 / 1560, 1 ins, 0 del, 0 sub]
%SER 0.06 [1 / 1560]
exp/mono/decode/wer_15
%WER 0.06 [1 / 1560, 1 ins, 0 del, 0 sub]
%SER 0.06 [1 / 1560]
exp/mono/decode/wer_16
%WER 0.06 [1 / 1560, 1 ins, 0 del, 0 sub]
%SER 0.06 [1 / 1560]
exp/mono/decode/wer_17
%WER 0.06 [1 / 1560, 1 ins, 0 del, 0 sub]
%SER 0.06 [1 / 1560]
exp/mono/decode/wer_7
%WER 0.38 [6 / 1560, 6 ins, 0 del, 0 sub]
%SER 0.38 [6 / 1560]
exp/mono/decode/wer_8
%WER 0.32 [5 / 1560, 5 ins, 0 del, 0 sub]
%SER 0.32 [5 / 1560]

Testavimui atlikta 10 atskirų kalbos atpažinimo bandymų, išvedant atpažinimo paklaidą (WER) procentais ir padarytą klaidingų atpažinimų skaičių iš visų ištarimų skaičiaus. Atpažinimo klaidos suskirstomos pagal klaidos pobūdį:

1. Žodžio įterpimas (angl. insertion) – kai vietoj tikslaus žodžio ištarimo, įterpiamas visiškai kitas žodis.
2. Žodžio sumaišymas (angl. substitution) – kai vietoj ištariamo žodžio parenkamas kitas žodžio ištarimas iš garsyno.
3. Žodžio ištrinimas (angl. deletion)– kai automatinis kalbos atpažintuvas neatpažysta ištarto žodžio.

Šiuo atveju, monofoniniu tyrimo metodu gauta didžiausia atpažinimo paklaida yra – **0,38 %**, o mažiausia – **0,06 %**.

Toliau atliktas tyrimas su 5 skirtingais akustinio modeliavimo metodais. Apmokymui skirti 18 diktorių garso įrašai, o testavimui skirti 3 diktorių garso įrašai. 2 lentelėje palyginami rezultatai tarp skirtingų modeliavimo metodų (monofoninis, trifoninis, LDA, SAT, SGMM):

2 lentelė. Skirtingų modeliavimo metodų atpažinimo paklaidos

Bandymo metodas	Paklaida, %
Monofoninis	0,06
Trifoninis	0
LDA	0,06
SAT	0
SGMM	0

Iš 2 lentelės rezultatų matyti, kad *trifoniniu, SAT ir SGMM* metodais atpažinimo paklaidos nėra, tai reiškia, kad visi garsyno įrašai atpažinti teisingai. Paklaida gauta tik su *monofoniniu* ir *LDA* metodu, kuri yra maža – 0,06 %. Galima teigti, kad vardų garsynas atpažįstamas be klaidų, tačiau, realiu atveju garso įrašai ne visada būna “švarūs”, fone yra girdimas triukšmas, pašaliniai garsai. Toliau tyrimas bus atliekamas su užtriukšmintais įrašais, ko pasekoje išryškės atskirų metodų privalumai.

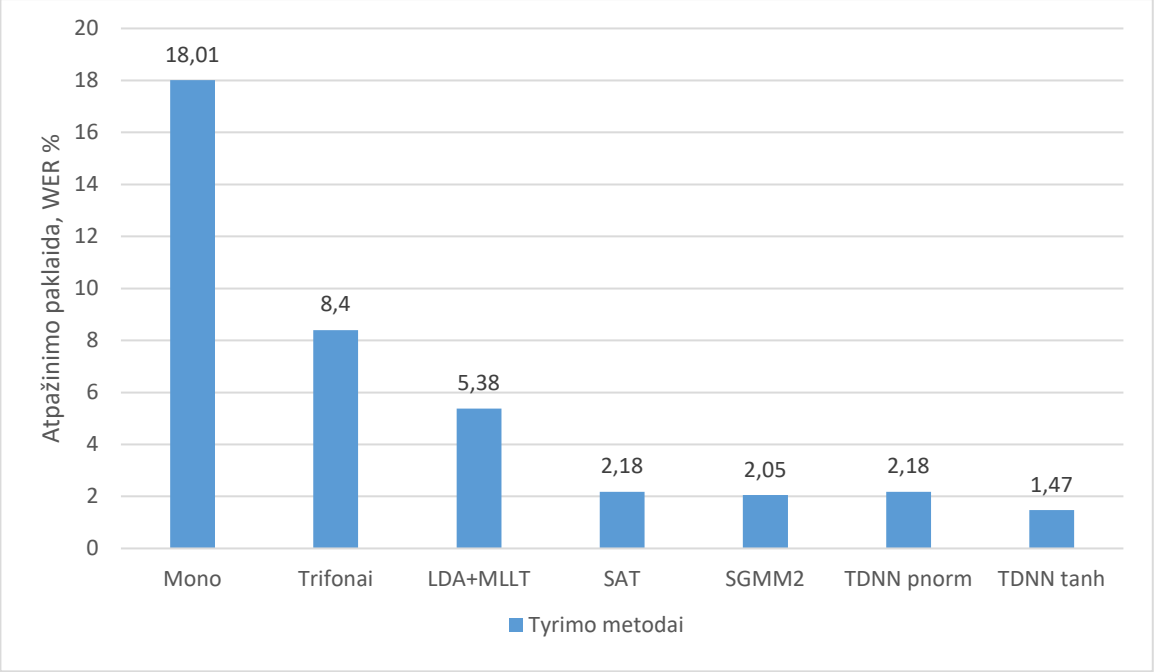
3.2 Vardų garsyno tyrimo su užtriukšmintais įrašais ir Kaldi paketu, rezultatai

Garsyno Vardai_18_3 charakteristikos: 21 diktoriaus po 20 kartų sudiktuoti 22 lietuviški vardai ir 4 kitokie žodžiai. Bendras įrašų skaičius 10920. 18 diktorių įrašai naudojami apmokymui, o likusių 3 diktorių – testavimui.

3 lentelė. Vardų garsyno atpažinimo klaida procentais

Garsynas	Metodas						
	mono	trifonai	LDA+MLLT	SAT	SGMM2	TDNN pnorm	TDNN tanh
Vardai_18_3	18,01	8,40	5,38	2,18	2,05	2,18	**1,47**

Geriausias atpažinimo rezultatas vardų garsynui gautas su TDNN *tanh* modifikacija. 35 pav. pateikiamas gautų rezultatų grafinis vaizdas.



35 pav. Skirtingų tyrimo metodų rezultatai

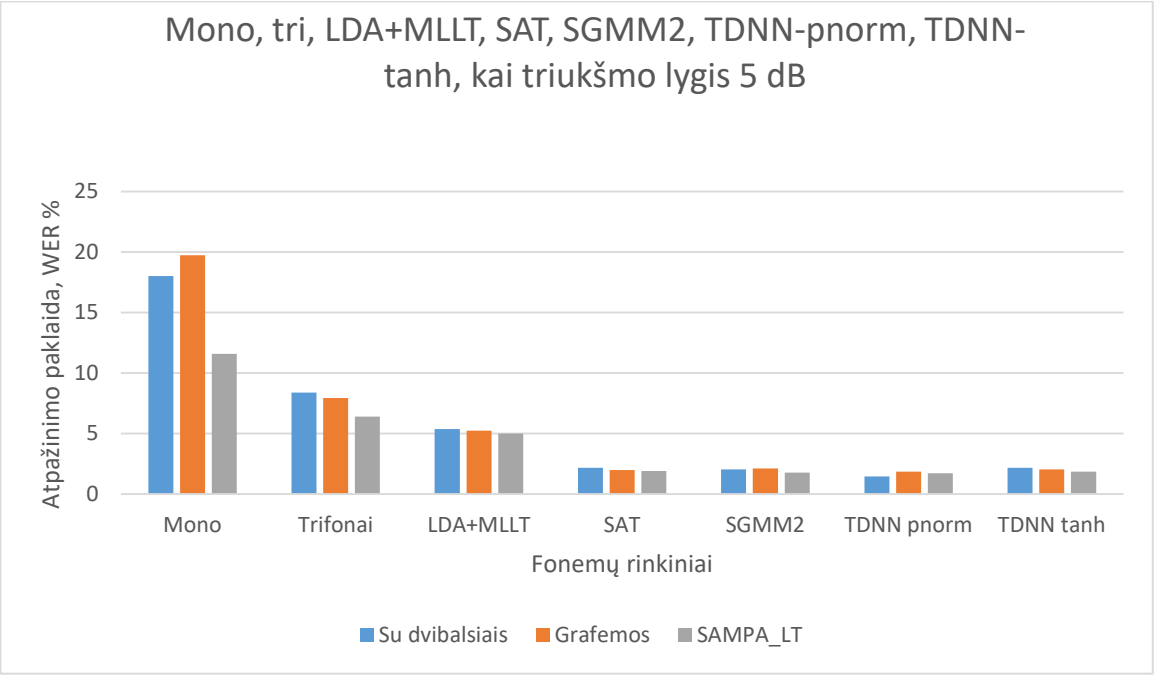
Tyrimai pakartoti su užtriukšmintu 5 dB lygyje garso įrašų rinkiniu Vardai_18_3. Panaudoti trys fonemų rinkiniai su dvibalsiais, grafemomis, SAMPA_LT. Rinkinio Vardai_18_3 rezultatai – 4 lentelėje.

4 lentelė. Garsyno Vardai_18_3 atpažinimo klaida procentais, kai triukšmo lygis 5 dB

Rinkinys	Metodas						
	mono	trifonai	LDA+MLLT	SAT	SGMM2	TDNN pnorm	TDNN tanh
su dvibalsiais	18,01	8,40	5,38	2,18	2,05	**1,47**	2,18
grafemos	19,74	7,95	5,26	1,99	2,12	1,86	2,05
SAMPA_LT	11,60	6,41	5,00	1,92	1,79	1,73	1,86

Tikrinant užtriukšmintų 5 dB lygyje Vardai_18_3 garsyno atpažinimo tikslumą naudojant fonemų su dvibalsiais, grafemų ir SAMPA_LT fonemų rinkinius nustatyta, kad mažiausia garsyno Vardai_18_3 įrašų atpažinimo paklaida gaunama naudojant fonemų su dvibalsiais rinkinį ir TDNN *pnorm* metodą. Didžiausia paklaida gaunama naudojant grafemų rinkinį ir monofoninį metodą.

Pastebėta, kad *monofoniniu* ir *trifoniniu* metodu gaunami žymiai prastesni rezultatai su užtriukšmintais garso įrašais, nei buvo prieš tai, su neužtriukšmintais. Nedaug skiriasi tik *TDNN pnorm*, *TDNN tanh*, *SAT* ir *SGMM2* metodų tyrimo rezultatai. 36 pav. pateikiamas gautų rezultatų grafinis vaizdas.



36 pav. Skirtingų tyrimo metodų su užtriukšmintais garso įrašais, rezultatai

3.3 Kryžminis patikrinimas

Užtriukšminto garso įrašų rinkinio Vardai_18_3-SNR5 atpažinimo tyrimas su KALDI paketu. Tyrimai vykdyti su 21 diktoriaus sudiktuotais 22 lietuviškais vardais ir keliais daiktavardžiais , kiekvieno diktoriaus ištartais po 20 kartų (signalo/triukšmo lygis 5 dB). Bendras įrašų skaičius 10920. 18 diktorių įrašai naudojami apmokymui, o likusių 3 diktorių – testavimui. Atliktas septynis kartus kryžminis atpažinimo klaidos patikrinimas keičiant apmokymo ir testavimo duomenis. Testavimo katalogų duomenys:

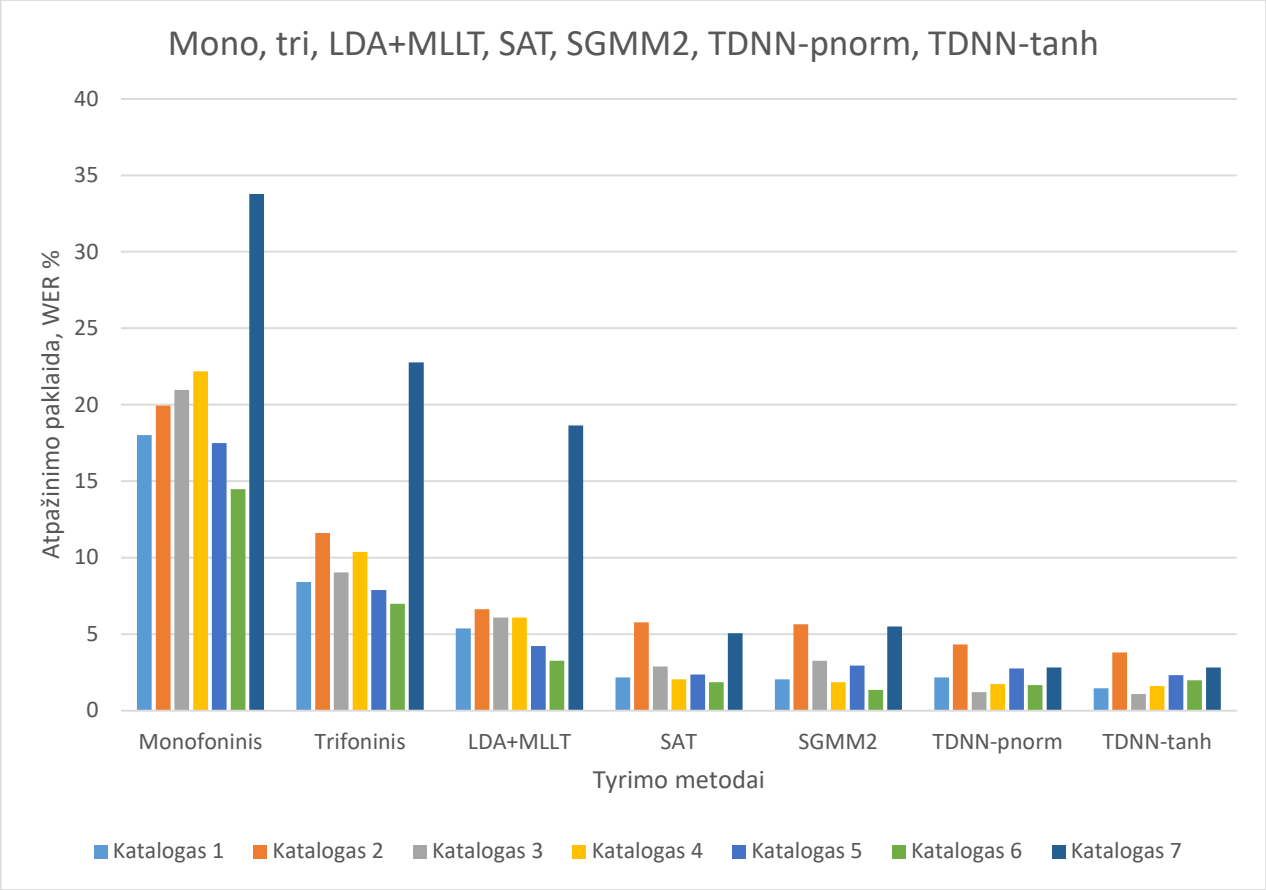
1 katalogas: 8f, 9m ir 16f diktorius (aštuntas diktorius-moteris, devintas diktorius-vyras ir šešioliktas diktorius-moteris).
2 katalogas: 0f, 1f, 2f.
3 katalogas: 3f, 4m, 5m.
4 katalogas: 6m, 7m, 10 f.
5 katalogas: 11f, 12f, 13f.
6 katalogas: 14f, 15f, 17m.
7 katalogas: 18f, 19m, 20f.

Tyrimuose naudotas fonemų su dvibalsiais rinkinys, o *lexicon.txt* failas:
austeja au s t eh j a
boleslovas b o l e s l o v a s
cecilija c e c i l i j a
donatas d o n a t a s
eimantas ei m a n t a s
fausta f au s t a
grazvydas g r a zh v y d a s
hansas h a n s a s
izaokas i z a o k a s
jonas j o n a s
karolis k a r o l i s
laima l ai m a
martynas m a r t y n a s
nojus n o j u s
oskaras o s k a r a s
patrikas p a t r i k a s
kju k j uh
ricardas r i cc a r d a s
sandra s a n d r a
teodoras t e o d o r a s
ulijona u l i j o n a
vacys v a c y s
washington v a sh i n g t o n
iksas i k s a s
ygrekas y g r e k a s
zacharijus z a x a r i j u s
Gauti rezultatai pateikti 5 lentelėje.

5 lentelė. Garso įrašų rinkinio Vardai_18_3-SNR5 kryžminio patikrinimo atpažinimo klaida procentais

Metodas	Katalogas							Vidurkis
	1	2	3	4	5	6	7	
monofoninis	18,01	19,95	20,97	22,18	17,50	14,49	33,78	20,98
trifoninis	8,40	11,61	9,04	10,38	7,88	6,99	22,76	11,01
LDA+MLLT	5,38	6,63	6,09	6,09	4,23	3,27	18,65	7,19
SAT	2,18	5,77	2,89	2,05	2,37	1,86	5,06	3,17
SGMM2	2,05	5,64	3,27	1,86	2,95	1,35	5,51	3,23
TDNN-pnorm	2,18	4,33	1,22	1,73	2,76	1,67	2,82	2,39
TDNN-tanh	1,47	3,81	1,09	1,60	2,31	1,99	2,82	**2,16**

Atlikus kryžminį garso įrašų rinkinio Vardai_18_3-SNR5 su 5 dB užtriukšminimu testavimą, mažiausias paklaidos vidurkis gautas su *TDNN-tanh* metodu, o didžiausias – *monofoniniu* metodu. *Monofoniniu* metodu atpažinimo paklaida kinta nuo 14,49 % iki 33,78 %, o *TDNN-tanh* metodu, tik nuo 1,47 % iki 3,81 % ir tai yra didelė pažanga neuroninių tinklų metodams. 37 pav. pateikiamas gautų rezultatų grafinis vaizdas.



37 pav. Skirtingų tyrimo metodų, kryžminio patikrinimo rezultatai

3.4 Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos priklausomybių nuo metodų parametrų tyrimai.

Pradiniuose tyrimuose buvo naudojamos tokios numatytosios metodų parametrų reikšmės:
Monofoninis metodas:
totgauss= 1000
Trifoninis metodas:
numLeavesTri1=2000
numGaussTri1=11000
LDA metodas:
numLeavesMLLT=2000
numGaussMLLT=11000
SAT metodas:
numLeavesSAT=2000
numGaussSAT=11000
SGMM2 metodas:
numGaussUBM=400
numLeavesSGMM=7000
numGaussSGMM=9000
TDNN *pnorm* metodas:
2 sluoksniai, p=2, *pnorm_input_dim=2000, pnorm_output_dim=200*
TDNN *tanh* metodasa:
2 sluoksniai, *hidden_layer_dim=300*

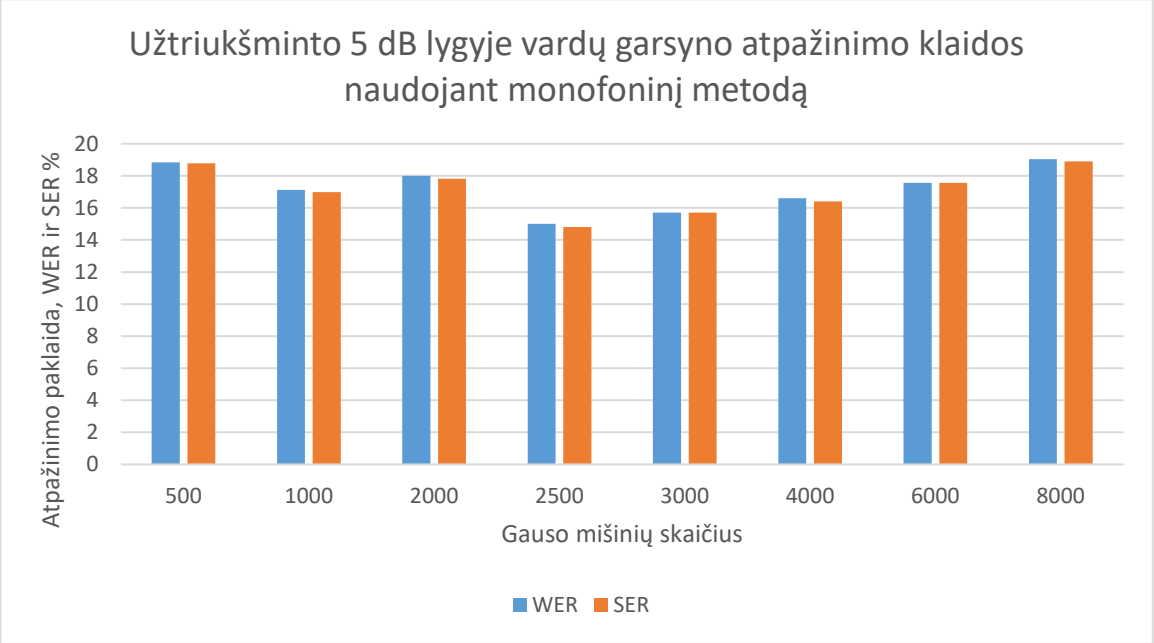
3.4.1 *Monofoninis metodas*

Bandome surasti mažiausią klaidą keisdami parametrą *totgauss*.
Gautos atpažinimo klaidos parodytos 6 lentelėje.

6 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant monofoninį metodą

Klaida	Gauso mišinių skaičius							
	500	1000	2000	2500	3000	4000	6000	8000
WER	18,85	17,12	18,01	15,00	15,71	16,60	17,56	19,04
SER	18,78	16,99	17,82	14,81	15,71	16,41	17,56	18,91

Mažiausia klaida gauta kai *totgauss=2500.* 38 pav. pateikiamas gautų rezultatų grafinis vaizdas.



38 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant monofoninį metodą, rezultatai

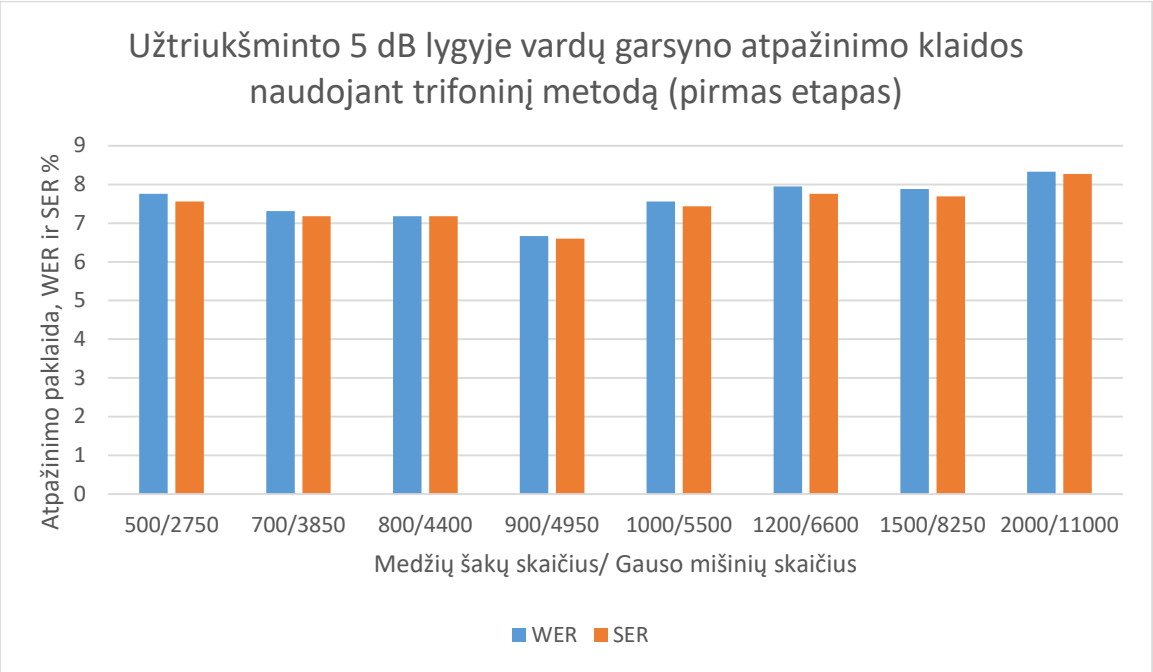
3.4.2 *Trifoninis metodas*

Pirmame etape keitėme *numLeavesTri1* perskaičiuodami parametrą *numGaussTri1*, kad būtų išlaikytas santykis 1:5,5. Gautos atpažinimo klaidos parodytos 7 lentelėje.

7 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (pirmas etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius							
	500/2750	700/3850	800/4400	900/4950	1000/5500	1200/6600	1500/8250	2000/11000
WER	7,76	7,31	7,18	6,67	7,56	7,95	7,88	8,33
SER	7,56	7,18	7,18	6,60	7,44	7,76	7,69	8,27

Mažiausia klaida gauta, kai *numLeavesTri1=900, numGaussTri1=4950.* 39 pav. pateikiamas rezultatų grafinis vaizdas.



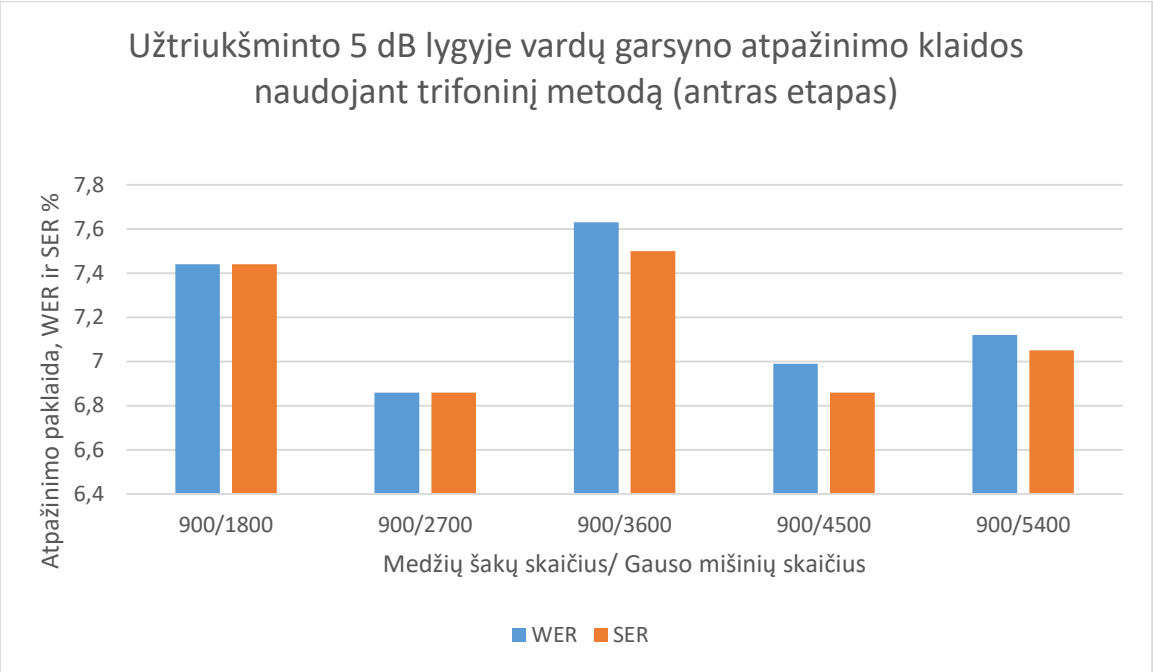
39 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (pirmas etapas), rezultatai

Antrame etape keitėme *numGaussTri1* išlaikydami pirmame etape surastą *numLeavesTri1*. Gautos atpažinimo klaidos parodytos 8 lentelėje.

8 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (antras etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius				
	900/1800	900/2700	900/3600	900/4500	900/5400
WER	7,44	6,86	7,63	6,99	7,12
SER	7,44	6,86	7,50	6,86	7,05

40 pav. pateikiamas gautų rezultatų grafinis vaizdas.



40 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant trifoninį metodą (antras etapas), rezultatai

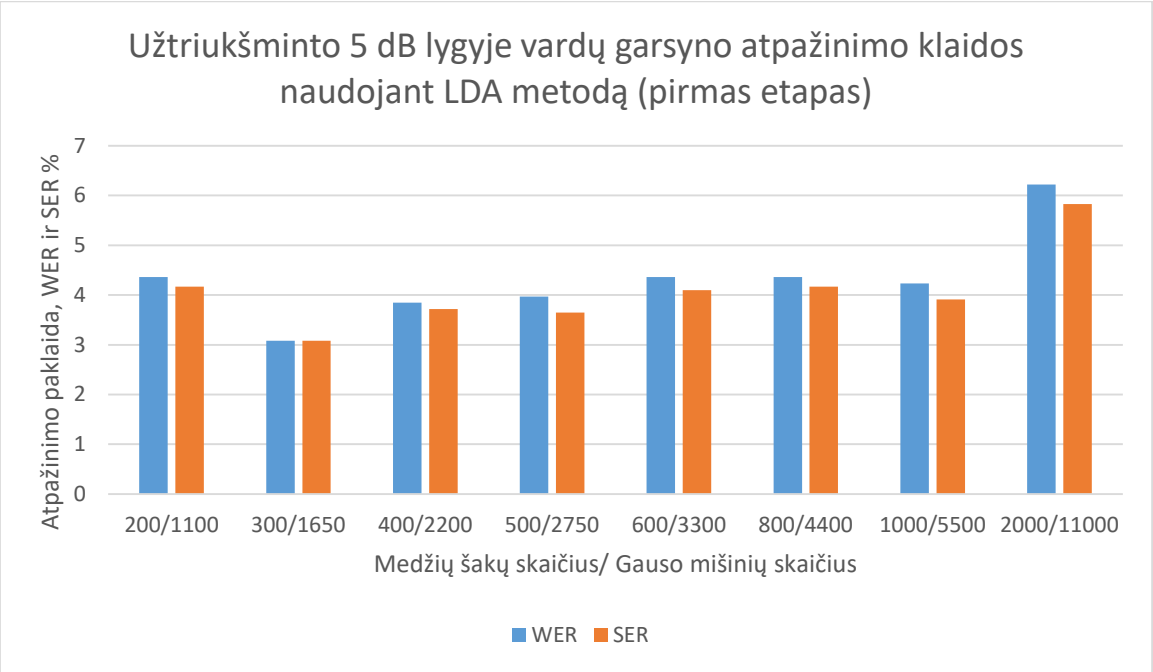
3.4.3 *LDA* metodas

Pirmame etape keitėme *numLeavesMLLT* perskaičiuodami parametrą *numGaussMLLT*, kad būtų išlaikytas santykis 1:5,5. Gautos atpažinimo klaidos parodytos 9 lentelėje.

9 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (pirmas etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius							
	200/1100	300/1650	400/2200	500/2750	600/3300	800/4400	1000/5500	2000/11000
WER	4,36	3,08	3,85	3,97	4,36	4,36	4,23	6,22
SER	4,17	3,08	3,72	3,65	4,10	4,17	3,91	5,83

41 pav. pateikiamas gautų rezultatų grafinis vaizdas.



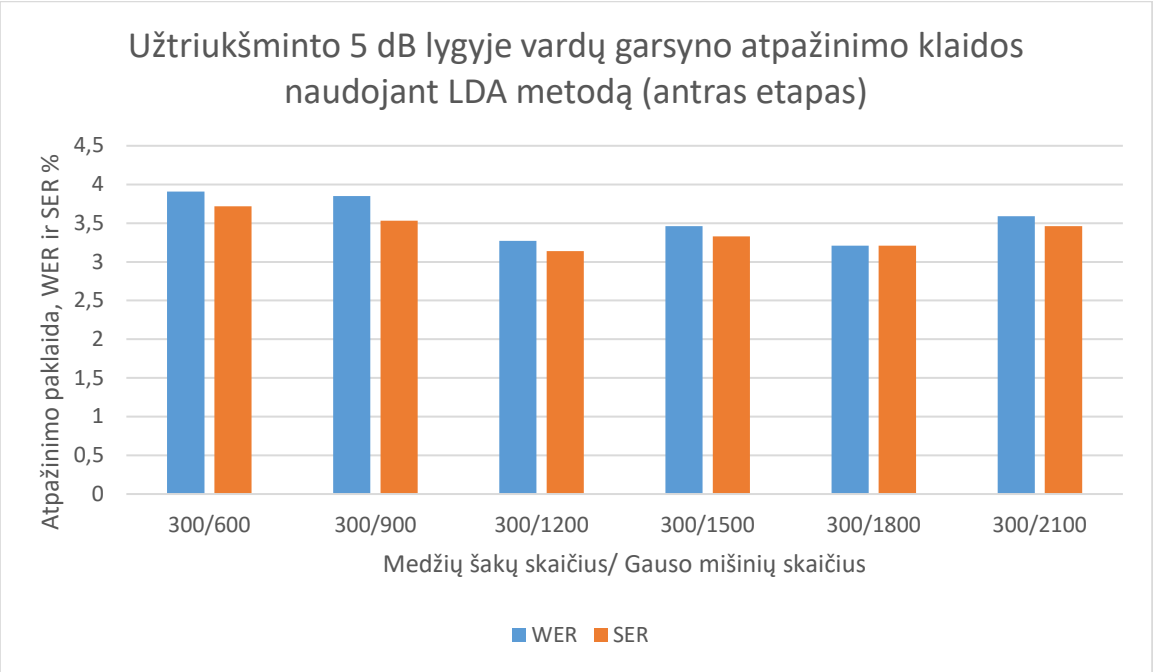
41 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (pirmas etapas), rezultatai

Antrame etape keitėme *numGaussMLLT* išlaikydami pirmame etape surastą *numLeavesMLLT*. Gautos atpažinimo klaidos parodytos 10 lentelėje.

10 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (antras etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius					
	300/600	300/900	300/1200	300/1500	300/1800	300/2100
WER	3,91	3,85	3,27	3,46	3,21	3,59
SER	3,72	3,53	3,14	3,33	3,21	3,46

42 pav. pateikiamas gautų rezultatų grafinis vaizdas.



42 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant LDA metodą (antras etapas), rezultatai

Mažiausia klaida gauta, kai *numLeavesMLLT=300, numGaussMLLT=1650.*

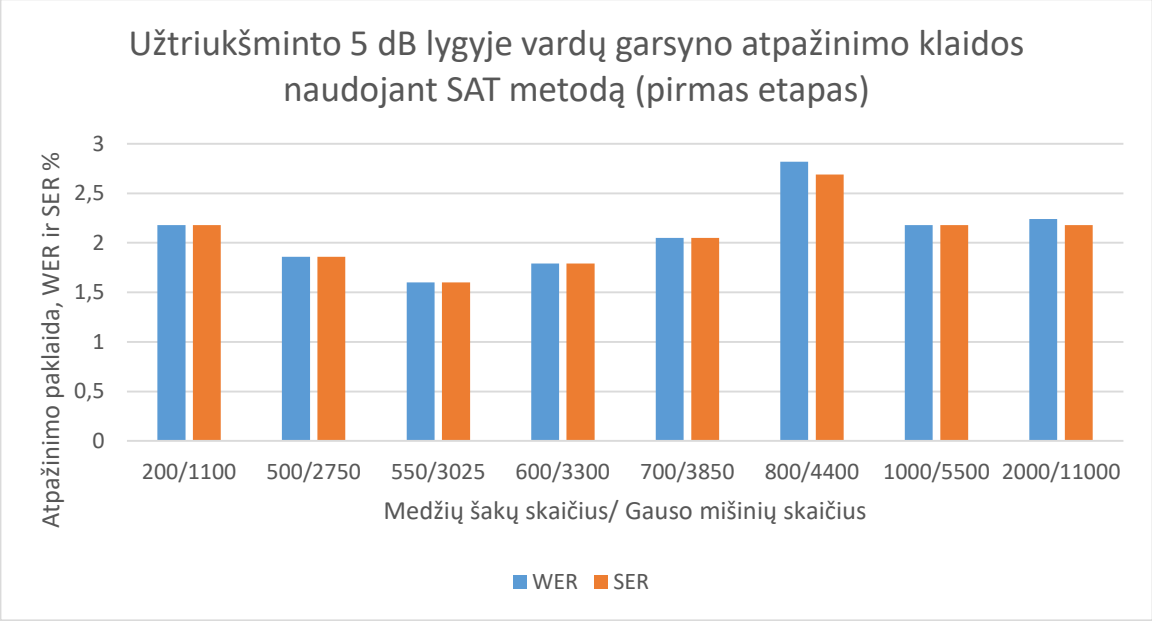
3.4.4 *SAT* metodas

Pirmame etape keitėme *numLeavesSAT* perskaičiuodami parametrą *numGaussSAT,* kad būtų išlaikytas santykis 1:5,5. Gautos atpažinimo klaidos parodytos 11 lentelėje.

11 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (pirmas etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius							
	200/1100	500/2750	550/3025	600/3300	700/3850	800/4400	1000/5500	2000/11000
WER	2,18	1,86	1,60	1,79	2,05	2,82	2,18	2,24
SER	2,18	1,86	1,60	1,79	2,05	2,69	2,18	2,18

43 pav. pateikiamas gautų rezultatų grafinis vaizdas.

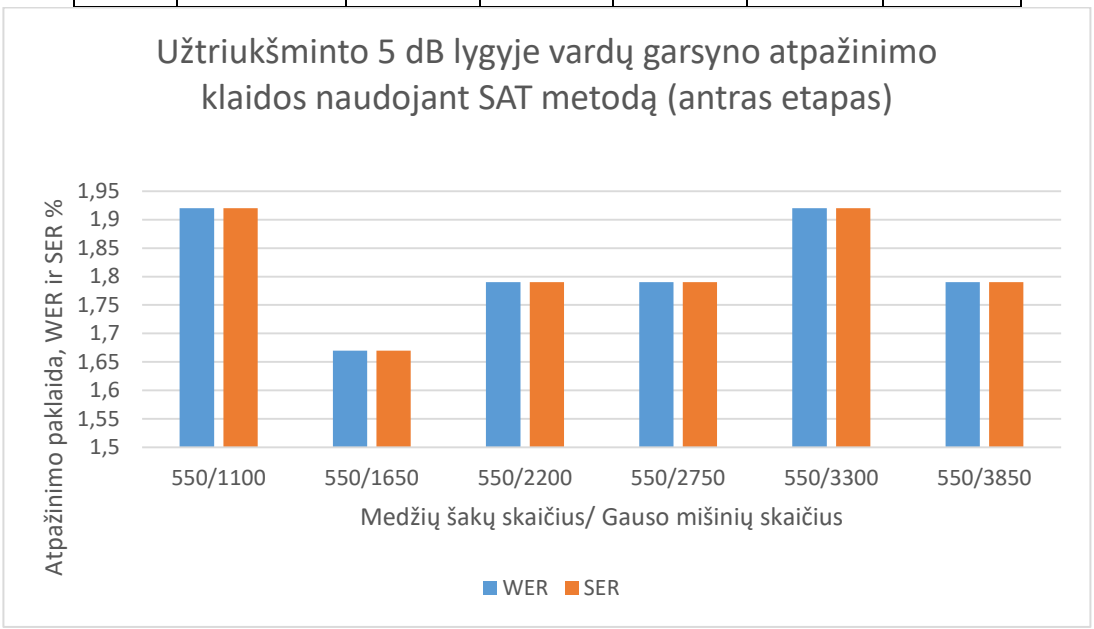


43 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (pirmas etapas), rezultatai

Antrame etape keitėme *numGaussSAT* išlaikydami pirmame etape surastą *numLeavesSAT*. Gautos atpažinimo klaidos parodytos 12 lentelėje. 44 pav. pateikiamas gautų rezultatų grafinis vaizdas.

12 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (antras etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių skaičius					
	550/1100	550/1650	550/2200	550/2750	550/3300	550/3850
WER	1,92	1,67	1,79	1,79	1,92	1,79
SER	1,92	1,67	1,79	1,79	1,92	1,79



44 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SAT metodą (antras etapas), rezultatai

Mažiausia klaida gauta, kai *numLeavesSAT=550*, *numGaussSAT=3025*.

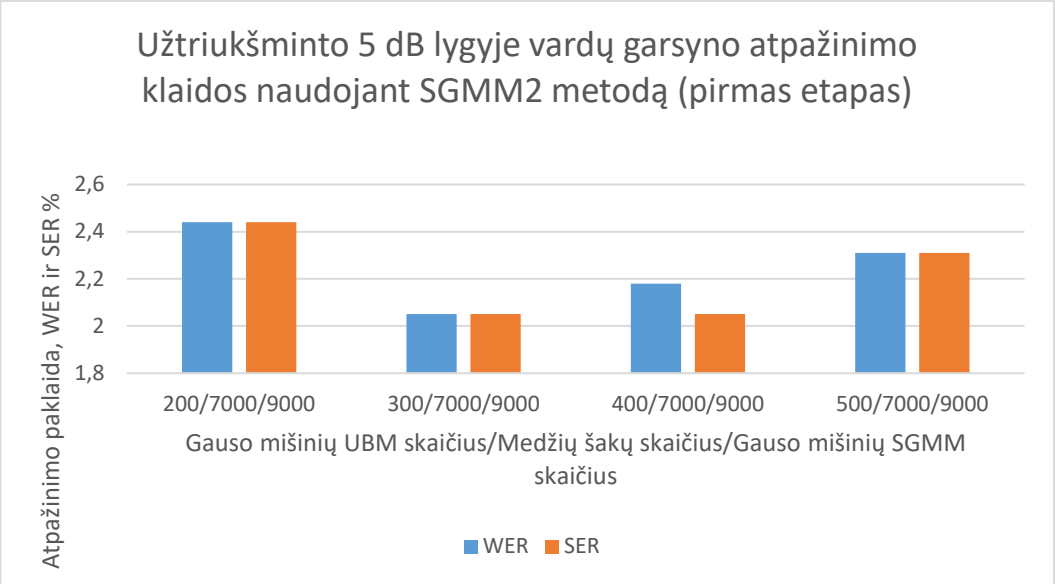
3.4.5 *SGMM2* metodas

Pirmame etape keitėme *numGaussUBM*, nekeičiant kitų parametrų. Gautos atpažinimo klaidos parodytos 13 lentelėje.

13 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (pirmas etapas)

Klaida	Gauso mišinių UBM skaičius/Medžių šakų skaičius/Gauso mišinių SGMM skaičius			
	200/7000/9000	300/7000/9000	400/7000/9000	500/7000/9000
WER	2,44	2,05	2,18	2,31
SER	2,44	2,05	2,05	2,31

45 pav. pateikiamas gautų rezultatų grafinis vaizdas.



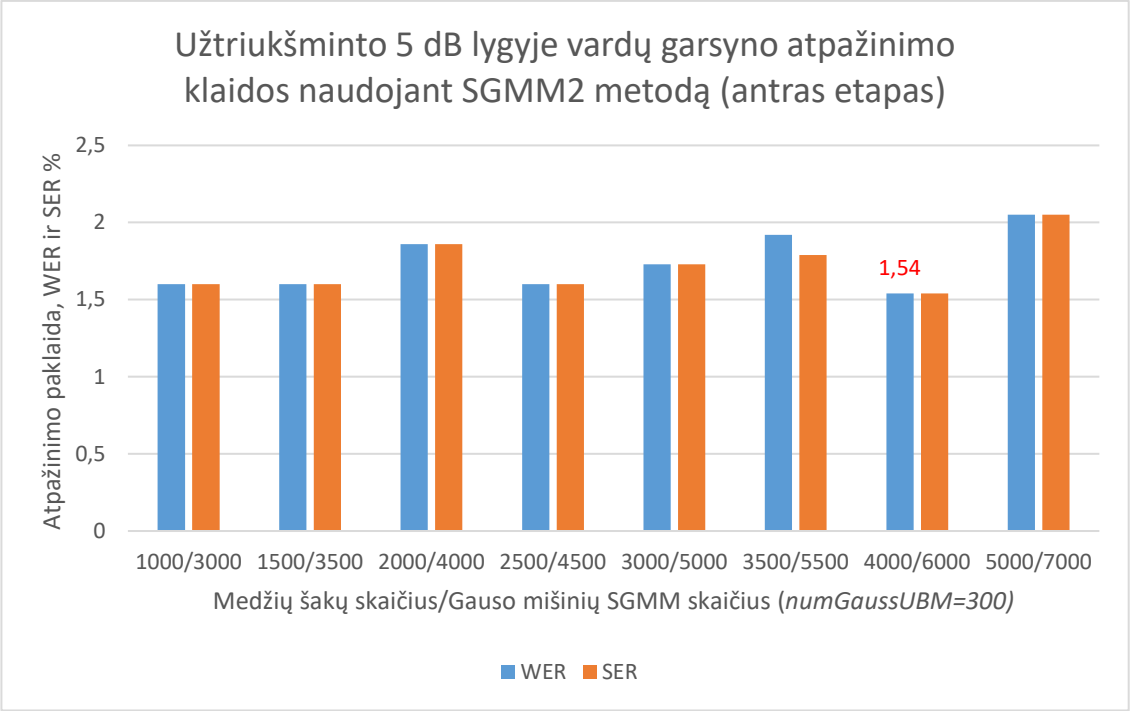
45 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (pirmas etapas), rezultatai

Antrame etape keitėme *numLeavesSGMM* išlaikydami *numGaussSGMM* 2000 vienetų didesnį už *numLeavesSAT*. Gautos atpažinimo klaidos parodytos 14 lentelėje. 44 pav. pateikiamas gautų rezultatų grafinis vaizdas.

14 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (antras etapas), rezultatai

Klaida	Medžių šakų skaičius/Gauso mišinių SGMM skaičius (*numGaussUBM=300)*							
	1000/3000	1500/3500	2000/4000	2500/4500	3000/5000	3500/5500	4000/6000	5000/7000
WER	1,60	1,60	1,86	1,60	1,73	1,92	1,54	2,05
SER	1,60	1,60	1,86	1,60	1,73	1,79	1,54	2,05

46 pav. pateikiamas gautų rezultatų grafinis vaizdas.



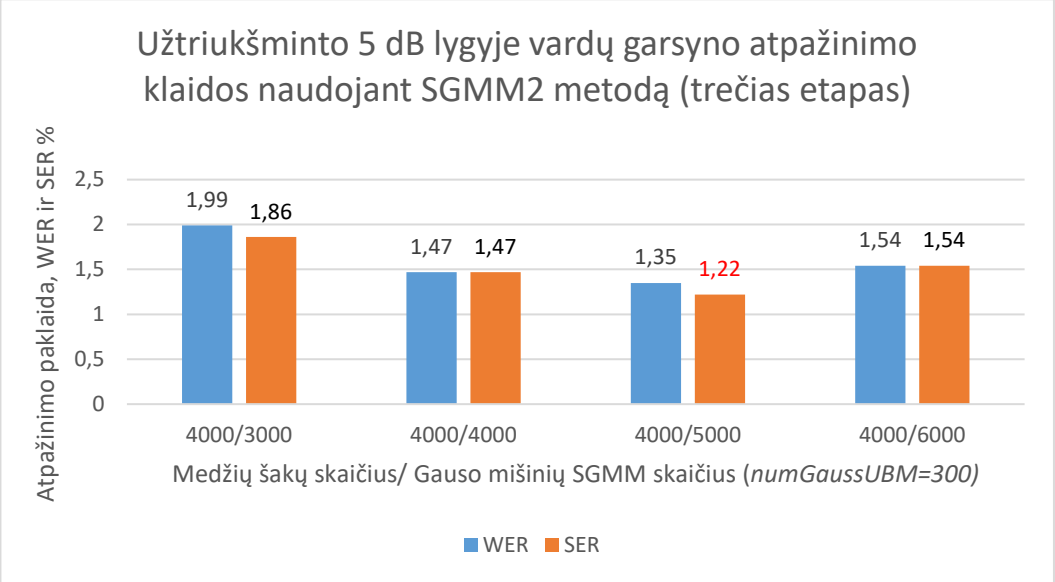
46 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (antras etapas), rezultatai

Trečiame etape keitėme *numGaussSGMM* išlaikant pastovų *numLeavesSGMM*. Gautos atpažinimo klaidos parodytos 15 lentelėje.

15 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (trečias etapas)

Klaida	Medžių šakų skaičius/ Gauso mišinių SGMM skaičius (*numGaussUBM=300)*			
	4000/3000	4000/4000	4000/5000	4000/6000
WER	1,99	1,47	1,35	1,54
SER	1,86	1,47	1,22	1,54

47 pav. pateikiamas gautų rezultatų grafinis vaizdas.



47 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant SGMM2 metodą (trečias etapas), rezultatai

Mažiausia klaida gauta, kai *numLeavesSGMM = 4000, numGaussSGMM = 5000.*

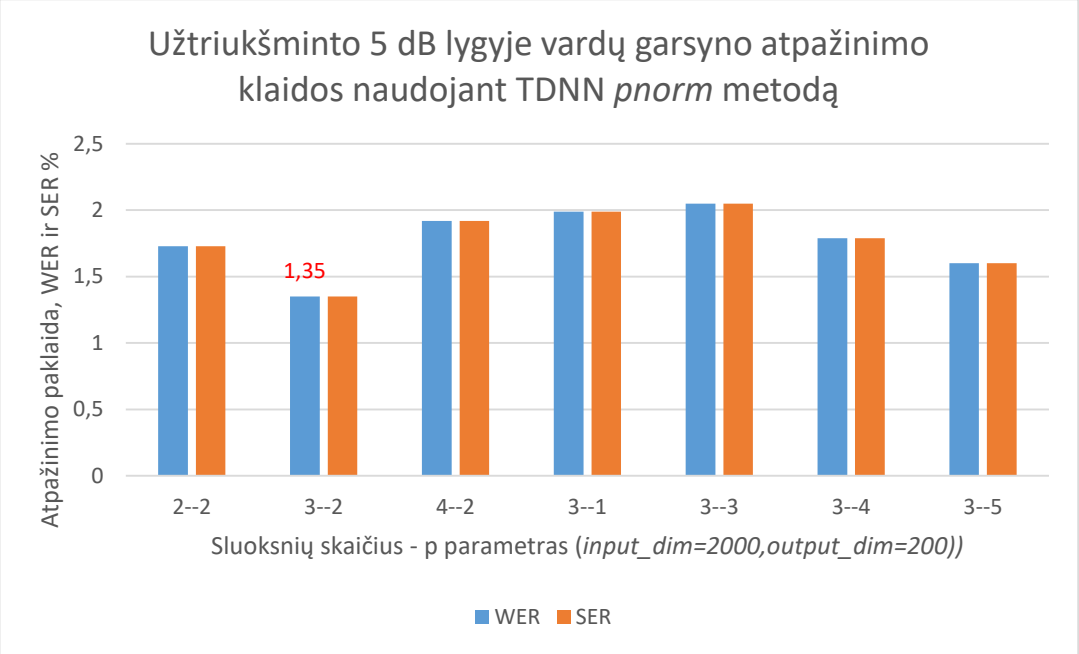
3.4.6 *TDNN* metodai

Tyrimams su TDNN *pnorm* metodu yra keičiamas paslėptųjų sluoksnių skaičius (2-4), parametras *p* (1 – 5), *pnorm_input_dim* ir *pnorm_output_dim*. Pasirinktos šios numatytosios reikšmės: 2-sluoksniai, p = 2, *pnorm_input_dim = 2000*, *pnorm_output_dim = 200*. Tyrimas atliktas su 20 epochų, kurios atstojo 20 iteracijų. Gautos atpažinimo klaidos parodytos 16 ir 17 lentelėse.

16 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *pnorm* metodą

Klaida	Sluoksnių skaičius - p parametras (*input_dim=2000,output_dim=200)*						
	2-2	3-2	4-2	3-1	3-3	3-4	3-5
WER	1,73	1,35	1,92	1,99	2,05	1,79	1,60
SER	1,73	1,35	1,92	1,99	2,05	1,79	1,60

48 pav. pateikiamas gautų rezultatų grafinis vaizdas.

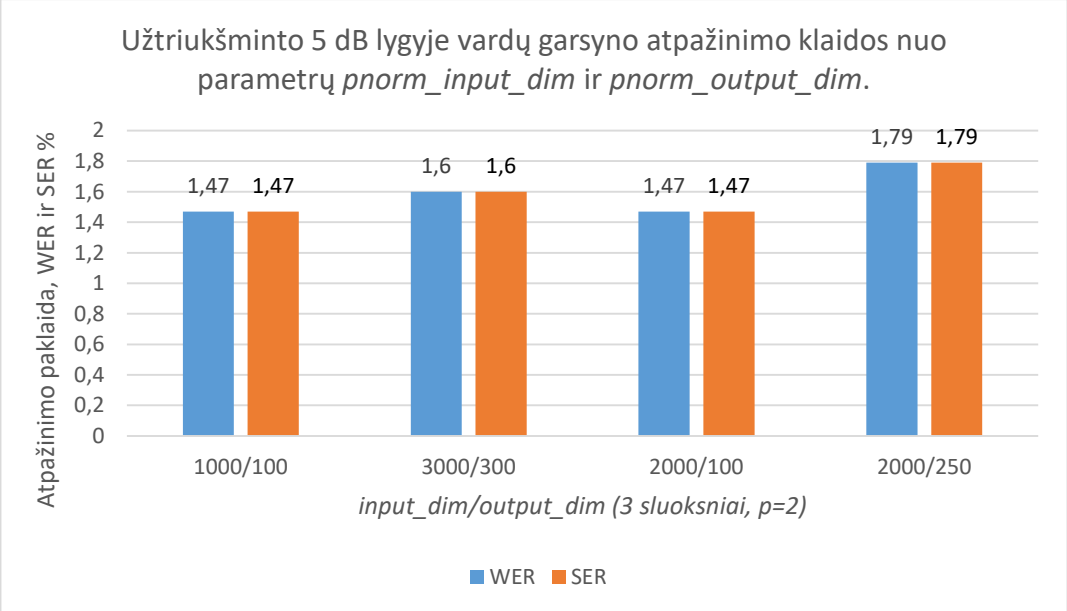


48 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *pnorm* metodą, rezultatai

17 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo parametrų *pnorm_input_dim* ir *pnorm_output_dim*.

Klaida	*input_dim/output_dim (3 sluoksniai, p=2)*			
	1000/100	3000/300	2000/100	2000/250
WER	1,47	1,60	1,47	1,79
SER	1,47	1,60	1,47	1,79

49 pav. pateikiamas gautų rezultatų grafinis vaizdas.



49 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo parametrų *pnorm_input_dim* ir *pnorm_output_dim*, rezultatai

Tyrimams su TDNN *tanh* metodu yra keičiamas paslėptųjų sluoksnių skaičius (1-4) ir *hidden_layer_dim*. Pasirinktos šios numatytosios reikšmės: 2 sluoksniai, *hidden_layer_dim=300.*

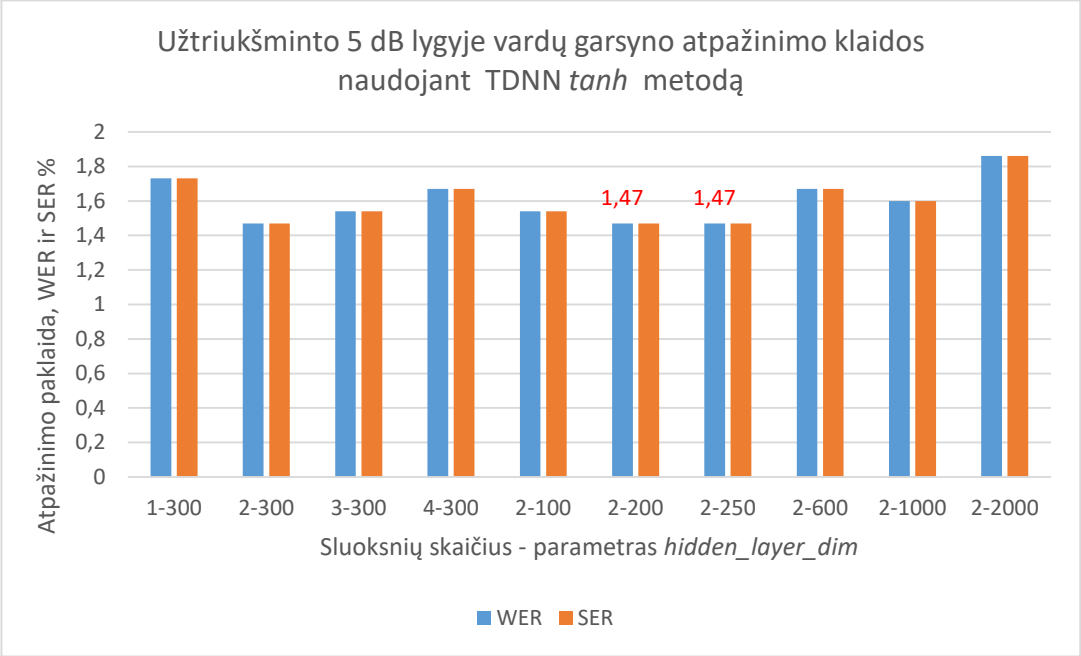
Tyrimas atliktas su 20 epochų, kurios atstojo 20 iteracijų. Gautos atpažinimo klaidos parodytos 18 lentelėje.

18 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *tanh* metodą

Klaida	Sluoksnių skaičius-parametras *hidden_layer_dim*									
	1-300	2-300	3-300	4-300	2-100	2-200	2-250	2-600	2-1000	2-2000
WER	1,73	1,47	1,54	1,67	1,54	1,47	1,47	1,67	1,60	1,86
SER	1,73	1,47	1,54	1,67	1,54	1,47	1,47	1,67	1,60	1,86

Mažiausia klaida gauta, kai sluoksnių skaičius = 2, o parametras *hidden_layer_dim=200, 250.*

50 pav. pateikiamas gautų rezultatų grafinis vaizdas.



50 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos naudojant TDNN *tanh* metodą, rezultatai

Atlikus tyrimus 5 atpažinimo metodais (*monofoninu, trifoniniu, LDA, SAT, SGMM2*) pastebėta, kad mažiausia vardų garsyno atpažinimo paklaida gaunama neuroninių tinklų, *TDNN p-norm* metodu, kai paslėptų sluoksnių skaičius – 3, o *p* parametras lygus 2 ir paklaida siekia vos 1,35 %.
TDNN tanh tyrimo metodas taip pat pasitvirtino. Paklaida lygi 1,47 %.
Didžiausia atpažinimo paklaida gauta monofoniniame režime, kai Gauso mišinių skaičius – 8000, WER- 19,04 %.

3.5 Vardų garsyno tyrimo su TensorFlow paketu, rezultatai.

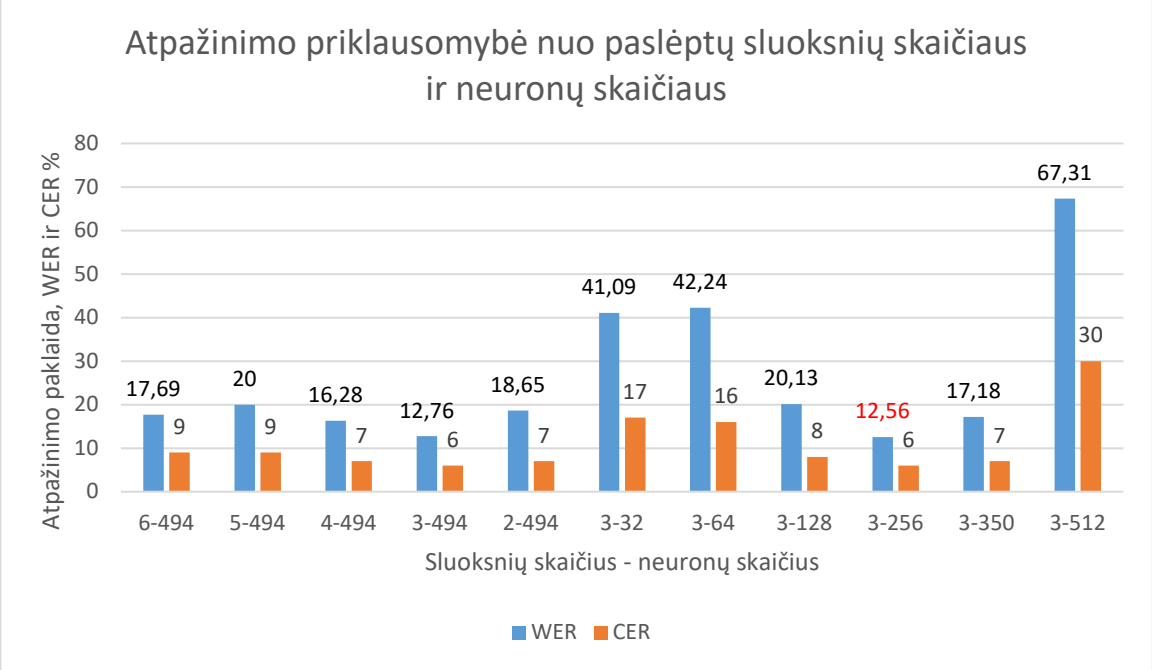
Užtriukšminto 5 dB lygyje vardų garsyno Vardai_18_3 atpažinimo tikslumo priklausomybės nuo paslėptųjų sluoksnių ir neuronų juose skaičiaus tyrimas vykdytas su DeepSpeech2 algoritmu.

Keičiamas sluoksnių skaičius ir neuronų skaičius, 6-494 – 19 epochų, kiti atvejai – 10 epochų. *Evaluation* rinkinys – sudaro 30% apmokymo duomenų rinkinio, atrinktas atsitiktiniu būdu. Tyrimo rezultatai pateikiami 19 lentelėje. Paslėptųjų sluoksnių skaičius buvo keičiamas nuo 2 iki 6, o neuronų skaičius – nuo 32 iki 512.

19 lentelė. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo tikslumo priklausomybės nuo paslėptųjų sluoksnių ir neuronų juose skaičiaus, tyrimo rezultatai

Klaida	Parametrai: pirmas sk.-sluoksniai, antras sk. - neuronai										
	6-494	5-494	4-494	3-494	2-494	3-32	3-64	3-128	3-256	3-350	3-512
WER	17,69	20,00	16,28	12,76	18,65	41,09	42,24	20,13	12,56	17,18	67,31
CER	9	9	7	6	7	17	16	8	6	7	30

Iš tyrimo rezultatų matyti, kad mažiausia atpažinimo paklaida gaunama, kai paslėptųjų sluoksnių skaičius – 3, o neuronų skaičius – 256. Esant šiems parametrams, žodžių atpažinimo paklaida siekia 12,56 %. Panaši paklaida gauta, kai paslėptųjų sluoksnių skaičius – 3, o neuronų skaičius – 494, kuri siekia 12,76 %. 51 pav. pateikiamas gautų rezultatų grafinis vaizdas.



51 pav. Užtriukšminto 5 dB lygyje vardų garsyno atpažinimo klaidos nuo paslėptų sluoksnių skaičiaus ir neuronų skaičiaus, rezultatai

Išvados

1. Atliktas neužtriukšminto lietuviškų vardų garsyno Vardai_18_3 tyrimas Kaldi programiniu paketu ir gauti rezultatai, kurie parodo, kad garsynas atpažįstamas praktiškai be klaidų, 99,94 % tikslumu. Paklaida gauta tik su *monofoniniu* ir *LDA* metodu, kuri yra maža – 0,06 %. Nuspręsta užtriukšminti garso įrašus 5 dB lygyje baltu triukšmu.

2. Tyrimai pakartoti su užtriukšmintu 5 dB lygyje garso įrašų rinkiniu Vardai_18_3. Panaudoti trys fonemų rinkiniai su dvibalsiais, grafemomis, SAMPA_LT. Mažiausia atpažinimo paklaida gauta su *TDNN pnorm* metodu – 1,47 %.

3. Atlikus kryžminį garso įrašų rinkinio Vardai_18_3 su 5 dB užtriukšminimu testavimą, mažiausias paklaidos vidurkis gautas su *TDNN-tanh* metodu - 2,16 %, o didžiausias *monofoniniu* metodu - 20,98 %.

4. Atlikus tyrimus 5 atpažinimo metodais (*monofoninu, trifoniniu, LDA, SAT, SGMM2*) pastebėta, kad mažiausia vardų garsyno atpažinimo paklaida gaunama neuroninių tinklų, *TDNN p-norm* metodu, kai paslėptų sluoksnių skaičius – 3, o *p* parametras lygus 2 ir atpažinimo paklaida siekia vos 1,35 %. Didžiausia paklaida gauta monofoniniu režimu, kai gauso mišinių skaičius lygus 8000, paklaida - 19,04 %.

5. Atlikus vardų garsyno Vardai_18_3 tyrimus su TensorFlow paketu, pastebėta, kad mažiausia atpažinimo paklaida gaunama, kai paslėptųjų sluoksnių skaičius – 3, o neuronų skaičius – 256. Esant šiems parametrams, žodžių atpažinimo paklaida siekia 12,56 %. Panaši paklaida gauta, kai paslėptųjų sluoksnių skaičius – 3, o neuronų skaičius – 494, kuri siekia 12,76 %.

6. Apžvelgus visus gautus tyrimų rezultatus, nuspręsta, kad Kaldi programinis paketas žymiai geriau atpažįsta vardų garsyną, *TDNN p-norm* metodu tikslumas siekia 98,65 %, palyginus su TensorFlow paketu – 87,24 % tikslumas.

Literatūros sąrašas

1. P. Kasparaitis. *Kompiuterinės lingvistikos įvadas. Kalbos atpažinimas.* 2010 [žiūrėta 2020-01-05]. Prieiga per: https://klevas.mif.vu.lt/~pijus/CLF/Atpazinimas.pdf
2. Prashant Upadhyaya, Sanjeev Kumar Mittal, Yash Vardhan Varshney, Omar Farooq, Musiur Raza Abidi. *Speaker Adaptive Model for Hindi Speech using Kaldi Speech Recognition toolkit.* 2018 [žiūrėta 2020-01-08]. Prieiga per: https://www.researchgate.net/publication/323239783_Speaker_Adaptive_Model_for_Hindi_Speech_using_Kaldi_Speech_Recognition_toolkit
3. Steven B. Davis and Paul Mermelstein. *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences.* 1980. [žiūrėta 2020-01-10]. Prieiga per: https://www.semanticscholar.org/paper/Comparison-of-Parametric-Representations-for-Word-Davis-Mermelstein/36cac502dd96788b7eef91bdef152d47b71bd1fb
4. *KALDI Window function.* 2016. [žiūrėta 2020-01-20]. Prieiga per: http://kaldi-asr.org/doc/structkaldi_1_1FeatureWindowFunction.html
5. Petras Kniūkšta. *Kompiuerinis lietuvių kalbos žinynas. Nuo morfologijos iki reikalų raštų*. 2004 [žiūrėta 2020-02-03]. Prieiga per: http://www.xn--altiniai-4wb.info/files/kalba/KE00/Lietuvi%C5%B3_kalbos_%C5%BEinynas._Gars%C5%B3_klasifikacija.KE1500.pdf
6. *Grafema.* 2017 [žiūrėta 2020-02-03]. Prieiga per: https://lt.wikipedia.org/wiki/Grafema
7. *SAMPA - computer readable phonetic alphabet*. 2005 [interaktyvus].[žiūrėta 2020-02-20]. Prieiga per: https://www.phon.ucl.ac.uk/home/sampa
8. A. Raškinis, G. Raškinis, A. Kazlauskienė. *SAMPA (Speech Assesment Methods Phonetic Alphabet) for encoding transcriptions of Lithuanian speech corpora.* 2003. Vytautas Magnus University. [žiūrėta 2020-02-20]. Prieiga per: https://www.vdu.lt/cris/handle/20.500.12259/55530
9. Li Deng Dong Yu. *Automatic speech recognition, a deep learning approach*. 2015. [žiūrėta 2020-02-25]. Prieiga per: http://125.234.102.146:8080/dspace/bitstream/DNULIB_52011/6853/1/automatic_speech_recognition_a_deep_learning_approach.pdf
10. Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. *Deep neural networks for acoustic modeling in speech recognition*. 2012. [žiūrėta 2020-02-28]. Prieiga per: https://research.google/pubs/pub38131/
11. Gonzalo Navarro. *A guided tour to approximate string matching.* 2001 [žiūrėta 2020-03-02]. Prieiga per: https://www.dcc.uchile.cl/TR/1999/TR_DCC-1999-005.pdf
12. *Kaldi projekto istorija*. [interaktyvus] 2010 [žiūrėta 2020-03-03]. Prieiga per: https://kaldi-asr.org/doc/history.html
13. Daniel Povey. *The Kaldi Speech Recognition Toolkit.* 2009 [žiūrėta 2020-03-04]. Prieiga per: http://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf
14. Kaldi apmokomasis tinklaraštis. *Kaldi for dummies.* [interaktyvus] [žiūrėta 2020-03-10]. Prieiga per: https://kaldi-asr.org/doc/kaldi_for_dummies.html

15. Kaldi programinio paketo atsisiuntimas ir įdiegimas. *Downloading and installing Kaldi*. [interaktyvus] [žiūrėta 2020-03-10]. Prieiga per: https://kaldi-asr.org/doc/install.html
16. *Tensorflow*. [interaktyvus] 2019 [žiūrėta 2020-03-12]. Prieiga per: https://en.wikipedia.org/wiki/TensorFlow
17. TensorFlow programinio paketo įdiegimas. *TensorFlow in Anaconda.* [interaktyvus] 2018 [žiūrėta 2020-03-13]. Prieiga per: https://www.anaconda.com/blog/tensorflow-in-anaconda
18. Spyder (Anaconda3) programinė aplinka. *Spyder Overview*. [interaktyvus] 2018 [žiūrėta 2020-03-15]. Prieiga per: https://docs.spyder-ide.org/overview.html
19. Kaldi garsyno paruošiamieji failai. [interaktyvus] [žiūrėta 2020-03-18]. Prieiga per: https://kaldi-asr.org/doc/kaldi_for_dummies.html#kaldi_for_dummies_acoustic
20. Sox programinis įrankis. *Sox- Sound Exchange.* [interaktyvus] 2015 [žiūrėta 2020-03-20]. Prieiga per: http://sox.sourceforge.net/