KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Laurynas Pipiras

LIETUVIŲ KALBOS ATPAŽINIMAS NAUDOJANT GILŲJĮ MOKYMĄSI

Baigiamasis magistro projektas

Vadovas

prof. dr. Rytis Maskeliūnas

KAUNAS, 2019

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

LIETUVIŲ KALBOS ATPAŽINIMAS NAUDOJANT GILŲJĮ MOKYMĄSI

Baigiamasis magistro projektas

Informatikos studijų programa (kodas 6211BX007)

Vadovas

(parašas) prof. dr. Rytis Maskeliūnas

(data)

Recenzentas

(parašas) doc. dr. Vytautas Rudžionis

(data)

Projektą atliko

(parašas) Laurynas Pipiras

(data)

KAUNAS, 2019

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Laurynas Pipiras

(Studento vardas, pavardė)

Informatikos studijų programa, 6211BX007

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Lietuvių kalbos atpažinimas naudojant gilųjį mokymąsi“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.
Kaunas

Patvirtinu, kad mano, Lauryno Pipiro, baigiamasis projektas tema „Lietuvių kalbos atpažinimas naudojant gilųjį mokymąsi“ 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)*

Pipiras Laurynas. „Lietuvių kalbos atpažinimas naudojant gilųjį mokymąsi“. Magistro baigiamasis projektas / vadovas prof. dr. Rytis Maskeliūnas; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Fiziniai mokslai, Informatika

Reikšminiai žodžiai: *automatinis lietuvių šnekos atpažinimas, gilus mokymasis, dirbtiniai neuroniniai tinklai*

Kaunas, 2019. 69 p.

SANTRAUKA

Nuo pat kompiuterių atsiradimo pradžios automatinio šnekos atpažinimo sritis buvo vienas didžiausių ir sudėtingiausių informatikos uždavinių. Didžioji dalis šioje srityje atliekamų tyrimų yra susiję su plačiai paplitusiomis kalbomis, pvz., anglų, prancūzų, kinų ir t.t. Istorinės priežastys, sudėtinga kalbos struktūra ir duomenų trūkumas lėmė mažą susidomėjimą lietuvių šnekos atpažinimu. Gilaus mokymosi metodai lietuvių šnekos atpažinimui iki šiol buvo taikyti labai konservatyviai, o šnekos atpažinimas iš fonemų sekų iki šiol nebuvo tirtas. Šio darbo tikslas – sukurti automatinio lietuvių šnekos atpažinimo sistemą, galinčią identifikuoti ištartus žodžius iš jų fonemų sekų panaudojant gilaus mokymosi algoritmus. Darbe apžvelgiami iki šiol atlikti su automatiniu lietuvių šnekos atpažinimu susiję tyrimai, giliųjų dirbtinių neuroninių tinklų ir specialaus jų tipo – rekurentinių neuroninių tinklų – struktūra ir veikimas. Tyrimo metu pasirinkta naudoti du populiarius rekurentinių neuroninių tinklų tipo modelius, gebančius dirbti su kintamo ilgio įvesties ir išvesties sekomis: tradicinį užkoduotojo-dekoduotojo tipo modelį ir modelį su dėmesio mechanizmu. Eksperimentais įvertintos tokių modelių galimybės atpažinti tiek izoliuotus žodžius, tiek rišlią kalbą. Darbe analizuojama, kaip kinta atpažinimo tikslumas nuo apdorojamos fonemų sekos ilgio, taip pat atlikti papildomi eksperimentai, kuriais buvo siekiama patikrinti, kaip modelių veikimą lemia įvairūs tyrėjų pasiūlyti metodai, galintys pagerinti modelių veikimą. Modelių prognozavimo efektyvumas įvertintas ir kryžminės patikros būdu.

Pipiras Laurynas. *Lithuanian Speech Recognition Using Deep Learning*: Master‘s thesis in Informatics / supervisor assoc. prof. Rytis Maskeliūnas. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Physical Science / Informatics

Key words: *automatic Lithuanian speech recognition, encoder-decoder models, deep learning, artificial neural networks*

Kaunas, 2019. 69 p.

SUMMARY

Since the early days of modern computer science, automatic speech recognition has been one of the biggest and hardest challenges in the field. A large majority of research that was conducted in this area focused on the most widely spoken languages, such as English, French, Mandarin Chinese, etc. Due to complicated language structure and scarcity of data, the problems of automatic Lithuanian language speech recognition attracted very little attention. Until now Deep Learning methods were rarely used to solve this problem, while phoneme-based automatic speech recognition has not been investigated at all. The purpose of this project was to create an automatic speech recognition system for the Lithuanian language, that would be based on Deep Learning methods and could identify spoken words purely from their phoneme sequences. A survey of previous research work in the automatic Lithuanian language speech recognition field as well as structure and behavior of Recurrent Neural Networks were examined and are presented in this thesis. Due to their ability to work with variable length input and output sequence pairs, two encoder-decoder models were selected to solve the automatic speech recognition task: traditional encoder-decoder model and model with attention mechanism. The performance of these models was evaluated in isolated speech recognition and continuous speech recognition tasks. Additional experiments were conducted to determine, how the methods proposed by other researchers can increase accuracy. Finally, cross-validation technique was used to assess predictive performance of these models.

Turinys

Įvadas ... 11
Projekto aktualumas ... 11
Darbo tikslai ir uždaviniai ... 11
Dokumento struktūra ... 12
1 Uždavinio sprendimo analizė ... 13
1.1 Automatinis kalbos atpažinimas ... 13
1.2 Tradicinis automatinio kalbos atpažinimo sistemos modelis ... 14
1.2.1 Požymių išgavimas ... 14
1.2.2 Akustiniai modeliai ... 15
1.2.3 Tarties modelis ... 15
1.2.4 Kalbos modelis ... 15
1.2.5 Ištartų žodžių nustatymas ... 15
1.3 Šnekos atpažinimas nuo signalo vieneto pradžios iki galo ... 16
1.4 Lietuvių kalbos atpažinimas ... 16
1.4.1 „Google“ teikiamos ASR paslaugos lietuvių kalbai ... 17
1.4.2 Projektas LIEPA ... 17
1.4.3 Izoliuotų žodžių atpažinimas ... 17
1.4.4 Transliuojamos lietuvių kalbos atpažinimas ... 18
1.4.5 Telefoninių pokalbių apdorojimas ... 18
1.4.6 Kitų kalbų modelių pritaikymas ... 19
1.4.7 Kiti tyrimai ... 19
1.5 Fonemomis remtos ASR sistemos ... 20
2 PĮ projektavimas ... 21
2.1 Reikalavimai programinei įrangai ... 21
2.1.1 Funkciniai reikalavimai ... 21
2.1.2 Nefunkciniai reikalavimai ... 21
2.2 Kokybės įvertinimas ... 21
2.2.1 Kokybės kriterijai ... 21
2.2.2 Kryžminė patikra ... 22
2.3 Panaudos atvejų diagrama ... 23
2.4 Programinės įrangos platformų analizė ... 24
3 Realizacija ... 26
3.1 Apie garsyną ... 26
3.2 Garsyno analizė ... 27
3.2.1 Fonemos ... 27
3.2.2 Grafemos ... 28
3.2.3 Specialūs simboliai ... 28
3.3 Gilus mokymasis ... 28
3.3.1 Dirbtiniai neuroniniai tinklai ... 28
3.3.2 Rekurentiniai neuroniniai tinklai ... 31

3.3.3 RNN apmokymas ... 34
3.4 Užkoduotojo-dekoduotojo tipo modeliai ... 34
3.4.1 Apmokymo per prievartą metodas ... 35
3.4.2 Mokymasis naudojant atbulas sekas ... 36
3.4.3 Gradiento apkarpymas ... 36
3.5 Modeliai su dėmesiu ... 36
3.6 Reguliarizacijos strategijos ... 37
3.6.1 Dirbtinių neuronų iškritimas ... 38
3.6.2 Ankstyvas stabdymas ... 38
4 Eksperimentai ... 39
4.1 Pasiruošimas eksperimentams ... 39
4.1.1 Mokymosi, patikros ir testavimo rinkinių sudarymas ... 39
4.1.2 Pradinis sekų apdorojimas ... 41
4.1.3 Sekų „paminkštinimas“ ... 41
4.1.4 Vienetinis kodavimas ... 42
4.1.5 Modeliai eksperimentams ... 43
4.2 Eksperimentai su izoliuotais žodžiais ... 43
4.2.1 Įprastas apmokymas ... 43
4.2.2 Apmokymas atbulomis sekomis ... 47
4.2.3 Tikslumas naudojant testavimo imtį ... 49
4.3 Eksperimentai su rišlia kalba ... 50
4.4 Kryžminė patikra ... 54
4.4.1 Izoliuotų žodžių rinkinys ... 55
4.4.2 Nuoseklios kalbos rinkinys ... 55
5 Išvados ... 58
6 Literatūros sąrašas ... 59
7 Priedai ... 62
7.1 Priedas A ... 62
7.2 Priedas B ... 68
7.3 Priedas C ... 69

Lentelių sąrašas

3.1 lentelė Įrašus sudarančių fonemų ir grafemų sekų analizė 26
4.1 lentelė Įprasto apmokymo metu *Seq2Seq* tipo modelių geriausi pasiekti rezultatai 44
4.2 lentelė Skirtingų konfigūracijų modelių su dėmesio mechanizmu geriausi pasiekti rezultatai 45
4.3 lentelė *Seq2Seq* tipo modelių su dirbtinių neuronų iškritimu geriausi pasiekti rezultatai 46
4.4 lentelė *Attention* tipo modelių su dirbtinių neuronų iškritimu geriausi pasiekti rezultatai 47
4.5 lentelė *Seq2Seq* tipo modelių geriausi pasiekti rezultatai naudojant atbulas sekas 48
4.6 lentelė *Attention* tipo modelių geriausi pasiekti rezultatai naudojant atbulas sekas 49
4.7 lentelė Modelių rezultatai su testavimo duomenų rinkiniu 49
4.8 lentelė Modelių sugeneruotų žodžių pavyzdžiai 50
4.9 lentelė Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio fonemų sekomis, geriausi pasiekti rezultatai 51
4.10 lentelė Modelio be dėmesio mechanizmo tikslumo įverčių pasikliautininieji intervalai 52
4.11 lentelė Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio fonemų sekomis, geriausi pasiekti rezultatai 52
4.12 lentelė Modelio su dėmesio mechanizmu tikslumo įverčių pasikliautinieji intervalai 53
4.13 lentelė Modelių sugeneruotų frazių pavyzdžiai, apmokius modelius fonemų sekomis, apribotomis iki 12 fonemų 54
4.14 lentelė Modelių sugeneruotų frazių pavyzdžiai, apmokius modelius fonemų sekomis, apribotomis iki 28 fonemų 54
7.1 lentelė Panaudos atvejo „Nurodyti hiperparametrų failą“ aprašymas 62
7.2 lentelė Panaudos atvejo „Nurodyti modelio katalogą“ aprašymas 62
7.3 lentelė Panaudos atvejo „Apmokyti modelį“ aprašymas 63
7.4 lentelė Panaudos atvejo „Ištestuoti modelį“ aprašymas 63
7.5 lentelė Panaudos atvejo „Nurodyti mokymosi duomenų rinkinį“ aprašymas 63
7.6 lentelė Panaudos atvejo „Nurodyti testavimo duomenų rinkinį“ aprašymas 64
7.7 lentelė Panaudos atvejo „Generuoti įprastus rinkinius“ aprašymas 64
7.8 lentelė Panaudos atvejo „Generuoti kryžminės patikros rinkinius“ aprašymas 65
7.9 lentelė Panaudos atvejo „Prognozuoti frazes“ aprašymas 65
7.10 lentelė Panaudos atvejo „Nurodyti fonemų sekų rinkinį“ aprašymas 66
7.11 lentelė Panaudos atvejo „Nurodyti rezultatų katalogą“ aprašymas 66
7.12 lentelė Panaudos atvejo „Nurodyti fonemų žodyną“ aprašymas 66
7.13 lentelė Panaudos atvejo „Nurodyti grafemų žodyną“ aprašymas 67
7.14 lentelė Duomenų rinkinių statistika 69

Paveikslėlių sąrašas

1.1 pav. Automatinės kalbos atpažinimo struktūros pavyzdys [1] 14
2.1 pav. Sistemos panaudos atvejų diagrama 24
3.1 pav. Fonemų ir jas atitinkančių grafemų sekos pavyzdys 26
3.2 pav. Įrašų ilgiai fonemomis ir grafemomis 26
3.3 pav. Fonemų ir grafemų sekų koreliacija 27
3.4 pav. Dirbtinio neurono pavyzdys [37] 29
3.5 pav. Gilus dirbtinis neuroninis tinklas su N+1 sluoksnių (N-1 iš jų yra paslėpti)[38] 30
3.6 pav. Supaprastinta rekurentinio neuroninio tinklo diagrama [40] 32
3.7 pav. Užkoduotojo-dekoduotojo tipo architektūros modelis 35
3.8 pav. Apmokymo per prievartą taikymas užkoduotojo-dekoduotojo tipo modeliuose 35
3.9 pav. Modelis su dėmesio mechanizmu 37
4.1 pav. Įrašų ilgių pasiskirstymas skirtinguose duomenų rinkiniuose 39
4.2 pav. Žodžių ilgių pasiskirstymas skirtinguose duomenų rinkiniuose. 40
4.3 pav. Izoliuotų žodžių fonemų sekų sudarymo pavyzdys 40
4.4 pav. Makro-sekų sudarymo iš fonemų sekų pavyzdys 41
4.5 pav. Sekų paminkštinimo pavyzdys (</> sekos pabaigos, < _ > „paminkštinimo“ simboliai) ... 42
4.6 pav. *Seq2Seq-2x64* tikslumo ir nuostolių funkcijos reikšmių grafikai 44
4.7 pav. *Attention-4x32* tikslumo ir nuostolių funkcijos reikšmių grafikai 45
4.8 pav. Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio sekomis, tikslumo grafikas 51
4.9 pav. Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio sekomis, WER grafikas 51
4.10 pav. Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio sekomis, tikslumo grafikas 53
4.11 pav. Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio sekomis, WER grafikas 53
4.12 pav. *Seq2Seq-2x128* ir *Attention-2x64* kryžminės patikros rezultatai (izoliuotų žodžių rinkinys) 55
4.13 pav. *Seq2Seq-2x128* modelio kryžminės patikros metu gautų tikslumo įverčių grafikas 56
4.14 pav. *Seq2Seq-2x128* modelio kryžminės patikros metu WER įverčių grafikas 56
4.15 pav. *Attention-2x64* modelio kryžminės patikros metu gautų tikslumo įverčių grafikas 57
4.16 pav. *Attention-2x64* modelio kryžminės patikros metu gautų WER įverčių grafikas 57
7.1 pav. Fonemų pasiskirstymo garsyne grafikas. 68
7.2 pav. Grafemų pasiskirstymo garsyne grafikas. 68
7.3 pav. Makro-sekose esančių žodžių kiekio pasiskirstymas duomenų rinkiniuose 69
7.4 pav. Makro-sekų ilgių pasiskirstymas duomenų rinkiniuose 69

Santrumpų žodynas

BRNN	(angl. *Bidirectional Recurrent Neural Networks*) dvikrypčiai rekurentiniai neuroniniai tinklai
ANN	Dirbtiniai neuroniniai tinklai (angl. *Artificial Neural Networks*)
GMM	Gauso mišinių modeliai (angl. *Gaussian mixture models*)
GRU	(angl. *Gated Recurrent Unit*)
LSTM	(angl. *Long Short-Term Memory*)
MFCC	Melų dažnių skalės kepstriniai koeficientai (angl. *Mel-Frequency Cepstral Coefficients*)
PLCP	Percepcinio tiesinio prognozavimo koeficientai (angl. *Perceptual linear predictive coeffiecients*)
HMM	Paslėpti Markovo modeliai (angl. *Hidden Markov Models*)
RNN	Rekurentiniai neuroniniai tinklai (angl. *Recurrent Neural Networks*)
WER	Klaidingų žodžių rodiklis (angl. *Word Error Rate*)

Įvadas

Projekto aktualumas

Augant kompiuterių galimybėms kartu auga ir kalbos atpažinimo technologijų pritaikymas kasdieninėje buityje: asmeniniuose kompiuteriuose, išmaniuosiuose telefonuose ir laikrodžiuose, automobiliuose, išmaniuosiuose namuose ir pan. Didžiulė tokių sistemų problema yra ta, kad kompanijos ir mokslininkai orientuojasi į plačiai paplitusias kalbas (anglų, kinų, ispanų ir pan.), o mažiau paplitusių kalbų, tarp jų ir lietuvių kalbos, vartotojams tokie produktai nėra pritaikyti. Todėl jiems lieka dvi išeitys: laukti ir tikėtis, kad produktuose naudojamas kalbos atpažinimo modelis greitu metu bus papildytas pageidaujama kalba, arba prisitaikyti ir naudoti užsienio kalbą sąveikaujant su produktais. Tačiau net ir geras kalbos mokėjimas negarantuoja gerų rezultatų – akcentai, aplinkos triukšmas, sintaksės ar leksikos klaidos ir panašūs kintamieji apsunkina naudojimąsi tokiomis sistemomis. Toks sprendimas taip pat sukuria naują problemą – pradėję su produktais bendrauti tik užsienio kalbomis, mes pasmerkiame savo gimtąją kalbą skaitmeniniam išnykimui. Nustoję naudoti kalbą, mes paverčiame ją neefektyvia ir nereikalinga, kadangi sunaikiname pagrindinę jos kaip bendravimo įrankio paskirtį. Visos šios priežastys lemia augantį automatinių kalbos apdorojimo sistemų, kurios pasiektų didelį tikslumą ir būtų pritaikytos mūsų gimtajai kalbai, poreikį.

Iki šiol kalbos atpažinimui atlikti buvo naudojami tiesiniai mašininio mokymosi modeliai, tokie kaip vektorinės atramos mašinos ar logistinė regresija. Pastaruoju metu suintensyvėjus dirbtinio intelekto srities tyrimams juos ėmė keisti daug tikslesni ir galingesni, giliaisiais neuroniniais tinklais paremti modeliai. Giliųjų neuroninių tinklų modeliai pasižymi abstraktumu, todėl juos galima nesunkiai modifikuoti ir pritaikyti įvairioms kalboms. Magistro baigiamuoju darbu tikimasi pritaikyti giliuosius neuroninius tinklus lietuvių kalbos atpažinimui ir įvertinti jų efektyvumą. arbo tikslas ir uždaviniai

Darbo tikslai ir uždaviniai

Pagrindinis magistrinio darbo tikslas – sukurti lietuvių šnekai skirtą automatinio kalbos atpažinimo sistemą, paremtą gilaus mokymosi metodais, kuri leistų atpažinti ištartus lietuviškus žodžius iš juos sudarančių fonemų sekų.

Darbo uždaviniai:

1. bendro automatinio kalbos atpažinimo modelio ir su lietuvių kalba susijusių tyrimų ir jai skirtų produktų analizė;
2. gilaus mokymosi algoritmų taikymo automatinio kalbos atpažinimo uždaviniams spręsti analizė;
3. uždavinio realizacija panaudojant gilaus mokymosi metodais paremtą modelį;
4. eksperimentais paremtas sukurtos sistemos tikslumo ir efektyvumo nustatymas izoliuotų žodžių atpažinimui;
5. eksperimentais paremtas sukurtos sistemos tikslumo ir efektyvumo nustatymas nenutrūkstamos šnekos atpažinimui.

Dokumento struktūra

Baigiamojo darbo ataskaitą sudaro keturi skyriai, aprašantys ASR sistemos kūrimą lietuvių šnekai ir eksperimentus, kuriais buvo siekiama įvertinti sukurtos sistemos efektyvumą.

Pirmajame skyriuje pristatomas automatinio šnekos atpažinimo uždavinys ir apibendrintas automatinio kalbos atpažinimo sistemos modelis. Pateikiama atlikta esamų sprendimų analizė automatinio lietuvių kalbos atpažinimo uždaviniui spręsti.

Antrajame skyriuje pateikti funkciniai ir nefunkciniai reikalavimai, kokybės metrikos, panaudos atvejų diagrama ir programinės įrangos platformų analizė.

Trečiajame skyriuje pateikta ASR sistemos vystymui naudoto garsyno analizė, pristatyti gilieji neuroniniai tinklai ir jų veikimas. Skyriuje taip pat glaudžiai aptariami rekurentiniai neuroniniai tinklai ir užkoduotojo-dekoduotojo tipo modeliai: tradicinis modelis ir modelis su dėmesio mechanizmu. Apžvelgiamos reguliarizacijos strategijos ir kitų tyrėjų pasiūlytos metodikos, galinčios pagerinti tokių modelių veikimą.

Ketvirtajame skyriuje aprašytas pasiruošimas eksperimentams ir pateikti izoliuotų žodžių ir nuoseklios kalbos eksperimentų rezultatai. Skyriuje taip pat aptariami vykdytos kryžminės patikros eksperimentai ir jų rezultatai.

Darbo pabaigoje pateiktos suformuluotos darbo metu gautos išvados.

1 Uždavinio sprendimo analizė

1.1 Automatinis kalbos atpažinimas

Automatinis kalbos atpažinimas (angl. *Automatic Speech Recognition*, ASR) – kompiuterinės lingvistikos sritis, nagrinėjanti metodus ir technologijas, leidžiančius paversti žmogaus sakomą kalbą (šneką) tekstu. ASR pradėtas tirti dar praeito amžiaus viduryje ir yra vienas didžiausių natūralios kalbos apdorojimo srities sprendžiamų uždavinių. Pastaraisiais metais ši sritis sulaukė didelio dėmesio dėl atpažinimo metodų ir skaičiavimo resursų pažangos.

ASR technologija turi dideles pritaikymo galimybes ir yra naudojama:

- balso paieškos programose;
- diktavimo sistemose, kurios naudojamos medicinos ir teisės srityse transkripcijai ir stenografavimui;
- automatinių subtitrų kūrimui televizijose ir kitose turinio platinimo sistemose;
- skambučių centruose;
- prietaisų valdymui, pakeičiant įprastą, jutimu grįstą sąveiką balsu paremta sąveika.

Automatinis kalbos atpažinimas taip pat yra daugybės sudėtingesnių uždavinių, pvz.: kalbos vertimo, kalbos apibendrinimo, sentimentų analizės ir pan., pagrindas.

Automatinio kalbos atpažinimo programos turi didelę paklausą tarp fizinę ar psichinę negalią turinčių žmonių, nes palengvina tokių žmonių integraciją į visuomenę. Įvairios sistemos, leidžiančios žmonėms bendrauti su kompiuteriais ir valdyti juos balsu, jau daugelį metų yra kuriamos ir tobulinamos, siekiant ne tik pagerinti tokių sistemų tikslumą, bet ir padaryti jas patogesnes, paprastesnes naudoti. Dabartinėmis ASR paremtos programos leidžia žmonėms, turintiems klausos sutrikimų, naudotis telefoniniu ryšiu, paverčiant pašnekovo ar pranešėjo šneką tekstu.

Kalbos atpažinimo sistemas galima skirti į dvi pagrindines grupes:

1. **Nuo kalbančiojo priklausomos** (angl. *speaker dependent*) sistemos. Tokios sistemos turi būti mokytos vartotojo, kuris ta sistema naudosis, balsu. Dažniausiai tai atliekama vartotojui skaitant iš anksto paruoštą tekstą (pvz., komandų sąrašą) arba žodyną į mikrofoną sistemai mokyti. Tokios sistemos yra ribotų galimybių ir nepatogios, tačiau leidžia prisitaikyti prie vartotojo akcento, tarties ir kalbėjimo tempo.
2. **Nuo kalbančiojo nepriklausomos** (angl. *speaker independent*) – tai sistemos, kurių nereikia mokyti žmogaus, kuris naudosis ta sistema, balsu. Tokios sistemos yra iš anksto paruoštos ir yra daug praktiškesnės ir patogesnės vartotojui, tačiau tikslumu dažnai negali prilygti personalizuotoms sistemoms.

Sistemas taip pat galima skirstyti pagal apdorojamos kalbos pobūdį:

1. **Izoliuotų (atskirų) žodžių** (angl. *isolated speech*) sistemos geba atpažinti atskirus žodžius, tačiau netinkamos rišliai kalbai atpažinti. Izoliuotų žodžių atpažinimo uždavinys yra paprastesnis: lengviau rasti žodžio pradžią ir pabaigą; žodžių tartys neturi įtakos kitiems žodžiams. Tokios kalbos atpažinimo sistemos tinka šneka valdomoms sistemoms kurti.
2. **Rišlios, nuoseklios kalbos** (angl. *continuous speech*) sistemos yra sudėtingesnės, tačiau geriau veikia praktikoje. Pirmiausia, tokioms sistemoms sunku nustatyti žodžio pradžią ir pabaigą, kadangi nėra aiškių pauzių tarp žodžių. Kalbėtojo tariami žodžiai taip pat turi įtakos kitiems žodžiams, pvz.: kintantis kalbėjimo greitis daro uždavinį sudėtingesnį.

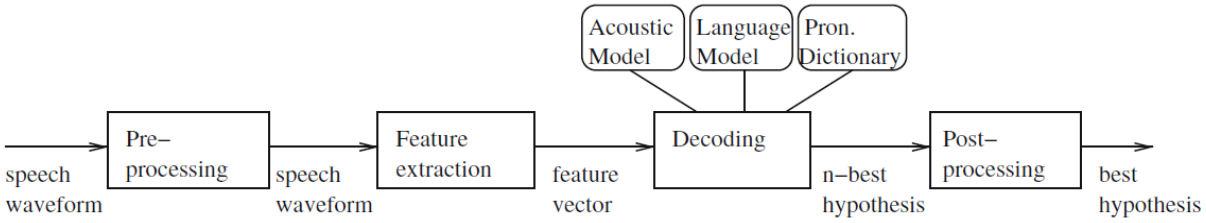
Nors šios automatinio kalbos atpažinimo sistemos skiriasi savo galimybėmis ir taikymu, jų veikimo principas yra labai panašus.

1.2 Tradicinis automatinio kalbos atpažinimo sistemos modelis

Žmogaus kalbos apdorojimas yra sudėtingas uždavinys, kadangi skirtingi žmonės turi skirtingus akcentus, tartis, kirčius, intonacijas ir kalbėjimo greičius; kalbai taip pat įtakos turi žmogaus amžius ar lytis. Be to, žmogaus skleidžiamas akustinis signalas taip pat yra veikiamas aplinkinių triukšmų, mikrofono ir kambario savybių, apsunkinančių apdorojimą.

Iki šiol kalbos atpažinimo metodai rėmėsi statistiniais modeliais. Tokie modeliai veikia darydami prielaidą, kad akustinis signalas atitinka žodį (ar žodžių seką) sistemos žinomame žodyne su tam tikra tikimybe. Tikimybės yra apskaičiuojamos remiantis akustinio signalo savybėmis ir lingvistinėmis žiniomis apie kalbą. Sistema įvertina visas hipotezes ir parenka labiausiai tikėtiną, nurodančią žodį ar jų seką.

Apibendrinta automatinio kalbos atpažinimo sistemos struktūra pateikta 1.1 pav..



1.1 pav. Automatinės kalbos atpažinimo struktūros pavyzdys [1]

Nors yra įvairių kalbos atpažinimo sistemų struktūrų, galima skirti penkis pagrindinius sistemos komponentus:

1. požymių išgavimo dalį (angl. *feature extraction*);
2. akustinį modelį (angl. *acoustic model*);
3. tarties modelį (angl. *pronunciation model*);
4. kalbos modelį (angl. *language model*);
5. paieškos dalį (angl. *decoding*).

Tolimesniuose poskyriuose išvardintos dalys yra apžvelgiamos plačiau.

1.2.1 Požymių išgavimas

Požymių išgavimo dalis yra atsakinga už požymių vektorių išgavimą iš akustinio signalo. Nors įmanoma atpažinti žodžius ar frazes iš neapdoroto akustinio signalo, tai atlikti yra sudėtinga, nes signale be lingvistinės informacijos yra papildomų duomenų, pvz.: mikrofono ir aplinkos triukšmų. Požymių vektorių paskirtis yra glaustu formatu išreikšti signale esantį gausų fonetinį turinį.

Požymių išgavimo metodai veikia priskirdami akustinius signalus lėtai kintančių signalų grupei. Signalo apdorojimo metu požymių vektoriai yra sudaromi dalijant akustinį signalą į mažesnius, 10, 15 ar 20 milisekundžių kadrus, kadangi nagrinėjant pakankamai mažus signalo intervalus galima laikyti, jog signalo savybės tame intervale nekinta ir signalas yra stacionarus.

Požymių išgavimo algoritmų pagalba kiekvienas kadras yra apibrėžiamas kaip požymių vektorius, kuris nusako akustinio signalo požymius ir fonetinį turinį tame kadre. Egzistuoja daugybė akustinio signalo apdorojimo algoritmų, kurie skiriasi veikimo metodais ir parenkamais signalo požymiais. Populiariausi metodai yra apdoroti signalą melų filtrais sudarant melų dažnių skalės kepstrinius koeficientus (angl. *Mel-frequency cepstral coefficients*, toliau MFCC) arba išgauti percepcinio tiesinio prognozavimo koeficientus (angl. *Perceptual linear predictive coeffiecients,* toliau PLPC).

1.2.2 Akustiniai modeliai

Akustinis modelis gautus požymių vektorius paverčia baziniais garsinės kalbos vienetais: skiemenimis, morfemomis arba dažniausiai – fonemomis. Fonemos – tai kalbos garsinės sistemos mažiausi vienetai, skiriantys tos kalbos žodžius ar morfemas. Skirtingos kalbos turi skirtingą skaičių fonemų: anglų kalboje skiriama apie 40, lietuvių – 59 fonemos. Šnekos modeliavimui gali būti naudojamos ne tik paprastos fonemos, bet ir jų junginiai – bifoniai, trifoniai[2].

Akustiniai modeliai sudaromi naudojant statistinius modelius, tokius kaip paslėpti Markovo modeliai (angl. *Hidden Markov models,* toliau HMM); Gauso mišinių modeliai (angl. *Gaussian mixture models*, toliau GMM), dirbtiniai neuroniniai tinklai (angl. *Artificial Neural Networks,* toliau ANN); rekurentiniai neuroniniai tinklai (angl. *Recurrent Neural Networks,* toliau RNN) ir pan.

1.2.3 Tarties modelis

Tarties modelio paskirtis – nustatyti tariamą žodį panaudojant iš akustinio modelio gautą informaciją. Tai atliekama naudojant papildomą duomenų bazę, kuri nurodo, iš kokių fonemų kiekvienas žodis yra sudarytas. Tarties modeliuose kiekvienas žodis gali turėti net keliolika skirtingų tarčių (fonemų sekų). Lyginant su dideliais tarties modeliais, maži tarties modeliai, kuriuose dauguma žodžių yra akustiškai skirtingi, leidžia pasiekti daug geresnius klasifikavimo rezultatus, tačiau smarkiai apriboja sistemos apdorojamų žodžių skaičių.

1.2.4 Kalbos modelis

Pagrindinė kalbos modelio paskirtis – apriboti alternatyvių žodžių hipotezių paieškos erdvę, t. y., nustatyti, koks žodis yra sakomas naudojat žinias apie kalbą ir informaciją apie prieš tai ištartus žodžius. Naudojant šį modelį sistema gali atskirti panašius akustinius signalus, o taip pat ištaisyti kai kurias klaidas padarytas akustinio modelio.

Kalbos modelio dalis, naudodama informaciją apie prieš tai apdorotų žodžių seką, pateikia tikimybę, jog tokia žodžių seka egzistuoja atpažįstamoje kalboje. Praktikoje dažnai naudojami *n*-gramų modeliai, pateikiantys statistinę tikimybę, kad seka su *n* iš eilės einančių žodžių egzistuoja kalboje [2]. Nors kalbos modelio panaudojimas sistemoje nėra būtinas, jis gali stipriai pagerinti automatinio kalbos atpažinimo sistemos veikimą.

1.2.5 Ištartų žodžių nustatymas

Dekodavimo metu sujungiamos akustinės žinios, gautos iš akustinio modelio, bei lingvistinė informacija, gauta iš kalbos modelio. Galimų žodžių sekų skaičius gali būti labai didelis, todėl dekodavimo

posistemės darbas yra pasinaudojant kitų posistemių teikiama informacija sumažinti galimų sprendimų erdvę ir parinkti labiausią tikėtiną ištartų žodžių seką $\widehat{W}$:

$$\widehat{W} = \underset{W}{\operatorname{argmax}}(P(W)P(Y|W)), \tag{1}$$

čia $P(W)$ tikimybė, kad žodžių seka egzistuoja vartojamoje kalboje, $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2 \dots, \boldsymbol{y}_{T'})$ – akustinio signalo savybių vektorių seka, $P(Y|W)$ – tikimybė nusakanti, kaip tikėtina, kad akustinio signalo požymių vektoriaus seka yra gauta iš tam tikros žodžių sekos W. Dekodavimo metu tikimybę $P(W)$ pateikia kalbos, o $P(Y|W)$ – akustinis modeliai.

1.3 Šnekos atpažinimas nuo signalo vieneto pradžios iki galo

Klasikinės automatinio kalbos atpažinimo sistemos, aptartos ankstesniuose poskyriuose, pasižymi sudėtinga struktūra, o jų pritaikymas kitoms kalboms yra daug inžinerinių pastangų reikalaujantis procesas. Kaip alternatyvą tokio tipo sistemoms mokslininkai ėmė kurti modelius, kurie sudaryti iš dirbtinių neuroninių tinklų ir gali atlikti automatinį šnekos atpažinimą iš neapdorotų (arba minimaliai apdorotų) garso įrašų, kitaip dar vadinamus šnekos atpažinimo nuo signalo vieneto pradžios iki galo sistemomis (angl. *end-to-end speech recognition systems*) [3, 4].

Sistemos, galinčios atlikti šnekos atpažinimą nuo signalo vieneto pradžios iki galo, yra paremtos gilaus mokymosi metodais ir pakeičia klasikinėse automatinio šnekos atpažinimo sistemose esančius modelius vienu modeliu. Tokie modeliai taip pat reikalauja daug mažiau inžinerinių pastangų, pvz.: ekspertų surinktų žinių suvedimo rankiniu būdu, o mokymosi procesas yra daug paprastesnis. Lyginant su klasikinėmis sistemomis, šios sistemos pasižymi paprastesne ir elegantiškesne architektūra ir gali būti nesunkiai pritaikyti skirtingoms kalboms (pvz., tos pačios architektūros modelis gali būti išmokytas anglų, vokiečių, prancūzų, čekų ir kt. kalboms).

Tokio tipo sistemų vystymas yra aktyvus mokslinių tyrimų subjektas, o dabartinės sistemos geba aplenkti ir ilgus metus tobulintas klasikines sistemas [5].

1.4 Lietuvių kalbos atpažinimas

Nors automatinio lietuvių kalbos atpažinimo sritis nėra nauja, ASR taikymų ir su šia sritimi susijusių tyrimų lietuvių kalbos apdorojimui nėra daug. Pagrindinės priežastys – lietuvių kalbos sudėtingumas ir lingvistinių duomenų trūkumas. Lietuvių kalba pasižymi sudėtinga tartimi, lanksčia žodžių tvarka ir gausiu žodynu, o rašybai naudojama papildyta lotynų abėcėlė. Kadangi lietuviškai kalba tik apie 3,5 mln. žmonių, sudėtinga gauti pakankamai duomenų kalbos ir šnekos modeliavimui.

Atlikti esamų sistemų, atliktų sprendimų analizę ir palyginimą nėra paprasta – lietuvių šnekos atpažinimo sistemos skiriasi ne tik mokymui naudojamų duomenų rinkiniais (fonemų formatais, įrašų kiekiais, jų turiniais ir pan.), bet ir naudojamomis atpažinimo metodikomis bei tikslais (kai kurios sistemos apsiriboja fonemų atpažinimu akustiniuose signaluose, kitos geba dirbti atpažinti ir atskirus žodžius). Tolimesniuose poskyriuose trumpai apžvelgiami su automatiniu lietuvių kalbos atpažinimu susiję tyrimai ir realizuotos sistemos.

1.4.1 „Google“ teikiamos ASR paslaugos lietuvių kalbai

Ne paslaptis, kad „Google“ yra viena iš stipriausių ir labiausiai kalbos atpažinimo technologijas vystančių kompanijų. Kompanija skelbia, kad jos ASR sistema atpažįsta daugiau nei 110 kalbų, yra atspari aplinkos triukšmui, gali apdoroti tiek įrašytą, tiek realiu laiku pateikiamą šneką ir veikia naudodama giliuosius neuroninius tinklus. 2016 m. Kauno technologijos universiteto mokslininkų pastangomis buvo ištirta „Google“ siūloma kalbos atpažinimo paslauga, siekiant nustatyti tokios sistemos efektyvumą ir tikslumą lietuvių šnekos atpažinimui [6]. Eksperimentams buvo panaudoti 63299 kalbos įrašai, kuriuose įrašytos 3318 skirtingos frazės. Atlikti eksperimentai parodė, kad atpažintuvas atpažįsta tik apie 67 proc. visų pateiktų įrašų, o atpažinti įrašai pasižymi tik 59.26 proc. tikslumu.

1.4.2 Projektas LIEPA

LIEPA – 2013-2015 m. Vilniaus universiteto ir partnerių vykdytas projektas, kurio metu sukurtos 6 lietuvių šneka valdomos internetinės paslaugos, skatinančios ir ugdančios lietuviškumą ir tautiškumą elektroninėje erdvėje [7]. Visų vartotojams siūlomų paslaugų pagrindą sudaro projekto metu sukurtos ir išvystytos infrastruktūrinės paslaugos: elektroninio teksto skaitytuvas, interneto puslapių atidarymo balsu programinė įranga, lietuvių šnekos garsynas ir lietuvių šnekos atpažinimo variklis. Atpažintuvas sukurtas naudojant 113 val. anotuotą Lietuvių šnekos garsyną LIEPA. Garsyną sudaro 100 val. įrašų šnekos atpažinimo paslaugoms ir 13 val. šnekos sintezavimui. Projekto metu sukurtos dvi atpažintuvo versijos:

- Frazių atpažintuvas, gebantis atpažinti iš anksto apibrėžtas ir fiksuotas žodžių sekas. Šis atpažintuvas geba atpažinti daugiau nei 100 lietuviškų frazių ne mažesniu nei 95 proc. tikslumu. Tikimasi, kad atpažintuvas gali būti panaudotas komandinėms, šneka valdomoms sistemoms kurti ir įrenginiams valdyti.
- Laisvų žodžių sekų atpažintuvas, gebantis atpažinti laisvas žodžių sekas, sudarytas iš daugiau nei 300 lietuviškų žodžių.

Kalbos atpažintuvas sukurtas naudojant JAV Carnegie Mellon universiteto „CMU Sphinx“ atviro kodo kalbos atpažinimo variklį. „ CMU Sphinx“ yra nuo vartotojo nepriklausoma kalbos atpažinimo sistema, naudojama ištisinės kalbos atpažinimui.

1.4.3 Izoliuotų žodžių atpažinimas

Šio amžiaus pradžioje Lipeika ir kt. [8] pasiūlė izoliuotų lietuviškų žodžių atpažintuvą, kurio pagrindą sudarė dinaminis laiko skalės kraipymo algoritmas (angl. *Dynamic Time Warping*), leidusi palyginti akustinius signalus su pavyzdiniais akustinių signalų šablonais, kuriuose fonemų lygiavimas jau žinomas, taip atliekant fonemų sekų išgavimą iš akustinio signalo. Atpažintuvo kūrimui ir įvertinimui buvo panaudotas garsynas, sudarytas naudojant 12 žodžių garsyną, padiktuotą 10 diktorių, su 120 išsireiškimų per diktoriu (viso 1200 ištartų žodžių). Nors tokio tipo sistema gali užtikrinti aukštus rezultatus mažo žodyno sistemoms, ją sunku pritaikyti dideliems žodynams, kadangi tinkamo šablono paieška dideliame pavyzdinių akustinių signalų rinkinyje reikalauja nemažų skaičiavimo resursų.

Raškinis ir Raškinienė [9] 2002 m. pristatė HMM paremta lietuvių šnekos atpažinimo sistemą izoliuotų žodžių atpažinimui. Sistemos kūrimui autoriai naudojo MFCC koeficientų išgavimo metodą, HMM akustinio modeliui kūrimui, o tarties modelį sudarė patys, panaudodami žodžius iš turimo garsyno. Atpažintuvo apmokymui panaudotas 60,6 minučių garsynas, kurį sudarė 1100 išsireiškimų padiktuoti 4

skirtingų diktorių. Garsyno įrašai, kuriuos sudarė fonemų sekos buvo suderintos laiko atžvilgiu su atitinkamomis transkripcijomis. Izoliuotų žodžių atpažinimo uždavinyje sistema sugebėjo pademonstruoti mažesnį nei 20 % žodžių klaidos rezultatą, o ištisų išsireiškimų atpažinimo uždavinyje – mažesnį nei 30 %. Didžiausias šios sistemos trūkumas – daug inžinierinių pastangų ir ekspertų žinių reikalaujantis duomenų rinkinio ir tarties modelio sudarymo procesas, kurį sunku pritaikyti didelio žodyno sistemoms.

Hibridinis modelio, sudaryto iš HMM ir ANN, panaudojimo lietuvių šnekos atpažinimui galimybių analizė pateikta 2004 m. Filipovič ir Lipeikos [10]. Autoriai naudojo Raškinio ir Raškinienės [9] naudotą garsyną ir sugebėjo pasiekti 86,7 proc. kalbos atpažinimo rezultatų tikslumą.

1.4.4 Transliuojamos lietuvių kalbos atpažinimas

Žiniasklaidoje vartojama kalba pasižymi aiškia ir taisyklinga tartimi, neturi nelietuviškų žodžių, o kalbos atpažinimo pritaikymo galimybės yra labai plačios (transkripcija, titravimas). Tokios kalbos duomenų (garso įrašų) taip pat nėra sudėtinga gauti elektroninėje erdvėje. Todėl nenuostabu, kad transliuojamų duomenų apdorojimas kalbos atpažinimo priemonėmis sulaukė mokslininkų dėmesio.

Tyrimai, susiję su transliuojama lietuvių kalba, pradėti dar 2004 m. Vilniaus universitete [11]. Laurinčiukaitė ir Lipeika [12] 2005 m. pristatė fonemomis ir skiemenimis paremta lietuvių šnekos atpažintuvą. Autorių sukurtas skiemenimis ir atpažintuvas sugebėjo pasiekti 56,67 proc. atpažinimo tikslumą naudojant 10 val. Lietuvos Radijo garsyną. Vėliau, 2006 m., atlikti eksperimentai siekiant palyginti kalbos atpažinimo tikslumą atpažinimo procese naudojant fonemas, skiemenis ir žodžius [13]. Sistema, sudaryta naudojant HMM ir apmokyta 10 val. trukmės garso įrašų rinkiniu, sugebėjo pasiekti 58,06 proc. atpažinimo tikslumą. Po dešimties metų, 2016 m., LIMSI institute atlikti eksperimentai, kuriais buvo bandoma sukurti lietuvių šnekos atpažintuvą transliuojamų duomenų transkripcijos sistemai [14]. Atpažintuvo kūrimui buvo panaudoti neanotuoti garso įrašai, kurių bendrą trukmė – apie 360 valandų. Atpažintuvas kurtas naudojant HMM ir GMM ir apmokytas naudojant mokymosi be mokytojo tipą. Siekiant pagerinti atpažinimo tikslumą, sistema buvo tobulinama įvairiais būdais: papildomai apmokant mažu, anotuotu 3 val. trukmės garso įrašų rinkiniu; keičiant GMM dirbtiniais neuroniniais tinklais; fonemų žodyną pakeičiant grafeminiu ir pan. Galutinė sistemos versija sugebėjo pasiekti 80,7 proc. tikslumą, kuris buvo išmatuotas naudojant rankiniu būdu anotuotus garso įrašus, kurių bendra trukmė – 3 val.

Lietuvių kalbos atpažinimu taip susidomėjo Talino technologijos universiteto mokslininkai – 2016 m. jie pritaikė estų kalbai skirtą ASR sistemą transliuojamos lietuvių kalbos atpažinimui [15]. Sistema, apmokyta 84 val. trukmės anotuotu garso įrašų rinkiniu ir daugiau nei 400 mln. žodžių su tekstine informacija, pasiekė net 85,3 proc. atpažinimo tikslumą. Atpažintuvas moka atlikti šnekos segmentavimą, kalbėtojo identifikavimą, šnekos transkripciją bei automatinį skyrybos grąžinimą ir yra naudojamas komerciniais tikslais.

1.4.5 Telefoninių pokalbių apdorojimas

Palyginus žiniasklaidoje ir telefoniniuose pokalbiuose vartojamas kalbas, nesunku pastebėti, kad pastaroji yra spontaniškesnė: varijuoja kalbančiųjų tempas, ne visados sekamos gramatikos taisyklės, kokybę lemia mikrofono ir aplinkos triukšmas. Telefoninės kalbos apdorojimas nėra lengvas, tačiau labai svarbus uždavinys – automatinė kalbos atpažinimo įranga gali būti panaudota ne tik komerciniais, medicininiais, bet ir teisėtvarkos tikslais.

Problemos sudėtingumas ir didelės pritaikymo galimybės lėmė mokslininkų susidomėjimą – 2015 m. IARPA-Babel programos metu sukurtas telefoninių pokalbių lietuvių šneka atpažintuvas, gebantis pasiekti 57,6 proc. tikslumą [16]. Tais pačiais metais grupė Jungtinės Karalystės mokslininkų įtraukė lietuvių kalbą į tyrimus, susijusius su mažai resursų turinčiomis kalbomis [17]. Tyrimams buvo panaudoti anotuoti telefoninių pokalbių įrašai, kuriais apmokyta šnekos atpažinimo sistema sugebėjo pasiekti 71,3 proc. tikslumą.

1.4.6 Kitų kalbų modelių pritaikymas

Pasiūlymas pritaikyti kitoms kalboms skirtus modelius lietuvių kalbos atpažinimui kilo pastebėjus, kad daug kalbėtojų turinčių kalbų modeliai ar jų dalys gali būti panaudoti mažiau paplitusių kalbų modeliavimui, taip sumažinant su kalbos atpažinimo sistemų vystymu susijusius kaštus. Idėja remiasi tuo, kad daugelis kalbų nėra izoliuotos ir buvo (tebėra) veikiamos kitų kalbų, todėl turi panašių, kartais net bendrų akustinių ir fonetinių savybių, kurias galima išnaudoti kuriant kitų kalbų atpažinimo sistemas.

Dar 2007 m. pradėti tyrimai siekiant įvertinti užsienio kalboms skirtų ASR priemonių pritaikymo galimybes lietuvių kalbos automatiniam atpažinimui [18]. Tyrimu metu palygintos dviejų užsienio kompanijų siūlomos anglų kalbos ASR sistemos, skirtos naudoti skambučių priėmimo centruose. Nustatyta, jog užsienio kalbų modelius galima naudoti lietuvių šnekos atpažinimui lietuvių kalbos žodžius transkribuojant anglų kalba, tačiau tai negarantuoja tikslių rezultatų.

2008 m. nustatytas modifikuoto anglų kalbai skirto atpažintuvo efektyvumas lietuvių šnekos uždaviniui spręsti [19]. Garsyną, kurį sudarė 10 skirtingų žodžių, sistema sugebėjo suklasifikuoti 99 proc. tikslumu, o didesnį, 500 žodžių garsyną, tik 84,2 proc. tikslumu. Prieita prie išvados, kad tokio tipo sistema tinka tik nedidelio žodyno uždaviniams spręsti.

Metais vėliau, naudojant lietuvių kalba tariamų skaičių garsyną, atlikti eksperimentai su keliomis skirtingomis ASR sistemomis (anglų, ispanų, vokiečių ir prancūzų) parodė, kad lietuvių šnekos atpažinimo uždaviniui spręsti labiausiai tinkama ispanų šnekai pritaikyta ASR sistema [20]–[22]. Remiantis šiais rezultatais, 2013 m. atlikti tyrimai siekiant palyginti ispanų ir lietuvių kalbai skirtų ASR sistemų efektyvumą atpažįstant lietuvių kalbą [23]. Eksperimentai parodė, kad užsienio kalbos modelis nusileidžia lietuvių kalbos modeliui, tačiau gali būti naudojamas kartu su lietuviškuoju hibridinėje sistemoje, taip padidinant atpažinimo tikslumą.

1.4.7 Kiti tyrimai

2016 m. atlikti tyrimai siekiant sukurti kasdieniniuose pokalbiuose naudojamai lietuvių šnekai atpažinti skirtą ASR sistemą [24]. Tokios kalbos atpažinimas nėra lengvas uždavinys – kasdieniuose pokalbiuose naudojama kalba pasižymi laisvumu: į pokalbius dažnai įterpiama svetimybių, barbarizmų, žargonizmų, kinta kalbėjimo greitis, tartis ir pan. Kalbos atpažintuvas kurtas naudojant HMM ir apmokytas naudojant 97 min. garso įrašų rinkinį, tačiau eksperimentai parodė, kad gero tikslumo jis pasiekti negali.

Ne visi tyrimai automatinio lietuvių kalbos atpažinimo srityje buvo atlikti siekiant sukurti pilną kalbos atpažinimo sistemą. Kai kurie tyrėjai nusprendė susikoncentruoti į pagrindinių tokios sistemos dalių tobulinimą. Pavyzdžiui, 1999 m. buvo atliktas tyrimas siekiant rasti tinkamiausius požymių vektoriaus sudarymo ir jų klasifikavimo būdus fonemų diskriminavimui [25]. Driaunys ir kt. pasiūlė fonemų sudarymo šabloną iš stacionarių ir pereinamųjų fonemų dalių, kuriuo galima akustinį signalą apibrėžti vienu požymių vektoriumi [26].

Siekiant paspartinti automatinio šnekos atpažinimo priemonių, skirtų lietuvių kalbai, vystymą, buvo sudarytas lietuviškų skaitmenų ir kompiuterio valdymo komandų garsynas LTDIGITS [27], kuriame yra aiškiai apibrėžtos ištartus žodžius sudarančių fonemų ribos. Renkant garsyną buvo surinkta nemažai priebalsio-balsio tipo junginių, tarp kurių ir sudėtingesnės nosinių priebalsių ir įvairių balsių poros (/na/, /ma/ ir pan.).

Statistiniai kalbos modeliai yra viena iš svarbiausių didelio žodyno ASR sistemų dalių, leidžiančių pagerinti jų veikimą. Didelis lietuvių kalbos žodynas ir kalbos struktūra, leidžianti išreikšti ryšius tarp žodžių sakinyje kaitant galūnes, yra vienos didžiausių kliūčių efektyvaus statistinio kalbos modelio kūrimui. Vaičiūnas ir kt. [28] pasiūlė klasėmis paremtą *n*-gramų kalbos modelį. Pasak tyrėjų, tokio tipo kalbos modeliai veikia geriau nei įprasti *n*-gramų kalbos modeliai.

1.5 Fonemomis remtos ASR sistemos

Viena iš sričių, susilaukusi mažesnio mokslininkų dėmesio yra automatinis kalbos atpažinimas iš jau turimų fonemų sekų. Iš pirmo žvilgsnio paprastas, uždavinys nėra lengvai išsprendžiamas, kadangi ta pati fonema (ar jų grupė) gali atitikti skirtingas grafemas, priklausomai nuo jų konteksto. Ilgas fonemų sekas, kurias sudaro daugiau nei vieno žodžio fonemos, statistiniams modeliais apdoroti yra ypač sunku, kadangi pauzės tarp žodžių ne visad yra daromos ir priklauso nuo šnekančiojo kalbėjimo greičio.

Klasikinėse automatinio kalbos atpažinimo sistemose toks fonemų sekų atpažinimas vykdomas naudojant tarties modelį (žodyną), kuris leidžia atpažinti žodžius pagal jų fonemų sekas, o gauti žodžiai yra patikslinami kalbos modelio pagalba dekodavimo žingsnio metu. Tarties ir kalbos modelių sudarymas yra sudėtingas ir ilgas procesas, ypač morfologiškai turtingoms kalboms, o gauti modeliai negarantuoja gerų ASR sistemos rezultatų. Šnekos atpažinimo sistemose, atliekančiose atpažinimą nuo signalo vieneto pradžios iki galo, fonemų sąvoka net neegzistuoja, kadangi jų reprezentacijos modelyje yra latentinio tipo.

Automatinio šnekos atpažinimo sistemos, paremtos fonemų sekų apdorojimų, būtų paprastesnės nei klasikinės kalbos atpažinimo sistemos, ir pasižymėtu lankstumu, kadangi leistų panaudoti jau egzistuojančius akustinius modelius. Manoma, kad mokymasis naudojant fonemų sekas turėtų leisti automatinio kalbos atpažinimo sistemai įsisavinti ryšį ne tik tarp grafemų ir jas atitinkančių fonemų ar jų grupių, bet ir išmokti atskirti atskirus žodžius, t. y. „susidaryti“ kalbos modelį. Glaudus ryšys tarp fonetinių ir morfologinių žodžių formų lietuvių kalboje turėtų leisti ASR sistemai pasiekti aukštus šnekos atpažinimo rezultatus.

2 PĮ projektavimas

2.1 Reikalavimai programinei įrangai

Šiame skyriuje pateikiami funkciniai ir nefunkciniai reikalavimai kuriamai programinei įrangai.

2.1.1 Funkciniai reikalavimai

Funkciniai reikalavimai kuriamai sistemai:

1. Sistema turi atlikti lietuvių kalbos žodžių klasifikaciją, kai žodžiai sistemai pateikiami tinkamu formatu ir yra suskaidyti fonemomis.
2. Numatyti galimybę keisti atpažinimui naudojamo modelio parametrus.
3. Turi būti galimybė pakartotinai apmokyti klasifikatorių.
4. Turi būti galimybė ateityje išplėsti sistemą į pilną automatinio kalbos atpažinimo sistemą.
5. Lietuvių kalbos atpažinimo sistema turi pateikti klasifikavimo tikslumo įvertinimą pagal iš anksto apibrėžtas tikslumo įvertinimo metrikas.

2.1.2 Nefunkciniai reikalavimai

Nefunkciniai reikalavimai kuriamai sistemai:

1. Kuriama sistema turi būti paremta giliuoju neuroniniu tinklu.
2. Duomenys (garsinė informacija, kurią sudaro atskiri žodžiai arba jų sekos) sistemai pateikiami fonemų pavidalu, kurių formatas žinomas iš anksto.
3. Kuriama PĮ įranga turi pateikti atpažintus žodžius grafemų sekomis.
4. Turi būti galimybė sistemą pritaikyti izoliuotų žodžių ir rišlios šnekos atpažinimui.
5. Kuriama sistema turi veikti stabiliai ir netrikdyti trečiųjų šalių PĮ ir operacinės sistemos darbo.

2.2 Kokybės įvertinimas

2.2.1 Kokybės kriterijai

Pagrindinis kuriamos sistemos kokybės kriterijus – atpažinimo tikslumas. Yra daugybė skirtingų atpažinimo tikslumą aprašančių metrikų, naudojamų mašininio mokymosi metodais paremtose ASR sistemose. Plačiausiai naudojamos žemiau išvardintos trys:

1. Tikslumas (angl. *Accuracy*).
2. Levenšteino atstumas (angl. *Levenshtein distance*) – simbolių eilučių panašumo matavimo metrika.
3. Žodžių atpažinimo klaidos rodiklis (angl. *Word Error Rate*, toliau WER).

Tikslumas – viena dažniausiai randamų mašininio mokymosi algoritmų efektyvumo matavimo metrikų. Tikslumas ACC yra matuojamas naudojant formulę:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \quad (2)$$

čia TP – teisingų teigiamų (angl. *True Positive*) rezultatų kiekis, TN – teisingų neigiamų (angl. *True Negative*) rezultatų kiekis, FP – klaidingų teigiamų (angl. *False Positive*) rezultatų kiekis, FN – klaidingų neigiamų (angl. *False Negative*) rezultatų kiekis.

Levenšteino atstumas yra naudojamas nustatant panašumą tarp dviejų simbolių eilučių (pvz., žodžių arba sakinių). Atstumas išreiškia minimalų atskirų simbolių redagavimų kiekį (simbolių įterpimų, ištrynimų ir pakeitimų), kuriuos reiktų atlikti, norint, kad dvi simbolių eilutės sutaptų. Matematiškai Levenšteino atstumas tarp dviejų simbolių eilučių a ir b gali būti aprašytas naudojant formulę:

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{jei } \min(i,j) = 0 \\ \min\begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(\mathrm{a}_i \neq b_j)} \end{cases} & \text{kitais atvejais,} \end{cases} \quad (3)$$

čia $\mathrm{lev}_{a,b}(i,j)$ – redagavimo atstumas tarp pirmųjų i a eilutės simbolių ir pirmųjų j b eilutės simbolių; $1_{(\mathrm{a}_i \neq b_j)}$ – yra funkcija įgyjanti 0 reikšmę, kai $\mathrm{a}_i = b_j$, ir 1, kai $\mathrm{a}_i \neq b_j$.

Levenšteino atstumas pasižymi tuo, kad gali dirbti su skirtingo ilgio simbolių eilutėmis, t. y. eilutės neprivalo tenkinti sąlygos $|a| = |b|$, kur $|a|$ yra a eilutę sudarančių simbolių, o $|b|$ yra b eilutę sudarančių simbolių kiekiai.

Žodžių atpažinimo klaidos rodiklis yra dažnai naudojamas ASR, mašininio vertimo ir pan. sistemose jų efektyvumui įvertinti. WER metrika skaičiuoja neteisingai sistemos atpažintų, ištrintų ar įterptų žodžių skaičių lygindama sistemos rezultatus su originalia fraze. WER yra apskaičiuojama pagal formulę (4), kur S – skaičius žodžių, kuriuos ASR atpažino neteisingai; I – sistemos įterpti žodžiai, kurių nebuvo originaliame garso įraše, kiekis; D – ASR sistemos pašalintų žodžių, kurie buvo originaliame garso įraše, kiekis; N – bendras ištartų žodžių kiekis.

$$WER = \frac{S + I + D}{N} \quad (4)$$

Pasikliautinasis intervalas (angl. *confidence interval*, toliau PI) yra dažnai naudojamas statistinis dydis, kuriuo galima įvertinti gautų taškinių įverčių (pvz.: tikslumo) patikimumą. Kai generalinės aibės standartinio nuokrypio reikšmė nėra žinoma, pasikliautinasis intervalas gali būti apskaičiuotas naudojant formulę (5), kur μ yra imties vidurkis, σ – imties standartinis nuokrypis, N – imties dydis, Q – pasikliovimo lygmuo, o koeficientas $t_{(1-(1-Q)/2)}(N-1)$ yra Stjudento skirstinio su $N-1$ laisvės laipsnių $1-(1-Q)/2$ lygmens kvantilis. Moksliniuose tyrimuose įprasta naudoti 0,95 ir 0,99 pasikliovimo lygmens reikšmes.

$$\left(\mu - t_{\left(\frac{1-Q}{2}\right)}(N-1)\frac{\sigma}{\sqrt{N}}, \mu + t_{\left(\frac{1-Q}{2}\right)}(N-1)\frac{\sigma}{\sqrt{N}}\right) \quad (5)$$

2.2.2 Kryžminė patikra

Įprastai statistiniai modelių apmokymui ir testavimui yra sudaromos 3 duomenų imtys:

1. Apmokymo duomenų imtis, kuria apmokomas modelis.

2. Patikros duomenų imtis, kurios pagalba ieškoma hiperparametrų (pvz., dirbtinių neuronų kiekio DNT), su kuriais modelis pasiekia didžiausia tikslumą matuojama kokybės metrikomis.
3. Testavimo duomenų imtis skirta galutiniam modelio įvertinimui. Radus tinkamiausius hiperparametrus, modelis yra ištestuojamas testavimo duomenimis siekiant nustatyti modelio efektyvumą ir kaip gerai modelis

Kitas plačiai naudojamas metodas, kuriuo galima įvertinti modelio prognostines savybes – kryžminė patikra (angl. *cross-validation*). Šis modelio patikros metodas ne tik padeda nustatyti kaip gerai pasirinktas modelis veiktų praktikoje, bet ir patikrinti ar modelis nekenčia nuo per didelio prisitaikymo prie mokymosi duomenų imties. Kryžminė patikra taip pat leidžia nustatyti ar modelis nebuvo paveiktas šališkumo problemos, kuri gali atsirasti sudarant mokymosi, patikros ir testavimo duomenų rinkinius.

Vienas populiariausių kryžminės patikros būdų – *k*-lypė kryžminės patikra (angl. *k-fold cross-validation*), kurią galima aprašyti keturiais žingsniais:

1. Turima duomenų imtis padalinama į *k* dalių.
2. Modelis apmokomas *k*-1 dalių duomenimis, paliekant 1 dalį neliestą.
3. Apmokytas modelis įvertinamas likusia dalimi, gautos kokybės metrikos (tikslumas, WER ir t.t.) išsaugomos.
4. Kartojami 2-3 žingsniai, kiekvieno žingsnio metu parenkant vis kitą duomenų rinkinio dalį modelio patikrai.

Kryžminės patikros pabaigoje galima apskaičiuoti *k*-1 rezultatų, gautų įvertinus modelio tikslumą su kiekviena iš *k* dalių, aritmetinį vidurkį.

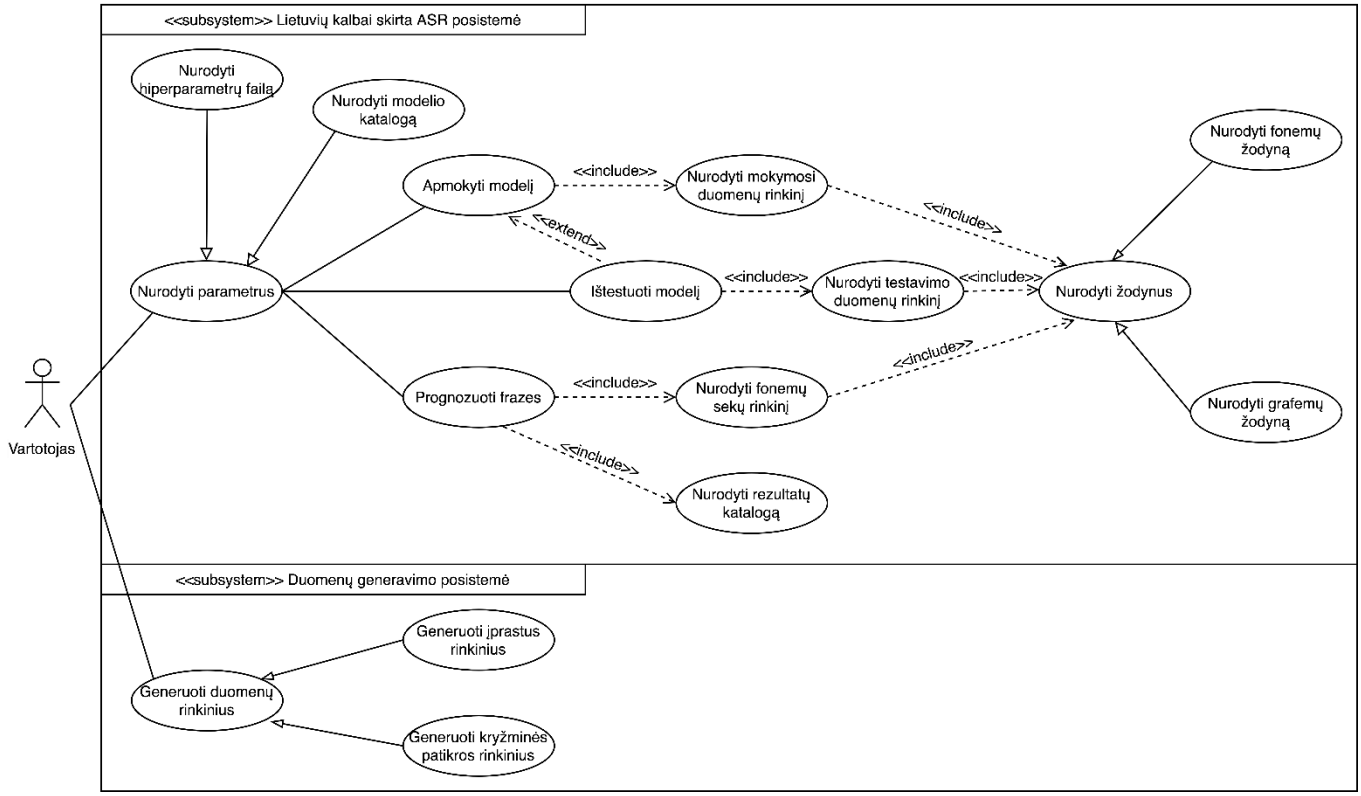
2.3 Panaudos atvejų diagrama

Lietuvių kalbai skirtos automatinio šnekos atpažinimo sistemos panaudos atvejų (angl. *use-case*, toliau PA) diagrama pateikta 2.1 pav.

Sistemą sudaro dvi posistemės: duomenų generavimo posistemė, kuri yra atsakinga už duomenų imčių generavimą iš pradinio garsyno ir lietuvių kalbai skirta ASR posistemė, kurios paskirtis išmokti atpažinti grafemų sekas iš jas atitinkančių fonemų seku. Sistemoje galima išskirti 5 pagrindinius panaudos atvejus:

1. „Nurodyti parametrus“ – sistemos vartotojas nurodo pagrindines ASR sistemos konfigūracijos dalis: katalogą, kuriame bus (arba jau yra) saugomas apmokytas modelis, ir hiperparametrų failą, kuriame nurodoma modelio konfigūracija (tipas, dirbtinių neuronų skaičius kiekviename sluoksnyje ir pan.).
2. „Generuoti duomenų rinkinius“ panaudos atvejis, kurio metu vartotojas sugeneruoja mokymosi, patikros ir testavimo duomenų rinkinius, arba duomenų rinkinius, kurie bus naudojami kryžminės patikros metu.
3. „Apmokyti modelį“. Šio panaudos atvejo metu yra sukuriamas naujas, pagal vartotojo pateiktus parametrus apibrėžtas modelis, kuris yra apmokomas nurodytu mokymosi duomenų rinkiniu.
4. „Ištestuoti modelį“ – modelio įvertinimui skirtas PA, kurio metu modelio kokybė yra įvertinama anksčiau aprašytomis metrikomis vartotojo nurodyto testavimo duomenų rinkinio pagalba.
5. „Prognozuoti frazes“ panaudos atvejis yra skirtas grafemų sekų generavimui iš pateiktų fonemų sekų.

Panaudos atvejai plačiau paaiškinti 7.1 skyriuje (Priedas A).



2.1 pav. Sistemos panaudos atvejų diagrama

2.4 Programinės įrangos platformų analizė

Kuriamą programinės įrangos pagrindą sudarys gilusis neuroninis tinklas, todėl realizacijai labai svarbu pasirinkti tinkamą programinę įrangą, kuri:

1. turėtų gerą palaikymą ir aktyvią bendruomenę;
2. išsamią, gerai sudarytą dokumentaciją;
3. būtų lengvai pritaikoma darbui su garsiniais duomenimis;
4. būtų skirta darbui su mašininio mokymosi algoritmais;
5. leistų paspartinti skaičiavimus grafinės vaizdo plokštės pagalba.

Šio darbo realizacijai buvo apsvarstyta keletas populiarių ir plačiai naudojamų mašininio mokymosi algoritmų kūrimui skirtų priemonių: TensorFlow, Kaldi, MATLAB ir CNTK.

TensorFlow yra atviro kodo biblioteka sukurta Google kompanijos 2015 m. ir skirta mašininio mokymosi algoritmų realizavimui [29]. Išsami dokumentacija, laisvai prieinamas ir atviras programinis kodas, didelis palaikymas iš kūrėjų, naudojimo paprastumas – visa tai lėmė tai, kad *TensorFlow* šiuo metu yra viena iš populiariausių mašininio mokymosi bibliotekų. *TensorFlow* palaiko daugybę standartinių mašininio mokymosi algoritmų ir modelių, pvz.: dirbtinius neuroninius tinklus (tarp jų ir sudėtingesnius konvoliucinius bei rekurentinius neuroninius tinklus), atramines vektorines mašinas (angl. *Suppor Vector Machines*), atsitiktinius miškus (angl. *Random Forests*). *TensorFlow* biblioteka gali būti naudojama per Python arba C++ programavimo kalbų sąsajas, nors kritiškai svarbios dalys bibliotekoje yra realizuotos C++ programavimo kalbą ir NVIDIA CUDA skaičiavimų aplinką, o tai leidžia perkelti daug resursų naudojančius skaičiavimus į grafinės vaizdo plokštės aplinką.

Kaldi – programinės įrangos įrankių rinkinys, sukurtas naudojant C++ programavimo kalbą ir skirtas automatinio kalbos atpažinimo sistemų kūrimui [30]. *Kaldi* projektas prasidėjo 2009 m. Johns Hopkins

universitete ieškant galimybių sukurti kalbos atpažinimo sistemas, kurios pasižymėtų aukšta kokybe ir mažais kūrimo kaštais. Šiuo metu *Kaldi* jau yra plačiai žinomas tarp automatinio kalbos atpažinimo tyrėjų, o jo vystymą remia daugelis gerai žinomų organizacijų. Rinkinys pasižymi daugelį tiesinės algebros metodų ir operacijų palaikančia biblioteka, algoritmų abstraktumu ir laisvai prieinamu kodu. Ši programinės įrangos platforma taip pat palaiko giliųjų, dirbtinių neuroninių tinklų modelius ir galimybę perkelti skaičiavimus į kompiuterio grafinės vaizdo plokštės aplinką. *Kaldi* įrankių rinkinio didžiausias trūkumas, kurį pripažįsta patys kūrėjai, yra išsamios dokumentacijos trūkumas, kuris atsirado dėl sudėtingo ir daug pastangų reikalavusio kūrimo proceso.

MATLAB – viena paprasčiausiai valdomų programų, skirta duomenų analizei ir algoritmų, programų bei matematinių modelių kūrimui [31]. *MATLAB* PĮ naudoja tokio pat pavadinimo programavimo kalbą ir palaiko daugelį operacinių sistemų bei aparatūros komponentų konfigūracijų. *MATLAB* leidžia PĮ įrangos kūrėjams naudoti didelį, specializuotų komponentų įrankių rinkinį, leidžiantį savo kuriamose programose panaudoti jau realizuotus, abstrakčius modelius ir algoritmus. *MATLAB* yra mokama programinė įranga, o norint naudoti giliuosius neuroninius tinklus reikia atskirai įsigyti darbui su ANN skirtus programinės įrangos paketus (pvz.: „Neural Network Toolbox“) ir tenkinti specifinius programinės įrangos ir aparatūros reikalavimus. Kaip ir anksčiau aptarta *TensorFlow* biblioteka, *MATLAB* programinė įranga pasižymi išsamia dokumentacija, dideliu palaikymu iš bendruomenės ir kūrėjų.

The Microsoft Cognitive Toolkit (CNTK) – atviro kodo programinės įrangos įrankių rinkinys, skirtas kurti komercinės klasės paskirstytas gilaus mokymosi sistemas. Iš pradžių CNTK buvo kurtas kaip įrankis automatinio kalbos atpažinimo tyrimams atlikti, tačiau ilgainiui pavirto apibendrintu, įvairaus tipo problemoms spręsti skirtu gilaus mokymosi algoritmų rinkiniu. CNTK leidžia lengvai realizuoti ir naudoti populiarius gilaus mokymosi modelius: gilius neuroninius tinklus, konvoliucinius neuroninius tinklus, rekurentinius neuroninius tinklus ir pan. CNTK gali būti naudojama kaip atskira biblioteka kuriant ASR sistemas Python, C# ir C++ programavimo kalbomis. Kaip ir anksčiau aptarta programinė įranga, CNTK taip pat leidžia paspartinti skaičiavimus išnaudojant grafinių vaizdo plokščių galimybes.

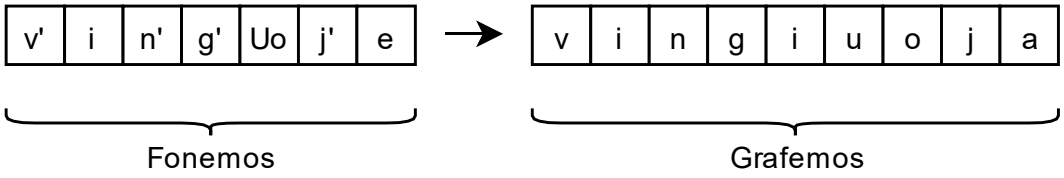
Mašininio mokymosi algoritmų realizacijoms taip pat gali būti panaudotos ir kitos bibliotekos bei įrankiai, tokie kaip *Caffe, Theano, Deeplearning4J* ar *PyTorch*. Baigiamojo darbo realizacijai nuspręsta naudoti TensorFlow biblioteką.

3 Realizacija

3.1 Apie garsyną

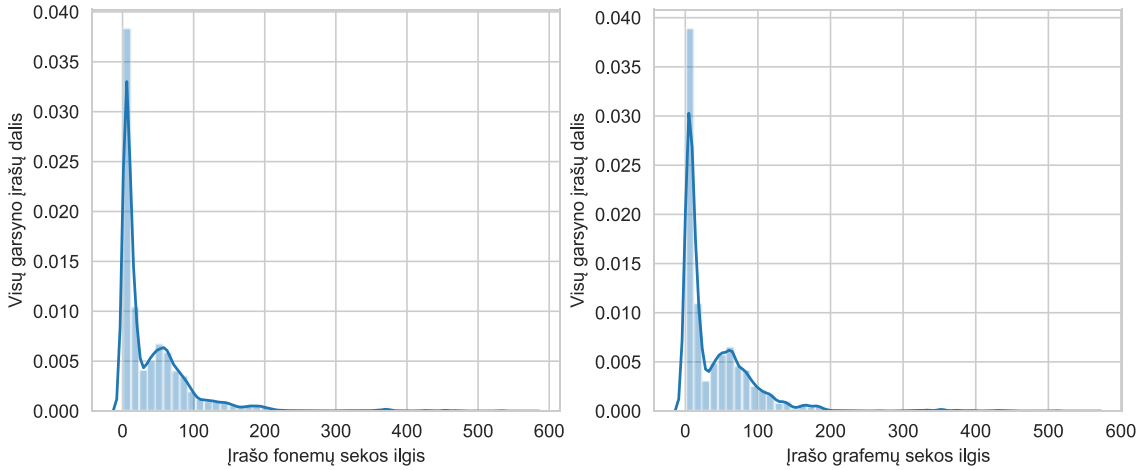
Garsyną sudaro 86 val. anotuotos lietuvių šnekos (406088 žodžiai). Garsyne esantys įrašai yra labai įvairūs: galima rasti trumpų komandų , pavienių žodžių ir ištisų sakinių.

Ši garso įrašų aibė buvo apdorota automatiniu būdu iš garso įrašų išgaunant fonemų sekas, kurios vėliau buvo susietos su atitinkamomis grafemų sekomis, žr. 3.1 pav.



3.1 pav. Fonemų ir jas atitinkančių grafemų sekos pavyzdys

3.2 pav. pateikti įrašų pasiskirstymai pagal juos sudarančių fonemų ir grafemų sekų ilgius. Nesunku pastebėti, kad dauguma įrašų yra labai trumpi (< 10 fonemų), tačiau garsyne galima rasti ir labai ilgų įrašų (iki 572 fonemų). Naudojantis 3.1 lentelėje pateikta įrašų analizės suvestine, galima teigti, kad dažniausiai įrašai turi vos po 2 fonemas: tai trumpi žodžiai (dažniausiai jungtukai ir prielinksniai), tokie kaip „į“, „ir“, „ar“ ir t. t., arba atskiros ištartos raidės „č“, „g“, „c“ ir kt.



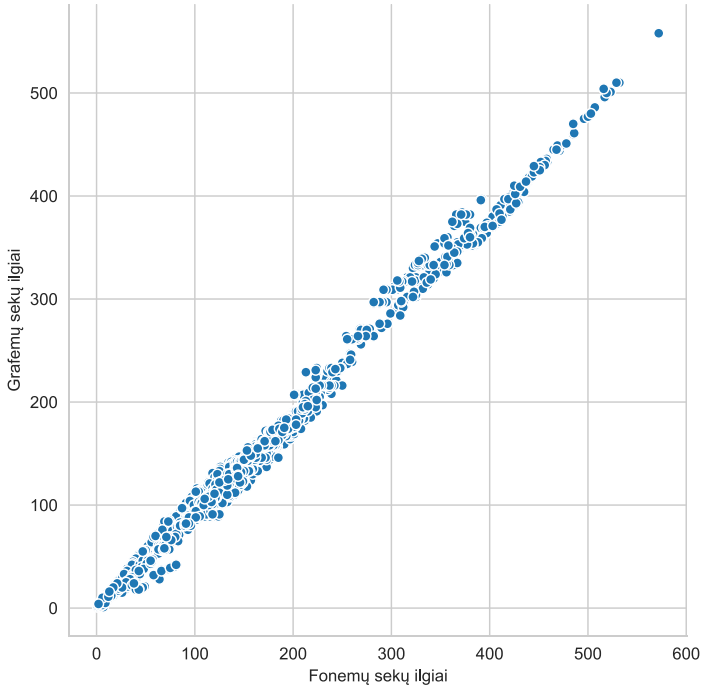
3.2 pav. Įrašų ilgiai fonemomis ir grafemomis

3.1 lentelė Įrašus sudarančių fonemų ir grafemų sekų analizė

Įrašo analizės tipas	Vidutinis įrašo ilgis	Ilgiausio įrašo ilgis	Trumpiausio įrašo ilgis	Įrašo ilgių mediana	Įrašo ilgių moda
Fonemomis	40	572	1	17	2
Grafemomis	40	558	1	17	6

Koreliacijos grafikas tarp fonemų ir grafemų sekų ilgių pateiktas 3.3 pav. Akivaizdu, kad fonemų ir grafemų ilgiai labai stipriai koreliuoja tarpusavyje (0,996713 Pearson koreliacijos koeficientas). Tokio stipraus ryšio egzistavimas leidžia daryti prielaidą, kad statistiniai modeliai mokydamiesi generuoti grafemas pagal jas atitinkančių fonemų sekas gali išnaudoti ne tik sekų turinius, bet ir jų ilgius. Labai svarbu

atkreipti dėmesį į tai, kad ne visos fonemų sekos yra ilgesnės už atitinkamas grafemų sekas. Ši duomenų rinkinio savybė užkerta kelią CTC (angl. *Connectionist Temporal Classification*) tipo modelių panaudojimui, kadangi modelių veikimu būtina užtikrinti, kad įvesties duomenų (fonemų) seka būtų ilgesnė už išvesties (grafemų) seką [32].



3.3 pav. Fonemų ir grafemų sekų koreliacija

Lyginant su paskutiniu metu pasaulinėje praktikoje naudojamais duomenų rinkiniais, šis garsynas yra sąlyginai mažas. Šiuo metu nauji, dar vystomi modeliai yra apmokomi garsynais, kurių trukmės yra iki 12500 valandų [33].

3.2 Garsyno analizė

Šiame skyriuje pateikiama garsyną sudarančių fonemų ir grafemų analizė.

3.2.1 Fonemos

Garsynas sudarytas naudojant 91 skirtingą fonemą. Fonemų dažnių pasiskirstymas pateiktas 7.2 skyriuje (Priedas B), 7.1 pav. Fonemų dažniai grafike pateikti logaritminėje skalėje. Toks sprendimas pasirinktas neatsitiktinai: fonemų dažnių pasiskirstymas yra labai netolygus, t. y. kai kurios fonemos garsyne yra randamos tik kelis šimtus (pvz., /*dz*/, /*h*/, /*tS*/) o kitos – daugiau nei šimtą tūkstančių kartų (pvz., /*a*/, /*s*/, /*i*/, /*k*/).

Svarbu pabrėžti, kad šiame duomenų rinkinyje naudojamos fonemos ir jų skaičius neatitinka lietuvių kalbos fonologijoje apibrėžtų fonemų: naudojamame garsyne yra 91, lietuvių kalboje – 56-59 fonemos [32, 33].

3.2.2 Grafemos

Palyginus su fonemomis, skirtingų grafemų yra daug mažiau – tik 61, tarp kurių yra visos mažosios ir didžiosios lietuvių kalbos abėcėlės raidės, išskyrus Ą, Ę, Ė, Ų, Z ir 2 papildomi skyrybos simboliai – tarpas ir brūkšnys. Taip pat tarp grafemų yra viena raidė, kurios nėra lietuvių kalbos abėcėlėje – w. Šią raidę galima rasti garsyne esančiuose žodžiuose, kurie nusako angliškus pavadinimus: “word”, “powerpoint” ir pan., tačiau tokių žodžių visame garsyne yra tik 15 (tai sudaro mažiau nei 0,004 proc. viso garsyno žodžių).

Grafemų pasiskirstymas pagal dažnius pateiktas 7.2 skyriuje (Priedas B), 7.2 pav. Grafemų dažniai pateikti logaritminėje skalėje.

Analizuojant pateiktą grafiką, galima pastebėti, kad balsės (a, e, i, o, u) garsyne yra randamos daug dažniau nei dauguma priebalsių. Tuo tarpu didžiosios raidės garsyne yra aptinkamos daug rečiau, nei mažosios.

3.2.3 Specialūs simboliai

Garsyne be fonemų ir grafemų taip pat galima rasti specialių simbolių, žyminčių pauzes tarp ištartų žodžių, ar žmogaus kalbėjimo metu skleidžiamus garsus (įkvėpimą, iškvėpimą, nurijimą ir pan.)

Šiuos specialius simbolius galima rasti tiek fonemų, tiek ir jas atitinkančių grafemų sekose. Dažniausiai šiuos simbolius galima rasti įsiterpusius tarp žodžių, tačiau ne kiekvienas žodis garsyne yra atskirtas specialiais simboliais, t. y. šie simboliai nebūtinai žymi žodžio pradžią ar pabaigą.

Kadangi šie simboliai nėra standartinė lietuvių kalbos dalis, jie buvo pašalinti iš grafemų sekų paliekant juos tik fonemų sekose. Manoma, kad jų egzistavimas fonemų sekose turėtų padėti ASR modeliui išmokti atrasti žodžių ribas ir taip palengvinti atpažinimą.

3.3 Gilus mokymasis

Jau nuo devintojo dešimtmečio dauguma ASR sistemų buvo kuriamos naudojant paslėptus Markovo modelius su tam tikromis modifikacijomis. Situacija pasikeitė 2009-2010 m., kai akademikai ir industrijos atstovai pradėjo automatinio kalbos atpažinimo sistemose taikyti giliuosius neuroninius tinklus [36]. Bandymų naudoti dirbtinius neuroninius tinklus buvo ir anksčiau, tačiau jie visi buvo nesėkmingi, kadangi tikslumu juos aplenkdavo HMM-GMM hibridai.

3.3.1 Dirbtiniai neuroniniai tinklai

Dirbtinio neuroninio tinklo tikslas yra aproksimuoti tikrąją funkciją f^*, kuri nusako sąryšį tarp duomenų $\boldsymbol{x}$ ir rezultato $\boldsymbol{y}$ ir kuri iš anksto dažniausiai nėra žinoma:

$$\boldsymbol{y} = f^*(\boldsymbol{x}) \tag{6}$$

ANN naudodami tikrosios funkcijos f^* artinį f, kuris yra parametrizuojamas parametrais $\boldsymbol{\theta}$, apibrėžia sąryšį tarp $\boldsymbol{x}$ ir $\boldsymbol{y}$:

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}), \tag{7}$$

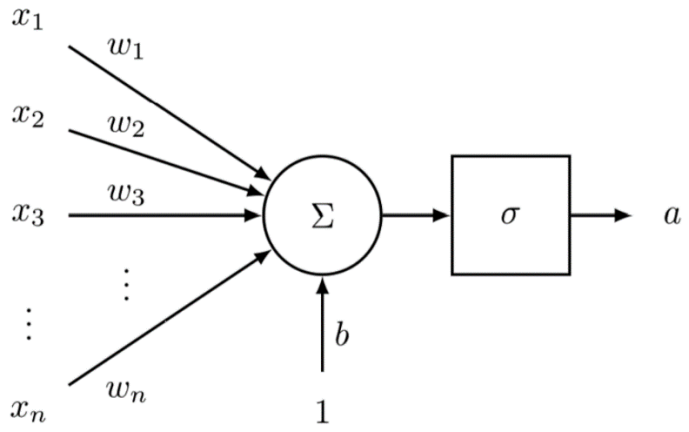
$\boldsymbol{y}$ yra tinklo pateikiamas rezultatas (išvestis), $\boldsymbol{x}$ – duomenys (įvestis), $\boldsymbol{\theta}$ – tinklo parametrai.

Dirbtiniai neuroniniai tinklai yra konstruojami iš dirbtinių neuronų – žmonių ir gyvūnų smegenyse randamų biologinių neuronų matematinių atitikmenų. Dirbtinį neuroną galima laikyti abstrakčiu skaičiavimo vienetu, kuris naudodamas gautus įvesties duomenis apskaičiuoja išvesties reikšmę (rezultatą).

Kiekvienas neuronas skiriasi generuojama reikšme, kuri priklauso ne tik nuo į neuroną patenkančių duomenų, bet ir nuo neurono viduje esančių parametrų: svorių ir aktyvavimo funkcijos. Paprasčiausią, „logistinės regresijos“ tipo dirbtinį neuroną galima aprašyti formule:

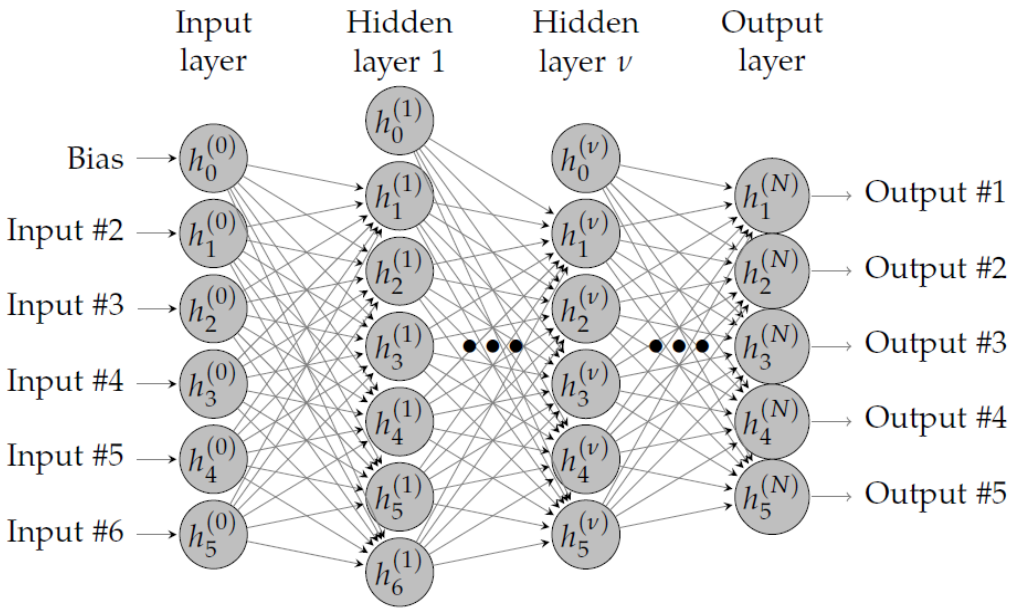
$$a = \frac{1}{1 + \mathrm{e}^{(\boldsymbol{w}^T + b)}}, \tag{8}$$

čia a – neurono pateikiamas rezultatas, $\boldsymbol{x} \in \mathbb{R}^n$ yra duomenų vektorius, $\boldsymbol{w} \in \mathbb{R}^n$ yra svorių vektorius, o b – fiksuotas, laisvas svoris. Grafinis dirbtinio neurono pavyzdys pateiktas 3.5 pav.



3.4 pav. Dirbtinio neurono pavyzdys [37]

Dirbtiniuose neuroniniuose tinkluose neuronai yra grupuojami į sluoksnius, kurie vėliau yra sujungiami tarpusavyje. Visi dirbtiniai neuroniniai tinkai turi mažiausiai 3 sluoksnius: vieną įvesties, vieną (ar daugiau) paslėptą ir vieną išvesties. ANN, turintys tik 3 sluoksnius, yra vadinami negiliaisiais neuroniniais tinklais (angl. *Shallow Neural Networks*). Gilūs neuroniniai tinklai (angl. *Deep Neural Networks*) naudoja ne vieną, o daugiau paslėptų sluoksnių. Tokios tinklų struktūros leidžia modeliuoti sudėtingus, netiesinėmis savybėmis pasižyminčius duomenis, t. y. atrasti ir išmokti sudėtingesnius ryšius tarp duomenų požymių. Gilaus neuroninio tinklo, kuris turi *N+1* neuronų sluoksnių, pavyzdys pateiktas 3.5 pav.



3.5 pav. Gilus dirbtinis neuroninis tinklas su N+1 sluoksnių (N-1 iš jų yra paslėpti)[38]

Dirbtiniai neuroniniai tinklai, kaip ir dirbtiniai neuronai, gali būti aprašyti vektoriais. Pavyzdžiui, vienas ANN sluoksnis, sudarytas iš m dirbtinių neuronų, aprašytų anksčiau, gali būti pateiktas kaip vieno ANN sluoksnio rezultatų vektorius:

$$\begin{aligned} \boldsymbol{a} &= \begin{bmatrix} a^{(1)} \\ \dots \\ a^{(m)} \end{bmatrix} = \sigma(\boldsymbol{W}x + \boldsymbol{b}) \\ \boldsymbol{b} &= \begin{bmatrix} b^{(1)} \\ \dots \\ b^{(m)} \end{bmatrix} \\ \boldsymbol{W} &= \begin{bmatrix} w^{(1)} \\ \dots \\ w^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times n}, \end{aligned} \tag{9}$$

čia $\boldsymbol{a}^{(i)} \in \mathbb{R}^m$ yra i-ojo neurono, esančio aprašomame sluoksnyje, aktyvacijos funkcijos išėjimas, $\sigma(\cdot)$ – aktyvacijos funkcija, o $\boldsymbol{W}$, $\boldsymbol{x} \in \mathbb{R}^n$ ir $\boldsymbol{b}$ atitinkamai yra neuronų svorių matrica, duomenų vektorius ir fiksuotų svorių vektorius.

Pasinaudojant vektorinėmis išraiškomis, visą dirbtinių neuronų tinklą galima aprašyti viena lygtimi, pvz., ANN, sudarytas iš dviejų dirbtinių neuronų sluoksnių, gali būti aprašytas kaip:

$$\boldsymbol{y} = \sigma\big(\boldsymbol{W_2}\big(\sigma(\boldsymbol{W_1}\boldsymbol{x} + \boldsymbol{b_1})\big) + \boldsymbol{b_1}\big), \tag{10}$$

čia $\boldsymbol{W_1}$ ir $\boldsymbol{b_1}$ pirmojo sluoksnio svorių ir laisvųjų svorių matricos, $\boldsymbol{W_2}$ ir $\boldsymbol{b_2}$ antrojo sluoksnio svorių ir laisvųjų svorių matricos.

Yra skiriami trys neuroninių tinklų mokymosi tipai [39]:

- **Mokymasis su mokytoju** (angl. *supervised learning*) – tai mokymosi tipas, kai algoritmui pateikiamos įvesties-išvesties duomenų poros, pagal kurias jis bando išmokti funkciją, leidžiančią susieti įvesties duomenis su išvesties rezultatais. Apmokytas modelis gali būti naudojamas dar nematytų duomenų apdorojimui, jei rasta funkcija sugebės pakankamai gerai generalizuoti mokymosi metu pateiktus duomenis.

- **Mokymasis be mokytojo** (angl. *unsupervised learning*). Tokio mokymosi metu algoritmai mokosi be grįžtamojo ryšio, t. y. algoritmai neturi informacijos apie tai, ar tinkamai atliekama duomenų klasifikacija. Tokio tipo mokymasis dažnai naudojamas grupavimo (angl. *clusstering*) uždaviniams spręsti randant duomenyse požymius ar jų sekas, pagal kurias jie galėtų būti grupuojami tarpusavyje.
- **Stiprinamasis mokymasis** (angl. *reinforcement learning*). Šio mokymosi idėja yra paremta serija skatinimų ir bausmių, pagal kurias algoritmas mokosi.

Gilieji ir negilieji neuroniniai tinklai be grįžtamųjų ryšių yra apmokomi naudojant gradientinio nusileidimo metodą. Algoritmo veikimo metu dirbtinių neuronų svorių reikšmės yra modifikuojamos taip, kad būtų padidintas ANN pateikiamų rezultatų tikslumas, drauge bandant rasti tinklo rezultatų nuostolių funkcijos globalų minimumą.

Pirmiausia algoritmas apskaičiuoja nuostolių funkcijos reikšmę $\mathcal{L}$, kuri nurodo, kaip klaidingai veikia statistinis modelis. Priklausomai nuo užduoties tipo ir duomenų rinkinio gali būti naudojamos skirtingos nuostolių matavimo funkcijos, tokios kaip vidutinė kvadratinė paklaida (angl. *mean squared error*), kryžminė entropija (angl. *cross entropy*) ir t.t. Žinant bendrą tinklo pateikiamų rezultatų paklaidą $\mathcal{L}$, k svoriai $\boldsymbol{w_1, w_2 \dots w_k}$ yra keičiami naudojant iteratyvų gradientinio nusileidimo procesą, kurio metu apskaičiuojamas gradientas (11) ir atnaujinamos svorių reikšmės apskaičiuotu prieauginiu dydžiu (12).

$$\nabla \boldsymbol{\mathcal{L}} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{w_1}}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{w_2}} \dots \frac{\partial \mathcal{L}}{\partial \boldsymbol{w_k}}\right) \tag{11}$$

$$\Delta \boldsymbol{w_i} = -\gamma \frac{\partial \mathcal{L}}{\partial \boldsymbol{w_i}}, \tag{12}$$

čia $\Delta \boldsymbol{w_i}$ yra inkrementinis dydis, kuriuo bus padidintas i-ojo neurono svoris; γ – mokymosi žingsnis, reguliuojantis, kaip greitai bus leidžiamasi neigiama gradiento puse.

3.3.2 Rekurentiniai neuroniniai tinklai

Nors įprasti ANN gali būti naudojami automatinio kalbos atpažinimo uždaviniams spręsti, jų galimybės yra labai ribotos – jie priima duomenis, pateiktus kaip fiksuoto dydžio vektorius, ir pateikia rezultatus fiksuoto dydžio vektorių formatu (pvz., tikimybių pasiskirstymą skirtingoms duomenų klasėms). Šnekos atpažinimo uždaviniams spręsti dažniau naudojami specialūs dirbtinių neuroninių tinklų modeliai – rekurentiniai neuroniniai tinklai (angl. *recurrent neural networks,* toliau RNN), kurie yra skirti dirbti su laiko atžvilgiu besikeičiančiais duomenimis (pvz.: akustiniais signalais) ir neturi duomenų sekos ilgio apribojimų.

Kitaip nei paprasti ANN, rekurentiniuose neuroniniuose tinkluose esantys dirbtiniai neuronai yra sujungti ir grįžtamaisiais ryšiais. Tokių ryšių buvimas lemia išskirtinę RNN savybę – atmintį, leidžiančią rekurentiniams neuroniniams tinklams apskaičiuoti rezultatą ne tik pasinaudojant konkrečiu momentu pateiktais, bet ir prieš tai buvusiais duomenimis. Kitaip tariant, rekurentinio neuroninio tinklo sprendimas, priimtas $t-1$ laiko momentu, bus panaudotas priimant sprendimą t laiko momentu. Šiuo požiūriu RNN primena žmogaus smegenis – sprendimai apie įvykius priimami remiantis patirtimi ir neseniai įvykusias įvykiais.

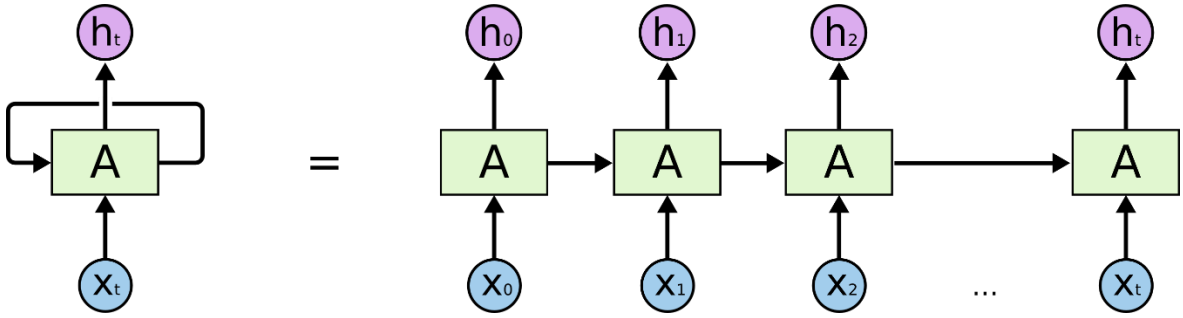
Grįžtamais ryšiais RNN realizuojama atmintis dar vadinama paslėpta tinklo būsena. Matematiškai rekurentinius neuroninius tinklus galima aprašyti formulėmis (13) ir (14), kur

- $\boldsymbol{h}_t$ – paslėptos tinklo būsenos vektorius laiko momentu t;
- $\boldsymbol{x}_t$ – duomenų vektorius laiko momentu t;
- $\boldsymbol{h}_{t-1}$ – paslėptos tinklo būsenos vektorius, gautas po netiesinės funckijos laiko momentu $t-1$
- $\boldsymbol{W}^{hx}$ – svorių matrica, skirta apdoroti duomenų vektorių x_t;
- $\boldsymbol{W}^{hh}$ – svorių matrica, panaudota generuojant h_{t-1};
- $\sigma(\cdot)$ – netiesinė funkcija;
- $\hat{y}_t$ – sprendimas (hipotezė), gautas laiko momentu t;
- $\boldsymbol{W}^{(S)}$ – paskutiniojo sluoksnio svorių matrica;
- $softmax()$ – minkšto maksimumo funkcija.

$$\boldsymbol{h}_t = \sigma(\boldsymbol{W}^{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}^{hx}\boldsymbol{x}_t) \tag{13}$$

$$\hat{y}_t = softmax(\boldsymbol{W}^{(S)}\boldsymbol{h}_t) \tag{14}$$

Rekurentinio neuroninio tinklo veikimo grafinis pavyzdys pateiktas 3.6 pav.: kairėje pusėje pateiktas bendras tinklo modelis (pažymėtas A), duomenų seka $\boldsymbol{X}_t$ ir tinklo generuojamų hipotezių seka $\boldsymbol{h}_t$; dešinėje pusėje parodytas „laike išskleisto“ tinklo modelis: formuojant $\boldsymbol{h}_i$ hipotezę naudojami ne tik $\boldsymbol{X}_i$ duomenys, bet ir tinklo būsena, gauta generuojant $\boldsymbol{h}_{i-1}$ hipotezę.



3.6 pav. Supaprastinta rekurentinio neuroninio tinklo diagrama [40]

Šios savybės leidžia rekurentiniams neuroniniams tinklams išmokti požymius, kuriais pasižymi duomenų sekos, todėl dažnai yra naudojami su laiku kintančios informacijos apdorojimui, pvz.: automatiniam kalbos atpažinimui [3, 40, 41], kalbos modeliavimui, vertimui ir antraščių kūrimui [43] ir pan.

Tradiciniai RNN susiduria su problemomis modeliuojant duomenų sekas, kurios pasižymi ilgalaikėmis priklausomybėmis. Kadangi modelio apmokymo metu jo skaičiuojamasis grafas yra „išskleidžiamas“, grafo dydis priklauso nuo apdorotos duomenų sekos ilgio. Apdorojant ilgas sekas, gradientų reikšmės, atsakingos už svorių keitimą, gali tapti labai mažos, t. y. susidaryti nykstančių gradientų problema. Mažos gradientų reikšmės sulėtina tinklo mokymąsi, o kritiniais atvejais gali šį procesą visiškai sustabdyti.

LSTM tinklai (angl. *Long Short Term Memory networks*) – specialūs RNN tipo tinklai, galintys išmokti ilgalaikes duomenų priklausomybes. Tokio tipo tinklai turi specialias atminties ląsteles, leidžiančias išlaikyti informaciją ilgą laiką. Kitaip nei įprastiniai RNN, kurie perrašo saugomą informaciją kiekviename apdorojimo proceso žingsnyje, LSTM ląstelės aptinka svarbias savybes, kuriomis pasižymi duomenų seka, ir įsimena jas ilgam laikui. Į ląsteles informacija gali būti įrašoma, jose saugoma ir iš jų skaitoma, labai panašiai kaip ir dirbant su įprasta kompiuterio atmintimi, tačiau LSTM ląstelės pačios pasirenka, kiek ir kokios informacijos bus saugoma bei kaip ilgai tai truks.

LSTM ląstelės turi tris „vartus“: $\boldsymbol{i}$ – informacijai patekti, $\boldsymbol{o}$ – duomenims grąžinti ir $\boldsymbol{f}$ – informacijai užmiršti. „Vartai“ gali būti atidaromi ir uždaromi, o jų būsenas lemia apdorojamų duomenų $\boldsymbol{x}_t$ ir ankstesnės

tinklo būsenos $\boldsymbol{h}_{t-1}$ kombinacija. „Vartai“ $\boldsymbol{i}_t$ ir $\boldsymbol{f}_t$ kontroliuoja, kiek informacijos bus išsaugoma ir ištrinama ląstelėje, o „vartai“ $\boldsymbol{o}_t$ valdo informacijos iš ląstelės perdavimą (grąžinimą) į kitas dirbtinio neuroninio tinklo dalis konkrečiu laiko momentu t. „Vartų“ būsenų reikšmės apskaičiuojamos pagal formules (15), (16) ir (17), kur laužtiniai skliaustai žymi vektorių susiejimo (angl. *concatenation*) operaciją, σ – netiesinę aktyvavimo funkciją, $\boldsymbol{W}_f$ ir $\boldsymbol{b}_f$ – informacijos užmiršimo „vartų“ svorių matricas; $\boldsymbol{W}_i$ ir $\boldsymbol{b}_i$ – informacijos patekimą į ląstelę reguliuojančių vartų svorių matricas; $\boldsymbol{W}_o$ ir $\boldsymbol{b}_o$ – duomenų grąžinimo „vartų“ svorių matricas. Svarbiausia ląstelės dalis – vidinė ląstelės būsena $\boldsymbol{s}_t$, kuri žymi ląstelės talpinamą informaciją laiko momentu t ir yra apskaičiuojama naudojantis formule (18), kur $\boldsymbol{f}_t$ – informacijos užmaršties „vartų“ reikšmė, $\boldsymbol{s}_{t-1}$ – vidinė ląstelės būsena laiko momentu $t-1$, $\boldsymbol{i}_t$ – informacijos patekimo „vartų“ reikšmė, o $\tilde{\boldsymbol{s}}_t$ – informacija, kuri pretenduoja būti įrašyta į ląstelę. Pastaroji reikšmė priklauso nuo prieš tai ląstelės grąžintos tinklo būsenos $\boldsymbol{h}_{t-1}$ ir įvesties duomenų vektoriaus $\boldsymbol{x}_t$ ir yra apskaičiuojama naudojant formulę (19), kur $\boldsymbol{W}_{\tilde{s}}$ ir $\boldsymbol{b}_{\tilde{s}}$ yra svorių matricos.

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \tag{15}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \tag{16}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \tag{17}$$

$$\boldsymbol{s}_t = \boldsymbol{f}_t \cdot \boldsymbol{s}_{t-1} + \boldsymbol{i}_t \cdot \tilde{\boldsymbol{s}}_t \tag{18}$$

$$\tilde{\boldsymbol{s}}_t = (\boldsymbol{W}_{\tilde{s}} \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\tilde{s}}) \tag{19}$$

Kartais vietoje LSTM ląstelių naudojamos GRU ląstelės (angl. *Gated Recurrent Units*). GRU – tai ląstelės, labai panašios į LSTM ląsteles. Didžiausias skirtumas tarp jų ir LSTM ląstelių – mažesnis parametrų skaičius, kadangi tik vieni „vartai“ kontroliuoja, kiek informacijos bus užmiršta ir kiek įsimenama. Tokio tipo ląstelėse yra tik dveji „vartai“: „vartai“ $\boldsymbol{r}$ nustato, kaip informacija ląstelėje bus sujungta su ateinančiais duomenimis; „vartai“ $\boldsymbol{u}$ nusako, kiek informacijos pasilikti. „Vartų“ reikšmės apskaičiuojamos pagal formules (20) ir (21), kur $\boldsymbol{h}_{t-1}$ yra tinklo paslėpta būsena laiko momentu $t-1$, $\boldsymbol{x}_t$ – įvesties duomenų vektorius laiko momentu t, $\boldsymbol{W}_u$, $\boldsymbol{b}_u$, $\boldsymbol{W}_r$ ir $\boldsymbol{b}_r$ – $\boldsymbol{u}$ ir $\boldsymbol{r}$ „vartų“ svorių matricos.

Nauja būsena $\boldsymbol{h}_t$ yra apskaičiuojama naudojantis suskaičiuotomis „vartų“ $\boldsymbol{u}_{t-1}$ ir $\boldsymbol{r}_{t-1}$ reikšmėmis pagal (22) formulę, kur $\boldsymbol{U}_h$, $\boldsymbol{W}_h$ ir $\boldsymbol{b}_h$ yra svorių matricos.

Tiek LSTM, tiek GRU leidžia pasiekti labai panašius rezultatus vertinant tinklų klasifikacijos tikslumus [44].

$$\boldsymbol{u}_t = \sigma(\boldsymbol{W}_u \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_u) \tag{20}$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) \tag{21}$$

$$\boldsymbol{h}_t = \boldsymbol{u}_{t-1}\boldsymbol{h}_{t-1} + (\boldsymbol{1} - \boldsymbol{u}_{t-1}) \cdot \sigma(\boldsymbol{U}_h\boldsymbol{x}_{t-1} + \boldsymbol{W}_h\boldsymbol{r}_{t-1}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{22}$$

Dvikrypčiai rekurentiniai neuroniniai tinklai (angl. *Bidirectional Recurrent Neural Networks*) yra paremti idėja, jog generuojama hipotezė priklauso ne tik nuo prieš tai buvusių duomenų, bet ir nuo būsimų. Dvikrypčiuose neuroniniuose tinkluose kiekvienas sluoksnis turi po dvi eiles dirbtinių neuronų, kurie sujungti priešingomis kryptimis: vienas sluoksnis apdoroja įvesties duomenų seką iš kairės į dešinę $(x_1, x_2 \dots, x_T)$, kitas sluoksnis – iš dešinės į kairę $(x_T, x_{T-1} \dots, x_1)$. Dvikrypčiai rekurentiniai neuroniniai

tinklai dažnai taikomi dėl savo savybės, leidžiančios generuoti rezultatus naudojant duomenis apie praeities ir ateities įvykius. Dvikrypčius neuroninius tinklus galima sudaryti ir iš LSTM ar GRU ląstelių.

3.3.3 RNN apmokymas

Rekurentiniai neuroniniai tinklai ir jų modifikacijos yra apmokomi naudojant gradientinio nusileidimo metodus. Kadangi RNN dirbtinių neuronų svorių gradientai stochastinio gradientinio nusileidimo metu priklauso ne nuo vieno įvesties kintamojo x_t, o nuo kintamųjų sekos $x_1, x_2 \dots, x_T$, mokymosi metu jie yra apskaičiuojami naudojant atgalinio klaidos sklidimo laike (angl. *backpropagation thought time*) algoritmą [45].

Algoritmo veikimas yra paremtas RNN modelio skaičiuojamojo grafo „išskleidimu“ laiko atžvilgiu. Išskleidus grafą dirbtinių neuronų svorių gradientus galima apskaičiuoti naudojant standartinį klaidos sklidimo atgal algoritmą.

3.4 Užkoduotojo-dekoduotojo tipo modeliai

Daug lankstesni yra RNN paremti užkoduotojo-dekoduotojo tipo modeliai (angl. *encoder-decoder models*), kurie naudojami sprendžiant mašininio vertimo [44, 45], teksto reziumavimo [48], automatinio vaizdų antraščių generavimo [49], 48] ir kt. uždaviniuose. Vienas žinomiausių tokio tipo modelių pavyzdžių – Sutskever ir kt. [46] pasiūlytas iš dviejų RNN (užkoduotojo ir dekoduotojo) sudarytas mašininio vertimo modelis.

Tokio tipo modelyje užkoduotojas (angl. *encoder*) nuskaito duomenų seką $X = (x_1, x_2 \dots, x_T)$ ir paverčia ją į kontekstinį vektorių $\boldsymbol{c}$. Vienas dažniausių metodų yra naudoti tokį RNN, kuris $\boldsymbol{c}$ apskaičiuotu iš paslėptų RNN būsenų ($\boldsymbol{h}_1, \boldsymbol{h}_2 \dots \boldsymbol{h}_T$):

$$\boldsymbol{h}_t = f(x_t, \boldsymbol{h}_{t-1}) \tag{23}$$

$$\boldsymbol{c} = q(\{(\boldsymbol{h}_1, \boldsymbol{h}_2 \dots \boldsymbol{h}_T\}), \tag{24}$$

čia $\boldsymbol{h}_t \in R^n$ yra paslėpta būsena laiko momentu t, o q ir f yra pasirinktos netiesinės funkcijos, pvz., iš LSTM ląstelių sudaryti dirbtiniai neuroniniai tinklai [47].

Dekoduotojo paskirtis nuspėti kitą sekos elementą $y_{t'}$ pasinaudojant konteksto vektoriumi $\boldsymbol{c}$ ir iki šiol sugeneruota žodžių seka $\{y_1, y_2 \dots y_{t'-1}\}$, t. y. dekoduotojas apibrėžia tikimybę, kad buvo ištarta seka Y, išskaidydamas bendrą tikimybę į atitinkamas sąlygines tikimybes:

$$p(Y) = \prod_{t'=1}^{T'} p(y_{t'} | \{y_1, y_2 \dots y_{t'-1}\}, \boldsymbol{c}) \tag{25}$$

Naudojant RNN, šias sąlygines tikimybes galima apibrėžti kaip:

$$p(y_{t'} | \{y_1, y_2 \dots y_{t'-1}\}, \boldsymbol{c}) = g(y_{t'-1}, \boldsymbol{s}_{t'}, \boldsymbol{c}), \tag{26}$$

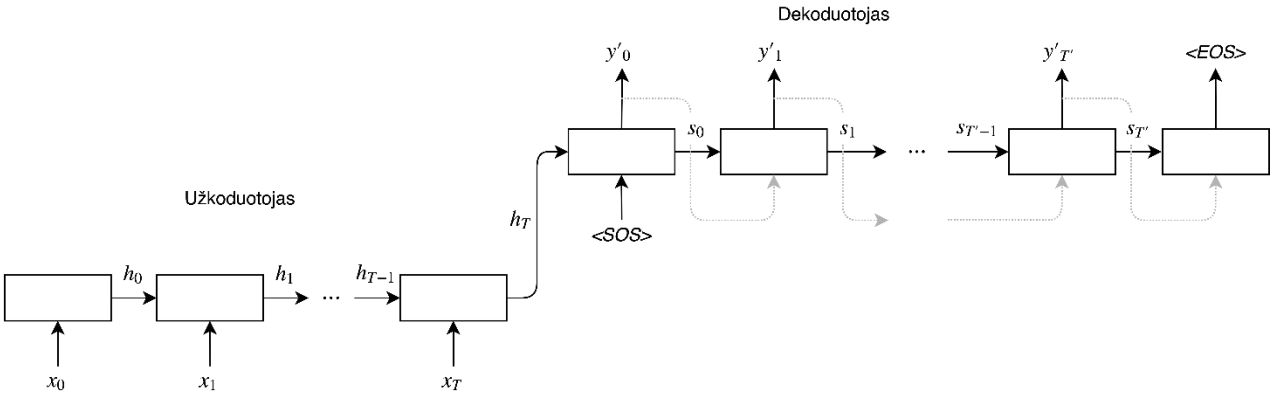
čia $\boldsymbol{s}_t$ yra paslėpta RNN būsena, o g – netiesinė funkcija (pvz., iš LSTM ląstelių sudarytas neuroninis tinklas).

Įprastai dekoduotojo realizavimui panaudojamas RNN su keletu LSTM ląstelių sluoksnių, kurių paskutinis – tankiai sujungtų dirbtinių neuronų sluoksnis su minkšto maksimumo (angl. *softmax*) funkcija:

$$p(y_{t'}|\{y_1, y_2 \dots y_{t'-1}\}, \boldsymbol{c}) = softmax(\boldsymbol{W}_s \boldsymbol{h}_{t'}), \quad (27)$$

čia $\boldsymbol{W}_s$ yra tankiai sujungtų neuronų sluoksnio svorių matrica, $\boldsymbol{h}_{t'}$ yra laiko momentu t' sugeneruota RNN paslėpta būsena.

Tokio modelio grafinis pavyzdys yra pateiktas 3.7 pav., kur konteksto vektorius yra paskutinė užkoduotojo būsena. Kadangi šie modeliai yra skirti darbui su kintamo ilgio sekomis, jie naudoja du specialius simbolius: sekos pradžios simbolį <*SOS*> ir sekos pabaigos simbolį <*EOS*>. Sekos pradžios simbolis yra naudojamas kaip pradžios signalas dekodavimo daliai, o modeliui sugeneravus sekos pabaigos simbolį <*EOS*> tolimesnės sekos elementų generavimas yra sustabdomas.



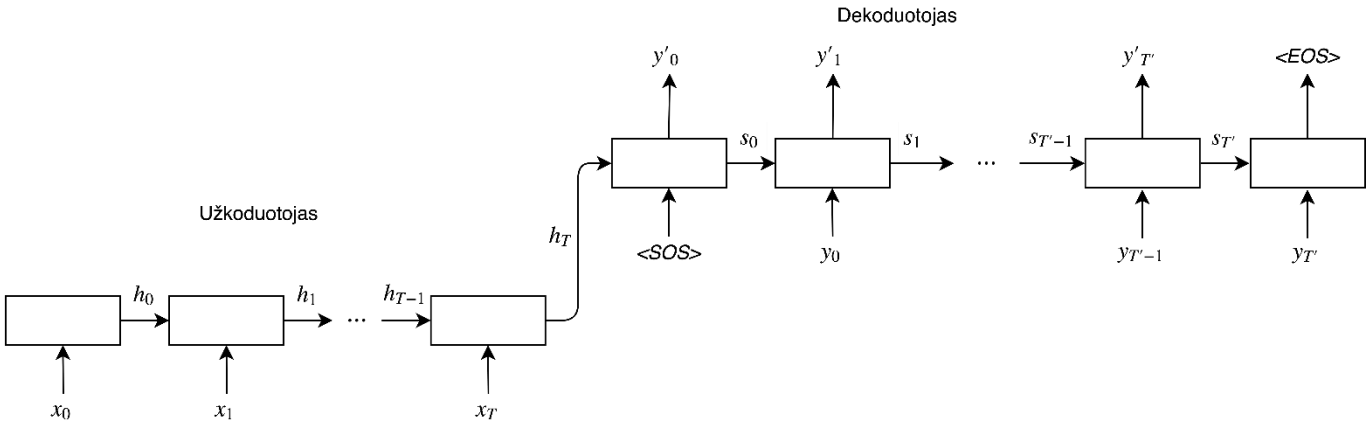
3.7 pav. Užkoduotojo-dekoduotojo tipo architektūros modelis

Siekdami pagerinti tinklo mokymosi greitį ir jo tikslumą, tyrėjai pasiūlė keletą patobulinimų tokio tipo modeliams.

3.4.1 Apmokymo per prievartą metodas

Apmokymo per prievartą (angl. *teacher forcing*) metodas leidžia greičiau ir efektyviau apmokyti RNN, kuris naudoja iki žingsnio $t+1$ sugeneruotos sekos elementus $(y'_1, y'_2 \dots y'_t)$ ir konteksto vektorių $\boldsymbol{c}$ kito elemento y'_{t+1} generavimui. Pasitelkus šį metodą RNN galima apmokyti daug sparčiau, jei elemento y'_{t+1} generavimui bus naudojama ne modelio prieš tai pateiktų elementų seka, o išorinis signalas $D = (d_1, d_2 \dots d_{t-1})$, pvz., teisinga elementų seka Y (žr. 3.8 pav.)

Užkoduotojo-dekoduotojo tipo architektūrose toks metodas yra naudojamas siekiant greičiau apmokyti dekoduotojo RNN.



3.8 pav. Apmokymo per prievartą taikymas užkoduotojo-dekoduotojo tipo modeliuose

3.4.2 Mokymasis naudojant atbulas sekas

Sutskever ir kt. [46] pastebėjo, kad tokio tipo modelius galima apmokyti greičiau, jei mokymosi proceso metu užkoduotojui perduodamos ir atbulos įvesties duomenų sekos, pvz., turint įvesties ir išvesties sekas x_1, x_2, x_3 ir y_1, y_2, y_3, modelis apmokomas sugeneruoti y_1, y_2, y_3 pagal duomenų seką x_3, x_2, x_1. Autorių teigimu toks metodas leidžia modeliui geriau išmokti priklausomybes tarp įvesties ir išvesties signalų, kadangi mokymosi metu x_1 yra arčiau y_1, x_2 arčiau y_2 ir t.t. Pritaikus šį mokymosi metodą modeliai pasiekia geresnių rezultatų mašininio vertimo uždaviniuose [46].

3.4.3 Gradiento apkarpymas

Rekurentiniai neuroniniai tinklai dažnai kenčia nuo „sprogstančių“ gradientų problemos (angl. *exploding gradients*), kai mokymosi metu gradiento norma išauga taip, kad atsiranda skaitmeninio nestabilumo problemos.

Šią problemą galima spręsti „apkarpant“ gradientą [38], t. y. nustatant minimalią ir maksimalią gradiento reikšmes. Įprastai tai gali būti atliekama normalizuojant gradientą, kai šis viršija nustatytą ribą panaudojant gradiento L2 normą:

$$\nabla f = \nabla f \times \frac{\alpha}{\|\nabla f\|}, \tag{28}$$

čia ∇f yra apskaičiuotas gradientas, ‖∇f‖ – jo L2 norma, o α – gradiento riba [51].

3.5 Modeliai su dėmesiu

Vienas iš didžiausių atradimų, padidinęs užkoduotojo-dekoduotojo tipo modelių tikslumą, buvo dėmesio mechanizmo (angl. *attention mechanism*) pritaikymas. Modeliai su dėmesio mechanizmu geba susikoncentruoti į svarbias įvesties sekos dalis išvesties generavimo metu.

Užkoduotojo-dekoduotojo architektūros modeliuose visa įvesties seka yra užkoduojama į vieną konteksto vektorių $\boldsymbol{c}$, kuris paskui yra panaudojamas dekodavimo metu generuojant išvesties seką. Naudojant tokį sprendimą modeliai veikia pakankamai gerai, kai įvesties sekos yra trumpos, tačiau nepasiekia gerų rezultatų su ilgesnėmis sekomis. Modeliai su dėmesio mechanizmu bando spręsti šią problemą leisdami dekoduotojui „žvilgtelėti“ atgal į įvesties seką kiekviename išvesties sekos generavimo žingsnyje.

Luong ir kt. [52] pasiūlė du dėmesio mechanizmo variantus: globalaus ir lokalaus, tačiau šiame darbe aptariamas tik paprastesnis, globalaus dėmesio mechanizmas.

Modeliai su tokio tipo dėmesiu naudoja specialų kintamo ilgio lygiavimo vektorių $\boldsymbol{a}_{t'}$, kurio ilgis yra lygus šaltinio sekos ilgiui. Lygiavimo vektorius yra gaunamas palyginant laiko momentu t' gautą dekoduoto paslėptą būseną $s_{t'}$ su kiekviena užkoduotojo paslėpta būsena h_t:

$$\boldsymbol{a}_{t'}(t) = \frac{e^{score(\boldsymbol{s}_{t'}, \boldsymbol{h}_t)}}{\sum_{i=0}^{T-1} e^{score(\boldsymbol{s}_{t'}, \boldsymbol{h}_i)}}, \tag{29}$$

čia $score$ yra palyginimo funkcija tarp dviejų dekoduotojo ir užkoduotojo būsenų. Loung ir kt. [52] pasiūlė keletą tokios funkcijos variantų, iš kurių paprasčiausia yra skaliarinė sandauga tarp dviejų būsenos vektorių:

$$\text{score}(\boldsymbol{s}_{t'}, \boldsymbol{h}_t) = \boldsymbol{s}_{t'}^{\top} \boldsymbol{h}_t \tag{30}$$

Apskaičiavus lygiavimo vektorių $\boldsymbol{a}_{t'}$, konteksto vektorius $\mathbf{c}_{t'}$ dekodavimo žingsniui t' yra gaunamas padauginus kiekvieną užkoduotojo būseną iš lygiavimo vektoriaus ir susumavus gautą rezultatą:

$$\mathbf{c}_{t'} = \sum_{T}^{t=0} \boldsymbol{a}_{t'} \boldsymbol{h}_t \tag{31}$$

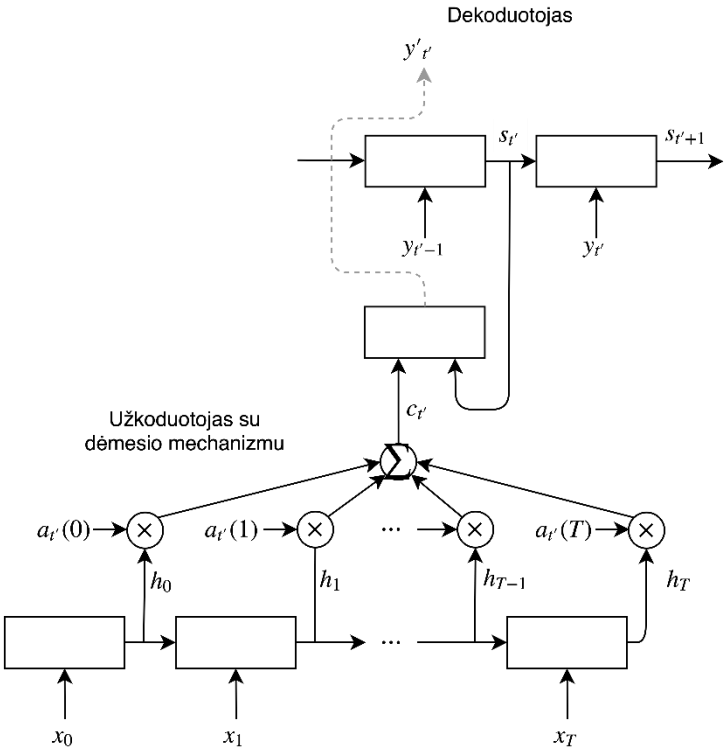
Šis konteksto vektorius yra sujungiamas paslėpta dekoduotojo būsena, o rezultatas perduodamas į pilnai sujungtų dirbtinių neuronų sluoksnį su minkšto maksimumo funkcija:

$$\widetilde{\boldsymbol{h}}_{t'} = tanh(\boldsymbol{W}_c[\boldsymbol{c}_{t'}; \boldsymbol{h}_{t'}]) \tag{32}$$

$$p(y_{t'}|\{y_1, y_2 \dots y_{t'-1}\}, \boldsymbol{c}) = softmax\big(\boldsymbol{W}_s \widetilde{\boldsymbol{h}}_{t'}\big), \tag{33}$$

čia $\boldsymbol{c}_{t'}$ yra konteksto vektorius, apskaičiuotas naudojant dėmesio mechanizmą, $\boldsymbol{W}_c$ svorių matrica, $tanh$ - hiperbolinio tangento funkcija.

Grafinis dėmesio mechanizmo pavyzdys pateiktas 3.9 pav.



3.9 pav. Modelis su dėmesio mechanizmu

3.6 Reguliarizacijos strategijos

Viena didžiausių dirbtinių neuroninių tinklų problemų yra tinklo savybė per daug prisitaikyti prie mokymosi duomenų (angl. *to overfit*). Per didelis prisitaikymas mokymosi metu lemia prastą modelio generalizaciją ir veikimą praktikoje. Egzistuoja daug skirtingų strategijų, kurių paskirtis – sumažinti per didelio tinklo prisitaikymo tikimybę, pvz., duomenų rinkinio augmentacija, modelių ansamblių panaudojimas ir pan. Toliau aprašytos strategijos, naudotos šiame darbe atliktų eksperimentų metu.

3.6.1 Dirbtinių neuronų iškritimas

Dirbtinių neuronų iškritimas(angl. *Dropout*) yra efektyvi ir didelių skaičiavimo resursų nereikalaujanti reguliarizacijos strategija, pasiūlyta Srivastava ir kt.[53]. Visa šios strategijos esmė yra atsitiktinis dirbtinių neuronų iškritimas iš neuroninio tinklo sluoksnių mokymosi metu, taip siekiant sumažinti priklausomybes tarp dirbtinių neuronų porų [45].

Iškritimą galima laikyti specialiu *Bagging* strategijos atveju. *Bagging* metu iš bazinio duomenų rinkinio sukonstruojama K skirtingų duomenų rinkinių, taikant ėmimo su pakeitimu metodą. Kiekvienas sudarytas duomenų rinkinys yra panaudojamas vienam iš K modelių apmokyti [45].

Dirbtinių neuronų iškritimo metu šį procesą yra bandoma aproksimuoti apmokant neuroninių tinklų ansamblį, kurį būtų galima sudaryti pašalinus visas įmanomas dirbtinių neuronų kombinacijas, naudojant tą patį duomenų rinkinį. Paprasčiausiai šį procesą galima atlikti ANN pritaikant kaukę, sudarytą iš dvejetainės skaičių aibės reikšmių, kadangi sudauginus neurono išvesties reikšmę su 0 šis neuronas yra ištrinamas. Taikant iškritimo strategiją nauja aibė yra sudaroma atsitiktiniu būdu kiekvieno mokymosi žingsnio metu.

3.6.2 Ankstyvas stabdymas

Apmokant ANN ilgesnį mokymosi laiką galima pastebėti, kad nuostolių funkcijos reikšmė mokymosi proceso metu tolydžiai mažėja, tačiau paklaida su patikros duomenų rinkiniu po kurio laiko pradeda kilti. Norint to išvengti galima taikyti ankstyvo stabdymo (angl. *early stopping*) strategiją, kurios metu pasirenkami ne galutiniai modelio parametrai, gauti mokymosi proceso gale, o parametrai, su kuriais modelis pasiekė mažiausią nuostolių funkcijos reikšmę su patikros duomenų rinkiniu. Įprastai tai atliekama apmokant modelį N epochų (epocha yra viena mokymosi algoritmo iteracija per mokymosi duomenų rinkinį), po kiekvienos epochos išsaugant modelio parametrus ir apskaičiuojant nuostolių funkcijos reikšmę, gautą su patikros duomenų rinkiniu. Mokymosi procesui pasibaigus pasirenkami geriausius rezultatus pasiekusio modelio parametrai.

4 Eksperimentai

4.1 Pasiruošimas eksperimentams

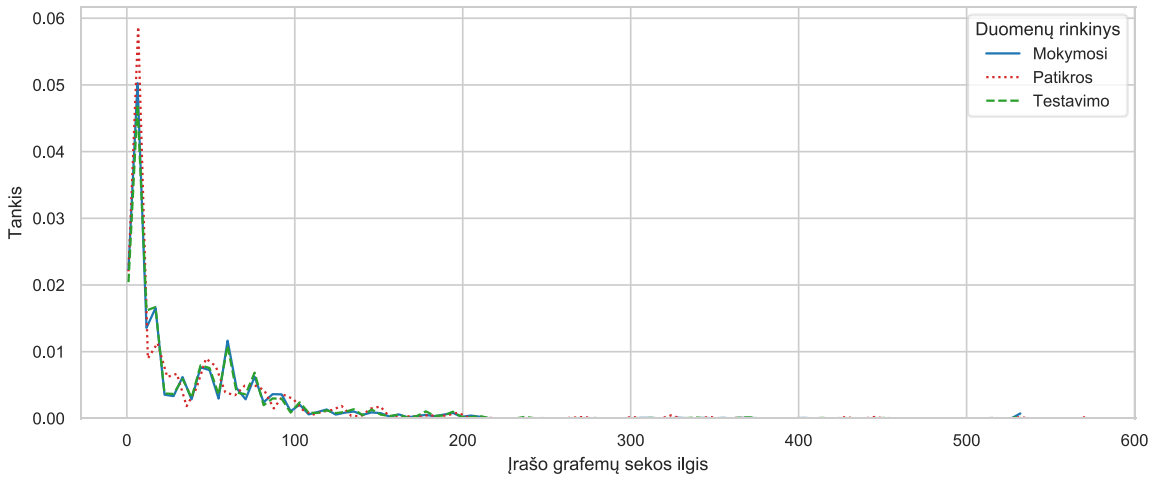
Šiame skyriuje pateiktas duomenų rinkinio apdorojimas ir mokymosi algoritmas.

4.1.1 Mokymosi, patikros ir testavimo rinkinių sudarymas

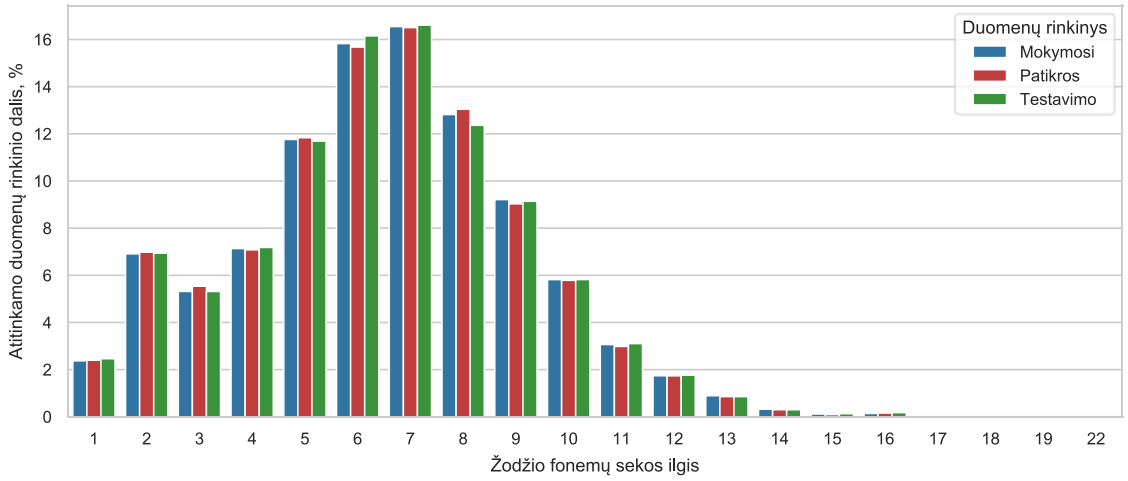
Vadovaujantis pasauline praktika turimas duomenų garsynas atsitiktiniu būdu buvo padalintas į tris dalis:

- **Mokymosi** duomenų rinkinį (60 % viso garsyno įrašų, 51 val. 57 min. anotuotų įrašų, 244860 žodžių), kuris buvo panaudotas modelio apmokymui.
- **Testavimo** duomenų rinkinį (30 % viso garsyno įrašų, 25 val. 31 min. anotuotų įrašų, 120387 žodžių), kuriuo buvo įvertintas galutinis modelio tikslumas.
- Duomenų rinkinį, skirtą modelio **patikrai** (10 % viso garsyno įrašų, 8 val. 41 min. anotuotų įrašų, 40841 žodžių), su kurio pagalba buvo ieškoma aukštus rezultatus galinčio pasiekti modelio parametrų.

Įrašų ilgių pasiskirstymas pagal juos sudarančių grafemų sekų ilgius yra pateiktas 4.1 pav. Iš duoto grafiko matyti, kad duomenų rinkiniai buvo padalinti labai panašiai, t. y. visuose rinkiniuose galima matyti tokį pat grafemų ilgių pasiskirstymą. Tą patvirtina ir 4.2 pav. grafikas, kuriame pateiktas atskirų žodžių ilgių pasiskirstymas tarp mokymosi, patikros ir testavimo rinkinių.



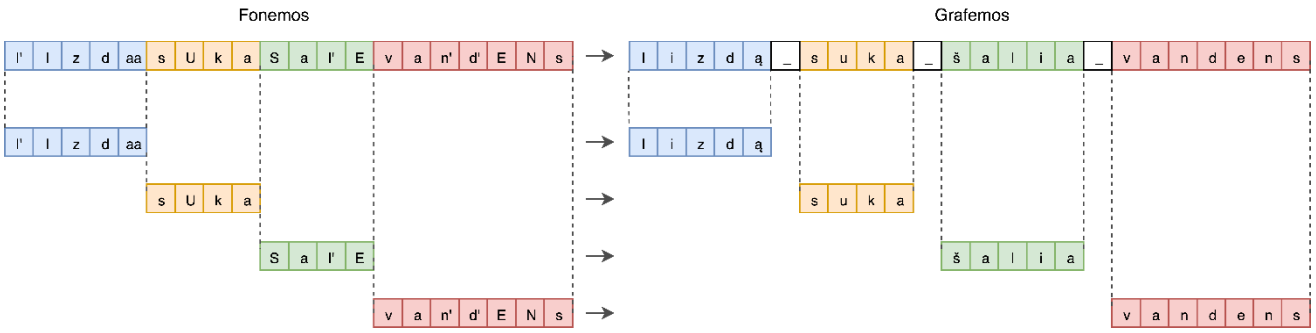
4.1 pav. Įrašų ilgių pasiskirstymas skirtinguose duomenų rinkiniuose



4.2 pav. Žodžių ilgių pasiskirstymas skirtinguose duomenų rinkiniuose.

Eksperimentų atlikimui duomenų rinkiniai buvo apdoroti: 1) izoliuojant atskirus žodžius ir juos atitinkančias fonemų sekas; 2) sudarant iš keleto žodžių susidedančias grafemų ir jas atitinkančių fonemų sekas.

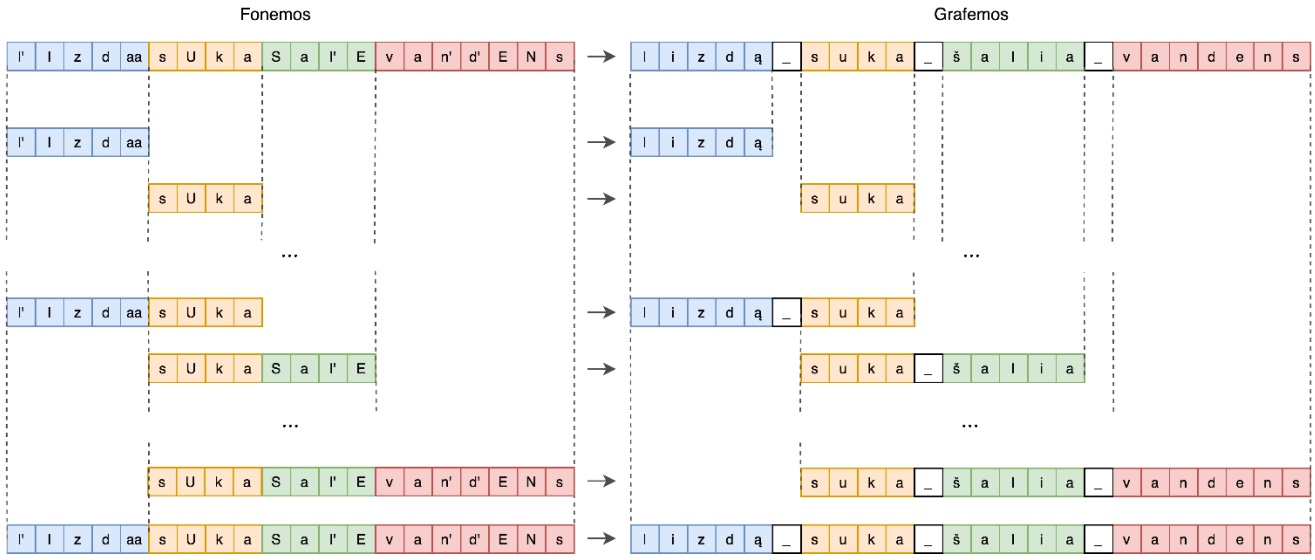
Izoliuotų žodžių rinkinio sudarymo principas parodytas 4.3 pav. Eksperimentai su tokio tipo rinkiniu pasirinkti dėl jų paprastumo – modeliui pakanka išmokti versti fonemų sekas grafemų sekomis. Eksperimentų su tokiu rinkiniu tikslas yra patikrinti, ar tokie modeliai geba rasti ryšį tarp duotų fonemų ir jas atitinkančių grafemų sekų.



4.3 pav. Izoliuotų žodžių fonemų sekų sudarymo pavyzdys

Sudėtingesni eksperimentai su iš keleto žodžių sudarytų fonemų (ir grafemų) sekų buvo atlikti vėliau. Jų paskirtis – modeliuoti tikrus šnekos atpažinimo duomenis, kuriuose ribos tarp atskirų žodžių yra labai blankios. Tokia užduotis yra sudėtingesnė, kadangi modeliui reikia mokytis ne tik naudojant daug ilgesnes sekas (ilgiausias žodis rinkinyje turi tik 24 grafemas, tuo tarpu ilgiausias įrašas 558 grafemas), bet taip pat ir „susidaryti“ kalbos modelį, kuris leistų teisingai atpažinti individualius žodžius sekose.

Tokio tipo rinkinio sudarymo pavyzdys pateiktas 4.4 pav. Kiekvienas įrašas rinkinyje buvo apdorojamas suskaidant jį į atskirus žodžius (grafemų sekas) ir jas atitinkančias fonemų sekas. Tuomet iš turimos sekų aibės buvo sudaromos makro-sekos, jungiant su iš eilės einančiais žodžiais susietas sekas, tokiu būdu iš kiekvieno įrašo sudarant $N!$ makro-sekų (N yra atskirų žodžių skaičius įraše).



4.4 pav. Makro-sekų sudarymo iš fonemų sekų pavyzdys

4.1.2 Pradinis sekų apdorojimas

Nepriklausomai nuo duomenų rinkinio tipo visos grafemų ir fonemų sekos buvo papildytos sekos pabaigos simboliais, kurie reikalingi užkoduotojo-dekoduotojo tipo architektūrų modeliams.

Apmokymo per prievartą metodo realizacijai grafemų sekos buvo dubliuojamos, sukuriant mokytojo sekas, t. y. sekas, kurios veikia kaip išorinis signalas dekoduotojo architektūrai. Kiekviena mokytojo seka buvo papildyta specialiu sekos pradžios simboliu, kurio paskirtis – nurodyti dekoduotojo daliai pradėti išvesties sekos generavimą.

4.1.3 Sekų „paminkštinimas“

Apmokant dirbtinius neuroninius tinklus buvo pastebėta, kad priklausomai nuo to, kaip duomenys yra perduodami į modelį ir kaip dažnai yra keičiami dirbtinių neuronų svoriai, modelio mokymosi greitis ir kokybė gali kisti. Yra skiriami trys apmokymo tipai: 1) mokymasis naudojant pilnas partijas; 2) interaktyvus mokymasis; 3) mokymasis naudojant mini-partijas.

Mokymosi naudojant pilnas partijas metu (angl. *full-batch learning*) dirbtinių neuronų svoriai yra keičiami vieną kartą per mokymosi epochą:

$$\mathcal{L}(\Theta) = \sum_{t=0}^{T_{train}-1} \mathcal{L}_{train}(\Theta), \tag{34}$$

čia $\mathcal{L}$ yra nuostolių funkcija, Θ modelio parametrai (visi dirbtinių neuronų svoriai), T_{train} mokymosi pavyzdžių skaičius mokymosi rinkinyje, $\mathcal{L}_{train}$ – nuostolių funkcija 1 mokymosi pavyzdžio atžvilgiu [38].

Toks mokymosi metodas yra paprasčiausias, tačiau nėra optimalus – nors apskaičiuotas gradientas nurodo globalaus minimumo kryptį, toks mokymasis reikalauja didelių kompiuterio resursų, o tai ne visados yra įmanoma (pvz., turint didelę duomenų imtį). Siekiant to išvengti dažniau pasitelkiami stochastinio gradientinio nusileidimo būdai: interaktyvus mokymasis ir mokymasis naudojant mini-partijas.

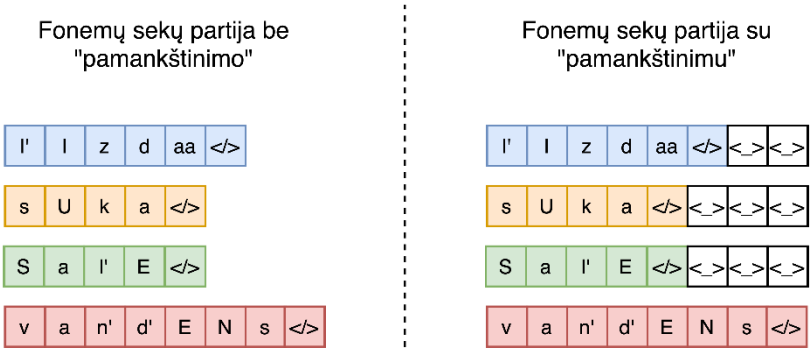
Interaktyvus mokymasis (angl. *online learning*) yra priešingas anksčiau aptartam mokymuisi naudojant pilnas partijas. Tokio mokymosi metu svoriai yra keičiami po kiekvieno mokymosi pavyzdžio apdorojimo.

Mokymasis naudojant mini-partijas (angl. *mini-batch learning*) yra daug efektyvesnis, tačiau mažiau stabilus [38]. Tokio mokymosi proceso metu visi duomenų rinkinyje esantys įrašai yra sugrupuojami į mini-partijas (pvz., po 32 įrašus). Mokymosi metu modeliai apskaičiuoja vidutinį gradientą svorių korekcijoms naudodami visus įrašus partijoje ir tokiu būdu gaudami gradiento artinį.

Nors metodui didelį poveikį daro savybės, kuriomis pasižymi duomenys, ir pasirinktas partijos dydis, , praktikoje jis naudojamas dažnai, kadangi leidžia apmokyti modelius greičiau nei naudojant pilnas partijas.

Dažna problema, kylanti apmokant RNN paremtus modelius, yra nevienodas sekų ilgis. Dėl realizacijos apribojimų norint naudoti skirtingų ilgių sekas vienoje partijoje (jei partijos dydis > 1) reikia sekų ilgius suvienodinti iki ilgiausios sekos ilgio. Tai galima atlikti dviem būdais: 1) grupuojant sekas pagal jų ilgius; 2) papildant trumpesnes sekas specialiais „paminkštinimo“ simboliais (žr. 4.5 pav.)

Grupavimas pagal sekų ilgius yra itin paprastas, tačiau ne pats tinkamiausias sprendimas, kadangi modelis gali pradėti prisitaikyti prie duomenų pradėdamas mokytis iš sekos ilgio. Kita vertus, didelė sekų ilgio variacija partijoje reikalauja daug skaičiavimo resursų paskirti sekų papildymui „paminkštinimu“.



4.5 pav. Sekų paminkštinimo pavyzdys (</> sekos pabaigos, < _ > „paminkštinimo“ simboliai)

Siekiant sušvelninti abiejų metodų neigiamas puses šiame darbe eksperimentai atlikti grupuojant sekas pagal jų ilgius į grupes, kuriose sekų ilgiai negali skirtis daugiau kaip 4 simboliais. Tokiu būdu partijose esančios sekos skiriasi ne tik turiniu, bet ir ilgiais, o resursai, reikalingi sekų „paminkštinimui“, yra minimalūs.

Svarbu atkreipti dėmesį, kad grafemų ir fonemų sekos yra susietos, t. y. grupavimas pagal sekų ilgius turi būti atliekamas tik vienos (grafemų arba fonemų) sekos atžvilgiu. Šiame darbe nuspręsta sekas grupuoti fonemų atžvilgiu.

4.1.4 Vienetinis kodavimas

Sekas sudarančias grafemas ir fonemas galima priskirti kategoriniam duomenų tipui. Unikalių fonemų (ir grafemų) reikšmių aibė yra iš anksto žinoma ir apibrėžiama fonemų sekų išgavimo proceso metu bei priklauso nuo duomenų rinkinio kalbos. Prieš perduodant kategorinius duomenis į statistinį modelį, juos yra įprasta apdoroti vienetinio kodavimo (angl. *one-hot encoding*) būdu.

Tokio kodavimo metu kiekvienas sekos $X = (x_1, x_2, \dots x_T)$ elementas x_i yra paverčiamas į vektorių $\boldsymbol{z} \in R^K$, kur K yra unikalių elementų skaičius visame duomenų rinkinyje (šio duomenų rinkinio kontekste tai yra unikalių fonemų arba grafemų skaičius). Vektorius $\boldsymbol{z}$ yra užpildomas 0, nustatant reikšmę k indekse į 1, (k unikalaus elemento įrašas žodyne).

4.1.5 Modeliai eksperimentams

Eksperimentų atlikimui pasirinkti du užkoduotojo-dekoduotojo tipo modeliai.

Pirmasis modelis be dėmesio mechanizmo yra paremtas Sutskever ir kt. [46] pristatytu „sekos į seką“ (angl. *Sequence to Sequence*) modeliu, skirtu mašininiam vertimui. Modelio užkoduotojo ir dekoduotojo dalims panaudotos LSTM ląstelės. Eksperimentuose tokio tipo modelių konfigūracijoms duotas *Seq2Seq* pavadinimas.

Antrasis modelis yra užkoduotojo-dekoduotojo tipo modelis su Luong ir kt. [52] pasiūlytu dėmesio mechanizmu ir dvikrypčių LSTM ląstelių sluoksniais užkoduotojo ir dekoduotojo dalyse. Tokio tipo modeliai eksperimentuose pavadinti *Attention* vardu.

Šių modelių mokymui buvo naudojama žemiau pateikta nuostolių funkcija $\mathcal{L}$:

$$\mathcal{L} = -\sum_{t=0}^{T'} \log p(y_t | y_1, \dots, y_{t-1}, X) \tag{35}$$

čia $\mathcal{L}$ - modelio nuostolių funkcija, $X = (x_1, \dots, x_T)$ – fonemų seka, $Y = (y_1, \dots, y_{T'})$ – fonemų seką atitinkanti grafemų seka.

Modeliai apmokyti stochastinio gradientinio nusileidimo modifikacija – Adam optimizacijos algoritmu [54], kuris apskaičiuoja prisitaikančius parametrų mokymosi greičius pasinaudodamas gradiento pirmuoju ir antruoju momentais.

Siekiant pagreitinti mokymosi procesą, visi eksperimentuose naudoti modeliai buvo apmokyti naudojant apmokymo per prievartą metodą (žr. 3.4.1 skyrių).

4.2 Eksperimentai su izoliuotais žodžiais

Pirmojo eksperimento tikslas – įvertinti, ar užkoduotojo-dekoduotojo tipo architektūra paremti modeliai gali išmokti efektyviai versti pavienių (izoliuotų) žodžių fonemų sekas į atitinkamas grafemų sekas.

Šio eksperimento metu abu ankstesniame skyrelyje aptarti modeliai buvo apmokyti naudojant 4.1.1 skyriuje aprašytą mokymosi duomenų rinkinį.

4.2.1 Įprastas apmokymas

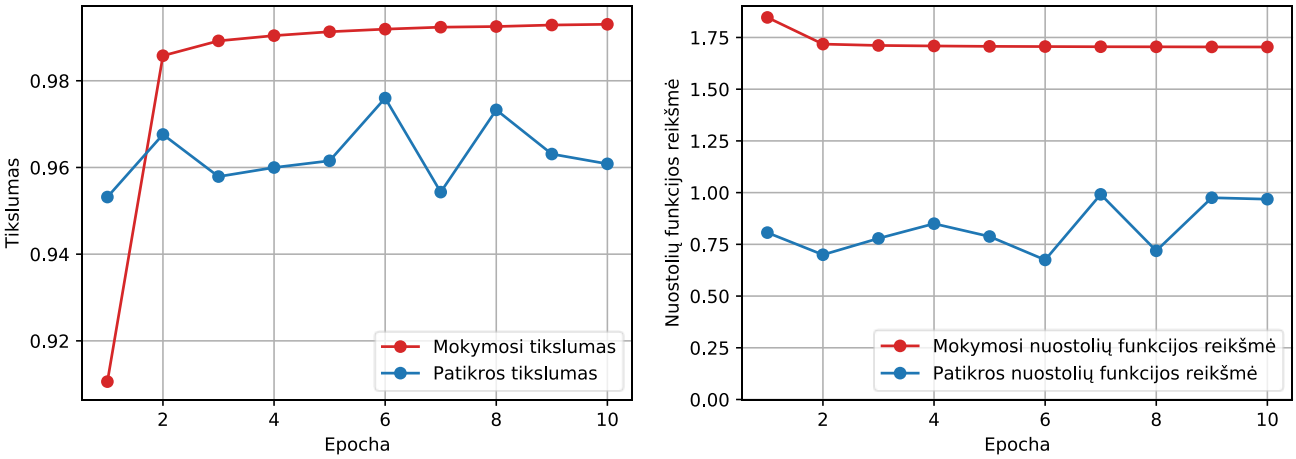
Pirmajai eksperimento daliai, kuria buvo siekiama įvertinti, kokia modelio konfigūracija veikia geriau, pasirinkta keletas skirtingų *Seq2Seq* ir *Attention* tipo modelių konfigūracijų, kurios skyrėsi dirbtinių neuronų sluoksnių skaičiumi ir LSTM ląstelių kiekiu. Kiekvieno modelio mokymosi procesas truko 10 epochų (10 iteracijų per mokymosi duomenų imtį), po kiekvienos epochos patikrinant modelio tikslumą patikros duomenų rinkiniu. Mokymuisi naudotas 0,01 mokymosi žingsnis.

Seq2Seq tipo modelio skirtingų konfigūracijų geriausi pasiekti rezultatai yra pateikti 4.1 lentelėje. Lentelėje esanti modelio pavadinimo konfigūracija nurodo modelio tipą, užkoduotojo ir dekoduotojo dalyse esančių sluoksnių skaičių ir LSTM ląstelių skaičių kiekviename sluoksnyje, pvz., *Seq2Seq-2x32* – modelis be dėmesio mechanizmo su dviem po 128 LSTM ląsteles turinčiais dirbtinių neuronų sluoksniais užkoduotojo dalyje ir su dviem po 128 LSTM ląsteles turinčiais dirbtinių neuronų sluoksniais dekoduotojo dalyje. Tokio modelio konfigūracijos pavadinimo formato nuspręsta laikytis ir kitose lentelėse.

4.1 lentelė Įprasto apmokymo metu *Seq2Seq* tipo modelių geriausi pasiekti rezultatai

Modelio pavadinimas	**LSTM sluoksnių skaičius**	**LSTM ląstelių skaičius sluoksnyje**	**Epocha**	**Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu**	**Tikslumas su mokymosi duomenų rinkiniu**	**Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu**	**Tikslumas su patikros duomenų rinkiniu**
Seq2Seq-2x32	2	32	5	1,718	0,986	0,835	0,954
Seq2Seq-2x64	2	64	6	1,706	0,992	0,675	**0,976**
Seq2Seq-2x128	2	128	4	1,704	0,993	0,661	0,966
Seq2Seq-2x256	2	256	3	1,713	0,989	0,865	0,947
Seq2Seq-3x32	3	32	10	1,713	0,989	0,823	0,944
Seq2Seq-3x64	3	64	10	1,706	0,992	0,721	0,958
Seq2Seq-3x128	3	128	7	1,716	0,987	0,838	0,938
Seq2Seq-3x256	3	256	3	1,971	0,763	4,430	0,488
Seq2Seq-4x32	4	32	10	1,762	0,967	1,192	0,876
Seq2Seq-4x64	4	64	7	1,832	0,923	2,067	0,777
Seq2Seq-4x128	4	128	8	1,785	0,952	1,247	0,862
Seq2Seq-4x256	4	256	2	1,751	0,968	0,862	0,928
Seq2Seq-5x32	5	32	8	1,763	0,965	1,525	0,850
Seq2Seq-5x64	5	64	8	1,958	0,868	1,166	0,814
Seq2Seq-5x128	5	128	4	2,091	0,685	3,910	0,487
Seq2Seq-5x256	5	256	4	2,862	0,328	2,434	0,348

Iš gautų rezultatų matyti, kad geriausią tikslumą su patikros duomenų imtimi pasiekia modelis su 2 LSTM ląstelių sluoksniais po 64 ląsteles užkoduotojo ir dekoduotojo dalyse (*Seq2Seq-2x64*). Įdomu, kad modelis (ir dauguma kitų konfigūracijų) geriausius rezultatus pasiekia ne po 10 mokymosi epochų, o daug anksčiau. 4.6 pav. pateiktas modelio mokymosi grafikas, iš kurio galima matyti, kad nors modelio tikslumas didėja su kiekviena mokymosi epocha, patikros tikslumas sumažėja. Panašią tendenciją galima pastebėti ir su modelio nuostolio funkcijos reikšme: nors bendra modelio paklaida pamažu mažėja su kiekviena epocha, nuostolių funkcijos reikšmė, apskaičiuota su patikros duomenų imtimi, po truputį didėja. Toks efektas rodo modelių persimokymą, kurio būtų galima išvengti pritaikius reguliarizavimo strategijas, pvz., iš anksto stabdant mokymąsi, kai tikslumas su patikros duomenų rinkiniu pradeda kristi arba tam tikrą epochų skaičių nekinta.



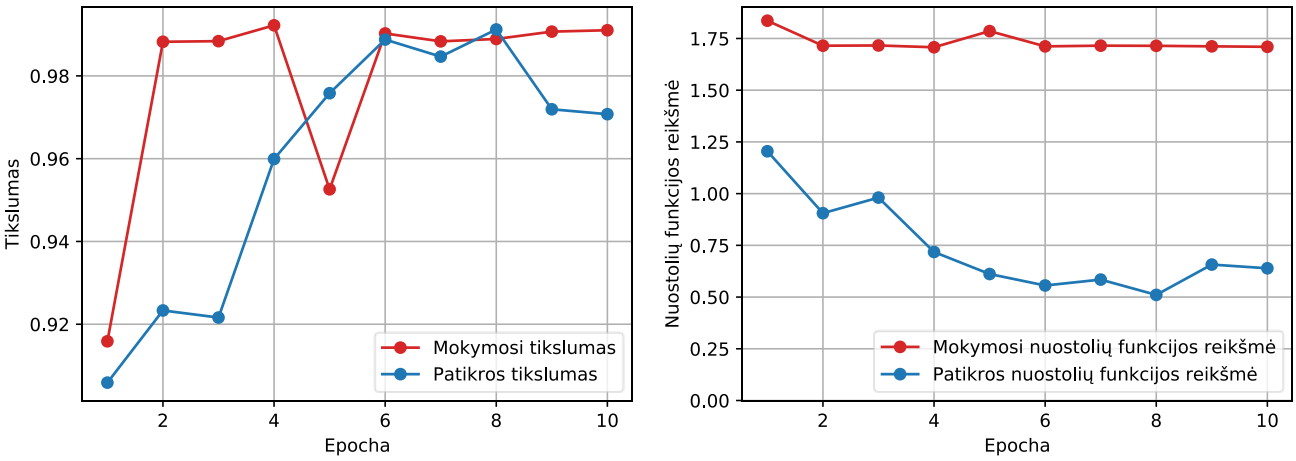
4.6 pav. *Seq2Seq-2x64* tikslumo ir nuostolių funkcijos reikšmių grafikai

Modelio su dėmesiu geriausi pasiekti rezultatai yra pateikti 4.2 lentelėje, kurioje galima matyti, kad didžiausią tikslumą su patikros duomenų rinkiniu pasiekęs modelis (*Attention-4x32*, tikslumas – 0,991) lenkia geriausią *Seq2Seq* tipo modelį (*Seq2Seq-2x64*, tikslumas – 0,976). Šis modelis yra gilesnis ir turi po keturis 32 LSTM ląstelių sluoksnius užkoduotojo ir dekoduotojo dalyse.

Geriausio modelio su dėmesio mechanizmu tikslumo ir nuostolių funkcijos reikšmių grafikai, gauti mokymosi metu, pateikti 4.7 pav. Palyginus su *Seq2Seq-2x64* mokymosi grafiku (žr. 4.6 pav.), galima teigti, kad *Attention-4x32* mokosi daug ilgiau ir tik po aštuntos epochos jo efektyvumas pradeda kristi.

4.2 lentelė Skirtingų konfigūracijų modelių su dėmesio mechanizmu geriausi pasiekti rezultatai

Modelio pavadinimas	**LSTM sluoksnių skaičius**	**LSTM ląstelių skaičius sluoksnyje**	**Epocha**	**Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu**	**Tikslumas su mokymosi duomenų rinkiniu**	**Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu**	**Tikslumas su patikros duomenų rinkiniu**
Attention-2x32	2	32	5	1,702	0,994	1,015	0,949
Attention-2x64	2	64	3	1,806	0,947	0,651	0,953
Attention-2x128	2	128	2	1,788	0,945	0,918	0,943
Attention-2x256	2	256	9	12,165	0,103	11,688	0,155
Attention-3x32	3	32	9	1,700	0,995	0,638	0,975
Attention-3x64	3	64	6	1,705	0,992	0,835	0,938
Attention-3x128	3	128	3	1,710	0,990	0,685	0,953
Attention-3x256	3	256	2	1,772	0,963	0,592	0,973
Attention-4x32	4	32	8	1,714	0,989	0,510	**0,991**
Attention-4x64	4	64	2	1,742	0,974	1,531	0,904
Attention-4x128	4	128	2	1,718	0,987	0,741	0,940
Attention-4x256	4	256	3	2,386	0,608	2,265	0,586
Attention-5x32	5	32	10	1,706	0,993	0,577	0,980
Attention-5x64	5	64	3	1,710	0,990	0,552	0,983
Attention-5x128	5	128	3	1,730	0,980	0,603	0,960
Attention-5x256	5	256	5	1,774	0,969	1,453	0,909



4.7 pav. *Attention-4x32* tikslumo ir nuostolių funkcijos reikšmių grafikai

Remiantis pirmąja eksperimento dalimi, galima teigti, kad neretai gilesni ir dažnai platesni modeliai nepasiekia aukštų mokymosi rezultatų. Sunku atsakyti, kodėl didesni modeliai nepasižymi aukštu mokymosi tikslumu.

Šios eksperimento dalies tikslas buvo įvertinti, kokią įtaką mokymosi rezultatams gali daryti neuronų iškritimo (žr. 3.6.1 skyrių) strategijos taikymas apmokant modelius. Eksperimentas atliktas naudojant tokias pat konfigūracijas kaip ir 4.2.1 skyriuje.

Eksperimento metu naudotos standartinės dirbtinių neuronų iškritimo tikimybių reikšmės, pasiūlytos Goodfellow ir kt. [45] (tikimybė palikti pirmųjų užkoduotojo ir dekoduotojo ANN sluoksnių neuronus – 0,8; tolimesnių, paslėptų sluoksnių neuronus – 0,5).

Seq2Seq ir *Attention* modelių rezultatai atitinkamai pateikti 4.3 lentelėje ir 4.4 lentelėje.

4.3 lentelė *Seq2Seq* tipo modelių su dirbtinių neuronų iškritimu geriausi pasiekti rezultatai

Modelio pavadinimas	**LSTM sluoksnių skaičius**	**LSTM ląstelių skaičius sluoksnyje**	**Epocha**	**Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu**	**Tikslumas su mokymosi duomenų rinkiniu**	**Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu**	**Tikslumas su patikros duomenų rinkiniu**
Seq2Seq-2x32	2	32	6	2,080	0,760	1,212	0,803
Seq2Seq-2x64	2	64	4	1,951	0,830	0,848	0,903
Seq2Seq-2x128	2	128	5	1,853	0,886	0,751	0,926
Seq2Seq-2x256	2	256	9	1,846	0,892	0,609	**0,965**
Seq2Seq-3x32	3	32	9	2,148	0,718	1,251	0,786
Seq2Seq-3x64	3	64	9	1,968	0,823	0,834	0,896
Seq2Seq-3x128	3	128	4	1,949	0,829	0,672	0,949
Seq2Seq-3x256	3	256	1	2,352	0,575	1,383	0,766
Seq2Seq-4x32	4	32	10	2,375	0,587	1,867	0,582
Seq2Seq-4x64	4	64	3	2,385	0,568	1,649	0,633
Seq2Seq-4x128	4	128	2	2,240	0,658	1,633	0,683
Seq2Seq-4x256	4	256	1	2,809	0,264	3,664	0,224
Seq2Seq-5x32	5	32	10	2,582	0,428	2,417	0,374
Seq2Seq-5x64	5	64	3	2,785	0,341	2,123	0,354
Seq2Seq-5x128	5	128	1	3,164	0,181	2,962	0,196
Seq2Seq-5x256	5	256	1	2,856	0,260	2,330	0,338

4.4 lentelė *Attention* tipo modelių su dirbtinių neuronų iškritimu geriausi pasiekti rezultatai

Modelio pavadinimas	**LSTM sluoksnių skaičius**	**LSTM ląstelių skaičius sluoksnyje**	**Epocha**	**Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu**	**Tikslumas su mokymosi duomenų rinkiniu**	**Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu**	**Tikslumas su patikros duomenų rinkiniu**
Attention-2x32	2	32	2	1,821	0,910	1,734	0,866
Attention-2x64	2	64	8	1,813	0,917	0,889	0,904
Attention-2x128	2	128	1	1,925	0,848	0,557	**0,981**
Attention-2x256	2	256	1	2,176	0,689	1,161	0,837
Attention-3x32	3	32	9	1,810	0,919	0,809	0,932
Attention-3x64	3	64	8	1,773	0,942	1,105	0,886
Attention-3x128	3	128	1	1,918	0,847	0,811	0,904
Attention-3x256	3	256	3	26,226	0,016	9,800	0,039
Attention-4x32	4	32	7	1,898	0,856	1,728	0,884
Attention-4x64	4	64	9	1,957	0,815	0,980	0,901
Attention-4x128	4	128	2	1,839	0,900	0,890	0,906
Attention-4x256	4	256	1	2,945	0,299	2,423	0,335
Attention-5x32	5	32	9	1,903	0,854	1,320	0,882
Attention-5x64	5	64	9	1,901	0,872	1,355	0,861
Attention-5x128	5	128	1	2,044	0,773	0,832	0,916
Attention-5x256	5	256	1	16,808	0,103	21,575	0,058

Iš gautų rezultatų matyti, kad modeliams be dėmesio dirbtinių neuronų iškritimas beveik nepadėjo – tik trijų modelių konfigūracijos viršijo ankstesnio eksperimento metu gautus rezultatus. Tuo tarpu modeliai su dėmesiu pasirodė geriau (net keturiolika konfigūracijų veikia geriau su dirbtinių neuronų iškritimu), tačiau nei viena modelio konfigūracija nesugebėjo pranokti pirmojo eksperimento metu gauto geriausio modelio rezultatų.

Vis dėlto tikėtina, kad parinkus kitokias dirbtinių neuronų iškritimo tikimybes galima būtų pagerinti modelių veikimo rezultatus. Tačiau reikia įvertintinti tai, kad įmanomų iškritimo tikimybių kombinacijų skaičius yra labai didelis ir dėl didelės šio hiperparametro paieškų erdvės šį uždavinį paliksime ateičiai.

4.2.2 Apmokymas atbulomis sekomis

3.4.2 skyriuje buvo pristatyta Sutskever ir kt. [46] pasiūlyta apmokymo, panaudojant atbulas įvesties sekas, strategija. Remiantis autorių patirtimi nuspręsta patikrinti, kaip mokymasis naudojant atbulas fonemų sekas gali paveikti modelių rezultatus.

Eksperimentas atliktas naudojant konfigūracijas, pateiktas 4.2.1 skyriuje. Kadangi šio eksperimento metu mokymui buvo naudojamos atbulos fonemų sekos, grafemų sekos, naudotos nuostolių funkcijos reikšmėms ir tikslumo metrikoms apskaičiuoti, taip pat buvo apsuktos.

Kaip matyti iš *Seq2Seq* tipo modelių rezultatų (4.5 lentelė), atbulų sekų panaudojimas leidžia padidinti modelių tikslumą: įprasto mokymosi metu (žr. 4.2.1 skyrių) geriausias modelis pasiekė 0,976 tikslumą su patikros duomenų rinkiniu, o naudojant atbulas sekas – net 0,991. Verta atkreipti dėmesį į tai, kad naudojant atbulas sekas dauguma modelių geriausią tikslumą pasiekė paskutinėmis mokymosi epochomis (8-10

epochos; visi modeliai buvo apmokyti po 10 epochų). Galima daryti prielaidą, kad mokant modelius ilgesnį laiką įmanoma pasiekti dar geresnių rezultatų.

Palyginus su anksčiau gautais rezultatais, modeliai su dėmesio mechanizmu, apmokyti naudojant atbulas sekas, veikia šiek tiek geriau (aukščiausias modelio pasiektas tikslumas naudojant atbulas sekas – 0,993; 0,991 – geriausias ankstesnių bandymų metu pasiektas tikslumas). Kitaip nei modeliai be dėmesio mechanizmo, šie modeliai dažniausiai geriausius rezultatus pasiekia po keleto mokymosi epochų.

4.5 lentelė *Seq2Seq* tipo modelių geriausi pasiekti rezultatai naudojant atbulas sekas

Modelio pavadinimas	**LSTM sluoksnių skaičius**	**LSTM ląstelių skaičius sluoksnyje**	**Epocha**	**Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu**	**Tikslumas su mokymosi duomenų rinkiniu**	**Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu**	**Tikslumas su patikros duomenų rinkiniu**
Seq2Seq-2x32	2	32	8	1,709	0,991	0,702	0,976
Seq2Seq-2x64	2	64	10	1,703	0,994	0,625	0,986
Seq2Seq-2x128	2	128	9	1,701	0,995	0,534	**0,991**
Seq2Seq-2x256	2	256	3	1,711	0,990	0,703	0,973
Seq2Seq-3x32	3	32	8	1,713	0,989	0,714	0,960
Seq2Seq-3x64	3	64	10	1,709	0,991	0,596	0,982
Seq2Seq-3x128	3	128	9	1,711	0,990	0,612	0,978
Seq2Seq-3x256	3	256	10	1,785	0,954	2,350	0,778
Seq2Seq-4x32	4	32	8	1,770	0,964	0,995	0,897
Seq2Seq-4x64	4	64	10	1,783	0,955	2,047	0,810
Seq2Seq-4x128	4	128	7	1,732	0,980	0,940	0,939
Seq2Seq-4x256	4	256	8	1,966	0,812	4,041	0,451
Seq2Seq-5x32	5	32	8	1,745	0,974	0,989	0,920
Seq2Seq-5x64	5	64	7	1,806	0,943	1,657	0,812
Seq2Seq-5x128	5	128	7	2,175	0,685	6,832	0,217
Seq2Seq-5x256	5	256	7	3,153	0,172	2,939	0,180

4.6 lentelė *Attention* tipo modelių geriausi pasiekti rezultatai naudojant atbulas sekas

Modelio pavadinimas	LSTM sluoksnių skaičius	LSTM ląstelių skaičius sluoksnyje	Epocha	Nuostolių funkcijos reikšmė su mokymosi duomenų rinkiniu	Tikslumas su mokymosi duomenų rinkiniu	Nuostolių funkcijos reikšmė su patikros duomenų rinkiniu	Tikslumas su patikros duomenų rinkiniu
Attention-2x32	2	32	8	1,701	0,994	0,552	0,989
Attention-2x64	2	64	2	1,710	0,990	0,514	**0,993**
Attention-2x128	2	128	1	1,759	0,959	0,530	0,989
Attention-2x256	2	256	1	2,438	0,654	7,113	0,111
Attention-3x32	3	32	5	1,703	0,994	0,527	0,985
Attention-3x64	3	64	3	1,707	0,992	0,538	0,988
Attention-3x128	3	128	1	1,787	0,942	0,523	0,990
Attention-3x256	3	256	5	1,832	0,959	0,691	0,969
Attention-4x32	4	32	6	1,702	0,994	0,605	0,965
Attention-4x64	4	64	6	1,706	0,992	0,820	0,949
Attention-4x128	4	128	4	1,711	0,990	0,508	0,990
Attention-4x256	4	256	4	3,170	0,311	2,523	0,302
Attention-5x32	5	32	6	1,707	0,992	0,630	0,967
Attention-5x64	5	64	5	1,706	0,992	0,608	0,976
Attention-5x128	5	128	3	1,750	0,973	0,797	0,921
Attention-5x256	5	256	5	2,588	0,496	2,297	0,457

4.2.3 Tikslumas naudojant testavimo imtį

Paskutinis eksperimentas su izoliuotų žodžių atpažinimo duomenų rinkiniu atliktas siekiant patikrinti modelio tikslumą su testavimo duomenų imtimi. Šiam eksperimentui atlikti atrinktos dvi geriausius rezultatus pasiekusios konfigūracijos iš eksperimentų, aprašytų 4.2.1-4.2.2 skyriuose: *Seq2Seq-2x128* ir *Attention-2x64*. Šalia nuostolių funkcijos reikšmės ir tikslumo įverčio taip pat pateikiami ir vidutinis Levenšteino atstumas, ir Levenšteino atstumo mediana.

Modelių rezultatai su testavimo duomenų rinkiniu pateikti 4.7 lentelėje. Modelis be dėmesio (*Seq2Seq-2x128*) sugebėjo pasiekti 0,991 tikslumo įvertį (0,991 su patikros duomenų rinkiniu), o modelis su dėmesio mechanizmu – 0,993. Levenšteino atstumo mediana ir vidutinis Levenšteino atstumas patvirtina, kad modeliai geba atpažinti žodžius iš jų fonemų sekų ir klaidas daro labai retai.

4.7 lentelė Modelių rezultatai su testavimo duomenų rinkiniu

Modelio konfigūracijos pavadinimas	Nuostolių funkcijos reikšmė	Tikslumas	Vid. Levenšteino atstumas	Levenšteino atstumo mediana
Seq2Seq-2x128	0,536	0,991	0,043	0
Attention-2x64	0,515	0,993	0,028	0

4.8 lentelėje pateikta keletas modelių sugeneruotų žodžių pavyzdžių.

4.8 lentelė Modelių sugeneruotų žodžių pavyzdžiai

Seq2Seq-2x128 **modelio sugeneruotas žodis**	***Attention-2x64*** **modelio sugeneruotas žodis**	**Tikrasis žodis**
išsaugok	išsaugok	išsaugok
technologijos	technologijos	technologijos
septyniasdešimties	septyniasdešimties	septyniasdešimties
aprašymas	aprašymas	aprašymas
stabdyk	stabdyk	stabdyk
žinduolių	žinduolių	žinduolių
tūkstantis	tūkstantis	tūkstantis
kriokliais	kriokliais	kriokliais

Apibendrinant eksperimento metu gautus rezultatus galima teigti, kad užkoduotojo-dekoduotojo tipo modeliai gali efektyviai išmokti ryšį tarp fonemų ir jas atitinkančių grafemų sekų ir gali pasiekti labai gerų rezultatų izoliuotų žodžių automatinio lietuvių šnekos atpažinimo iš fonemų sekų uždavinyje.

4.3 Eksperimentai su rišlia kalba

Antrasis eksperimentas atliktas siekiant patikrinti, kaip gerai modeliai gali išmokti versti fonemų sekas į atitinkamas grafemų sekas, kai sekose yra daugiau nei vienas žodis. Iš pirmo žvilgsnio paprastas, toks uždavinys yra daug sudėtingesnis, kadangi modeliui reikia ne tik atrasti ryšį tarp fonemų ir grafemų sekų, bet ir „susidaryti“ kalbos modelį, kuris leistų rasti žodžių pradžias ir pabaigas sekose.

Eksperimentui atlikti sudaryti 6 duomenų rinkiniai iš pradinio rinkinio formuojant makro-sekas 4.1.1 skyriuje aprašytu būdu. Kiekvienas duomenų rinkinys skyrėsi didžiausiu leistinu makro-sekos ilgiu T. Pasirinkta keletas skirtingų fonemų sekų ilgių: 8, 12, 16, 20, 24 ir 28.

Šiam eksperimentui atlikti buvo parinktos dvi geriausiai pirmojo eksperimento pasirodžiusios modelių konfigūracijos: *Seq2Seq-2x128* (modelis be dėmesio mechanizmo) ir *Attention-2x64* (modelis su dėmesio mechanizmu).

Modelio be dėmesio mechanizmo užkoduotojo ir dekoduotojo dalis sudaro 4, 128 LSTM ląstelių sluoksniai (2 sluoksniai užkoduotojo dalyje ir 2 sluoksniai dekoduotojo dalyje). Modelis su dėmesio mechanizmu yra tokio pat gylio (po 2 LSTM ląstelių sluoksnius užkoduotojo ir dekoduotojo dalyse) ir turi po 64 dvikrypčio tipo LSTM ląsteles kiekviename užkoduotojo ir dekoduotojo sluoksnyje. Remiantis rezultatais, gautais pirmojo eksperimento metu, nuspręsta *Seq2Seq-2x128* ir *Attention-2x64* modelius apmokyti ir ištestuoti naudojant atbulas fonemų sekas.

Siekiant sumažinti gradientų „sprogimo“ tikimybę mokymosi metu, mokymosi žingsnis apribotas iki 0,001 ir nustatytos mažiausios ir didžiausios gradiento reikšmės: atitinkamai -5 ir +5. Mokymuisi naudotos mini-partijos, sudarytos iš 32 sekų. Kiekvienas modelis apmokytas 20 epochų, parenkant geriausią modelio konfigūraciją pagal jo pasiektus rezultatus su patikros duomenų rinkiniu (žr. 3.6.2 skyrių).

Modelio be dėmesio mechanizmo *Seq2Seq-2x128* rezultatai su įvairaus ilgio fonemų sekomis pateikti 4.9 lentelėje, o tikslumo ir WER grafikai 4.8 pav. ir 4.9 pav. atitinkamai.

Visų pirma, galima pastebėti, kad modelis geba pasiekti labai aukštą mokymosi tikslumą nepriklausomai nuo fonemų sekos ilgio. Be to, nesunku nustatyti, kad skirtingų modelio konfigūracijų rezultatai su patikros ir testavimo duomenų rinkiniais yra labai panašūs, o tai leidžia daryti prielaidą, kad modelis geba gerai generalizuoti.

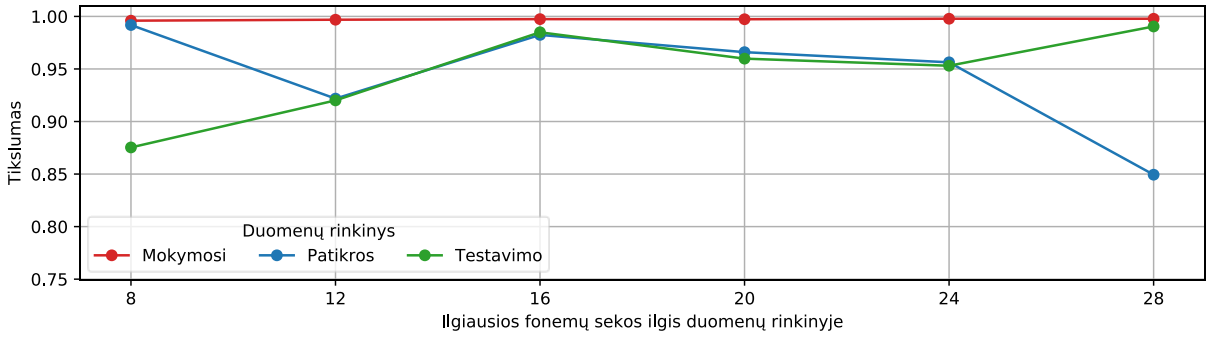
Analizuojant rezultatus išskiriami du kraštutiniai atvejai:

1) modelis, apmokytas su duomenų rinkiniu, kuriame ilgiausia fonemų seka turi tik 8 fonemas, pasiekė labai aukštą tikslumo įvertį patikros metu, tačiau santykinai mažą tikslumą su testavimo duomenų rinkiniu;
2) modelis, apmokytas su duomenų rinkiniu, kuriame ilgiausia fonemų seka turi 28 fonemas, pasiekė prasčiausią patikros tikslumą, bet sugebėjo pasiekti geriausią tikslumą su testavimo duomenų rinkiniu.

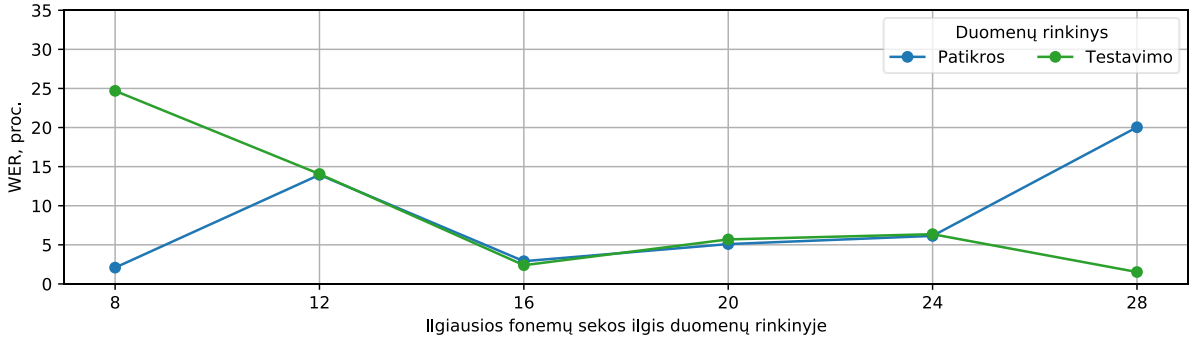
Verta atkreipti dėmesį į tai, kad modeliai, apmokyti su trumpesnėmis fonemų sekomis ($T = 8$ ar $T = 12$), pasižymi aukštesniu žodžių atpažinimo klaidos rodikliu. Taip yra todėl, kad trumpesnės fonemų sekos yra susietos su mažesniu žodžių skaičiumi išsireiškime (žr. 7.3 skyrių), todėl modelio padarytos klaidos yra daug svaresnės. Pavyzdžiui, klaidingai atpažinus vieną žodį keturių žodžių frazėje $WER = 0{,}25$, o suklydus vieną kartą dviejų žodžių sakinyje, žodžių atpažinimo klaidos rodiklis išauga iki 0,5. Tai galima pastebėti ir iš arčiau panagrinėjus rezultatus, gautus su $T = 8$ ir $T = 12$ duomenų rinkiniais: nors tikslumas gautas su patikros duomenų rinkiniu yra 0,07 (7 proc.), WER skiriasi 11.8 proc.

4.9 lentelė Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio fonemų sekomis, geriausi pasiekti rezultatai

Ilgiausios fonemų sekos ilgis duomenų rinkinyje (*T*)	**Epocha**	**Mokymosi tikslumas**	**Tikslumas su patikros duomenų rinkiniu**	**WER (%) su patikros duomenų rinkiniu**	**Tikslumas su testavimo duomenų rinkiniu**	**WER (%) su testavimo duomenų rinkiniu**
8	18	0,996	0,992	2,086	0,875	24,692
12	19	0,997	0,922	13,953	0,920	14,057
16	19	0,997	0,982	2,905	0,985	2,394
20	13	0,997	0,966	5,097	0,960	5,693
24	20	0,998	0,956	6,142	0,953	6,362
28	20	0,998	0,849	20,039	0,990	1,524



4.8 pav. Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio sekomis, tikslumo grafikas



4.9 pav. Modelio be dėmesio mechanizmo, apmokyto įvairaus ilgio sekomis, WER grafikas

4.10 lentelėje pateikti modelių patikros ir testavimo metu gautų tikslumo rezultatų pasikliautinieji intervalai, kurie įrodo, kad modelių pasiekti aukšti rezultatai nėra iškreipti. Verta atkreipti dėmesį į tai, kad modelį apmokant trumpesnėmis fonemų sekomis, gaunamas pasikliautinasis intervalas yra platesnis.

4.10 lentelė Modelio be dėmesio mechanizmo tikslumo įverčių pasikliautininieji intervalai

Ilgiausios fonemų sekos ilgis duomenų rinkinyje	**Patikros metu gauto tikslumo įverčio PI (0.95 pasikliovimo lygmuo)**	**Testavimo metu gauto tikslumo įverčio PI (0.95 pasikliovimo lygmuo)**	**Patikros metu gauto tikslumo įverčio PI (0.99 pasikliovimo lygmuo)**	**Testavimo metu gauto tikslumo įverčio PI (0.99 pasikliovimo lygmuo)**
8	0.992±0.052	0.875±0.199	0.992±0.077	0.875±0.294
12	0.922±0.127	0.920±0.128	0.922±0.179	0.920±0.181
16	0.982±0.060	0.985±0.059	0.982±0.082	0.985±0.082
20	0.966±0.076	0.960±0.085	0.966±0.103	0.960±0.116
24	0.956±0.077	0.953±0.084	0.956±0.104	0.953±0.114
28	0.849±0.002	0.990±0.000	0.849±0.003	0.990±0.001

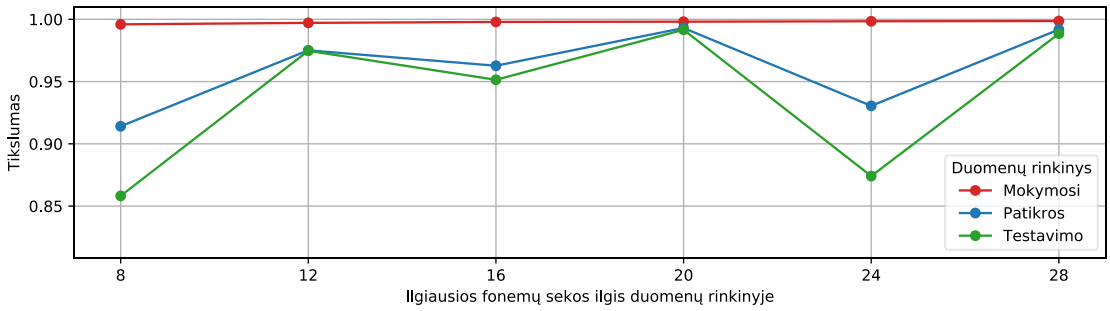
Modelio su dėmesio mechanizmu *Attention-2x64* rezultatai su įvairaus ilgio fonemų sekomis pateikti 4.11 lentelėje.

Palyginus modelio su dėmesio mechanizmu su prieš tai nagrinėto modelio be dėmesio rezultatais, galima teigti, kad modeliai pasiekia labai panašius rezultatus. Sunku nustatyti, kuris modelio tipas veikia geriau, kadangi nėra aiškios tendencijos vieno ar kito modelio veikime: abu modeliai geba išmokti net ir ilgas sekas.

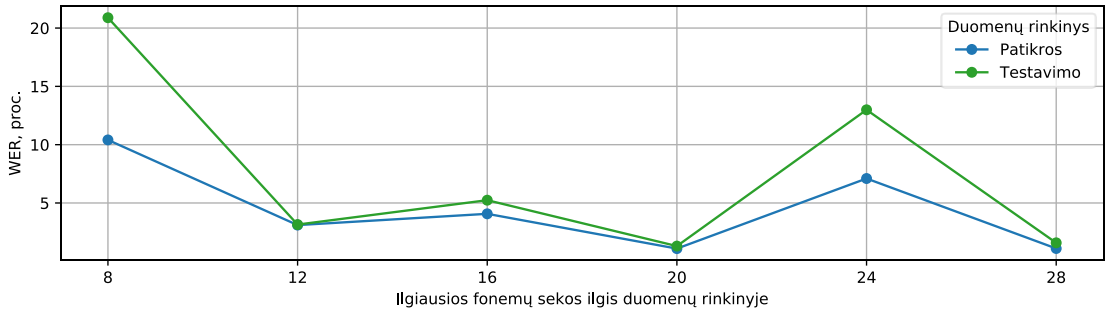
Bendras bruožas, siejantis abi modelių konfigūracijas yra prasti rezultatai su trumpomis fonemų sekomis ($T = 8$). Tokius mokymosi rezultatus galėjo lemti mažas duomenų rinkinys (žr. 7.3 skyrių, 7.14 lentelė).

4.11 lentelė Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio fonemų sekomis, geriausi pasiekti rezultatai

Ilgiausios fonemų sekos ilgis duomenų rinkinyje (*T*)	**Epocha**	**Mokymosi tikslumas**	**Tikslumas su <u>patikros</u> duomenų rinkiniu**	**WER (%) su <u>patikros</u> duomenų rinkiniu**	**Tikslumas su <u>testavimo</u> duomenų rinkiniu**	**WER (%) su <u>testavimo</u> duomenų rinkiniu**
8	16	0,996	0,914	10,406	0,858	20,887
12	19	0,997	0,975	3,104	0,975	3,150
16	20	0,998	0,963	4,071	0,951	5,239
20	17	0,998	0,993	1,087	0,992	1,303
24	18	0,998	0,930	7,098	0,874	12,989
28	20	0,999	0,992	1,098	0,988	1,583



4.10 pav. Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio sekomis, tikslumo grafikas



4.11 pav. Modelio su dėmesio mechanizmu, apmokyto įvairaus ilgio sekomis, WER grafikas

Modelio su dėmesio mechanizmu tikslumo įverčio pasikliovimo intervalus galima rasti 4.12 lentelėje. Naudojantis pasikliautinųjų intervalų reikšmėmis galima teigti, kad mokantis net ir su labai trumpomis fonemų sekomis, modeliai vis tiek sugeba išmokti atpažinti ištartus žodžius iš fonemų sekų.

4.12 lentelė Modelio su dėmesio mechanizmu tikslumo įverčių pasikliautinieji intervalai

Ilgiausios fonemų sekos ilgis duomenų rinkinyje	**Patikros metu gauto tikslumo įverčio PI (0.95 pasikliovimo lygmuo)**	**Testavimo metu gauto tikslumo įverčio PI (0.95 pasikliovimo lygmuo)**	**Patikros metu gauto tikslumo įverčio PI (0.99 pasikliovimo lygmuo)**	**Testavimo metu gauto tikslumo įverčio PI (0.99 pasikliovimo lygmuo)**
8	0.914±0.228	0.858±0.247	0.914±0.337	0.858±0.365
12	0.975±0.094	0.975±0.094	0.975±0.132	0.975±0.133
16	0.963±0.098	0.951±0.113	0.963±0.136	0.951±0.156
20	0.993±0.034	0.992±0.038	0.993±0.046	0.992±0.052
24	0.930±0.106	0.874±0.139	0.930±0.144	0.874±0.188
28	0.992±0.001	0.988±0.000	0.992±0.001	0.988±0.001

4.13 lentelėje pateikiama keletas frazių pavyzdžių, kurios buvo sugeneruotos naudojant modelius, apmokytus su duomenų rinkiniu, kuriame ilgiausias fonemų sekos ilgis siekė 12 fonemų. Kaip matyti iš gautų rezultatų, abu modeliai veikia labai panašiai ir beveik nedaro klaidų atpažįstant frazes iš jų fonemų sekų.

Sekančioje, 4.14 lentelėje, pateikti frazių pavyzdžiai, apmokymui panaudojus duomenų rinkinį kuriame fonemų sekos buvo apribotos iki 28 fonemų. Nors abu modeliai pasiekė labai aukštus tikslumo įvertinimus (žr. 4.9 lentelė ir 4.11 lentelė), nesunku pastebėti, kad modelis su dėmesio mechanizmu veikia daug geriau už modelį be dėmesio mechanizmo - *Attention-2x64* sugeneruotos frazės yra daug tikslesnės, tuo tarpu nors *Seq2Seq-2x128* pateiktose grafemų sekose galima išskirti atskirus žodžius, dauguma jų yra sudarkyti ir

netaisyklingi. Nors modelis su dėmesio mechanizmu daug geriau atpažįsta ilgas frazes nei tradicinis užkoduotojo-dekoduotojo tipo modelis, abu modeliai puikiai veikia atpažįstant trumpus išsireiškimus.

4.13 lentelė Modelių sugeneruotų frazių pavyzdžiai, apmokius modelius fonemų sekomis, apribotomis iki 12 fonemų

Seq2Seq-2x128 **modelio sugeneruota frazė**	***Attention-2x64*** **modelio sugeneruota frazė**	**Tikroji frazė**
ir vilkai nuo	ir vilkai nuo	ir vilkai nuo
lūšies liemuo	lūšies liemuo	lūšies liemuo
augalais	augalais	augalais
rengykle	rengykleTy	rengykle
vienuolių kurie čia	vienuolių kurie čia	vienuolių kurie čia
lietuvių liaudies	lietuvių liaudies	lietuvių liaudies
diena šiltaTa o	diena šilta o	dienaNa šiltaTa o
du tūkstančius	du tūkstančius	du tūkstančius

4.14 lentelė Modelių sugeneruotų frazių pavyzdžiai, apmokius modelius fonemų sekomis, apribotomis iki 28 fonemų

Seq2Seq-2x128 **modelio sugeneruota frazė**	***Attention-2x64*** **modelio sugeneruota frazė**	**Tikroji frazė**
į pistuenuos šalosija ždėpiusiuose	į pietus nuo sacharos išsidėsčiusiuose	į pietus nuo sacharos išsidėsčiusiuose
ilgio ir diekimų stodų snovis ori	ilgio ir iki dviejų šimtų tonų svorio	ilgio ir iki dviejų šimtų tonų svorio
rodyk turinykse yrusi turnys	rodyk turinį skyriaus turinys	rodyk turinį skyriaus turinys
saugo karachonos serutanama	saugoma kauno fortuose randama	saugoma kauno fortuose randama
paleisk pamokro pampa lepaisk	paleisk pašto programą paleisk	paleisk pašto programą paleisk
parodyk ankstesnį	parodyk ankstesnį	parodyk ankstesnį
originalus mastelis	originalus mastelis	originalus mastelis
peržiūrą	peržiūrą	peržiūrą
upės enakortinkolposi dra sent	upės esančio krioklio plotis yra net	upės esančio krioklio plotis yra net
ir kelvadžtoriudesu lmekmeninius	ir perkeldavo didžiulius akmeninius	ir perkeldavo didžiulius akmeninius
iš keliasdešimtiai yra zuvedyno	iš keliasdešimties įvairaus dydžio	iš keliasdešimties įvairaus dydžio
inkilasiegaliotejų tvektviškuose	inkiluose medinių pastatų plyšiuose	inkiluose medinių pastatų plyšiuose

4.4 Kryžminė patikra

Siekiant kuo tiksliau įvertinti modelių efektyvumą nuspręsta juos patikrinti *k*-lypės kryžminės patikros būdu. Šiam eksperimentui pasirinkti du prieš tai naudoti modeliai: *Seq2Seq-2x128* ir *Attention-2x64*. Kryžminė patikra atlikta su izoliuotų žodžių ir nuoseklios šnekos duomenų rinkiniais.

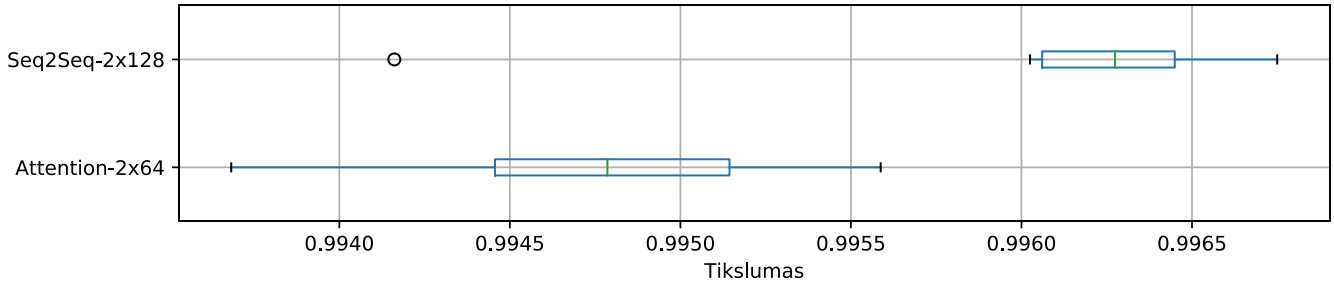
Toks papildomas modelių patikros metodas pasirinktas ir dėl savo savybės, leidžiančios nustatyti ar gautiems rezultatams įtakos neturėjo šališkumas, atsiradęs atsitiktiniu būdu sudarant mokymosi, testavimo ir patikros duomenų imtis.

4.4.1 Izoliuotų žodžių rinkinys

k-lypei kryžminei patikrai atlikti nuspręsta duomenų rinkinį, susidedantį iš atskirų (izoliuotų) žodžių fonemų sekų, padalinti į 10 dalių ($k = 10$). Kiekvieno žingsnio per padalintą duomenų rinkinį metu modeliai buvo apmokyti po 1 epochą su 0,001 mokymosi žingsniu, naudojant 9 dalis apmokymui ir 1, likusią, dalį tikslumui įvertinti. Mažas mokymosi epochų skaičius pasirinktas dėl itin ilgo kryžminės patikros proceso – dalinant duomenų rinkinį į 10 dalių, modelį reikia apmokyti 10 kartų. Mokymosi metu naudotos atbulos fonemų sekos.

4.12 pav. stačiakampės diagramos formatu pateikti *Seq2Seq-2x128* ir *Attention-2x64* pasiekti rezultatai kryžminės patikros metu. *Seq2Seq-2x128* modelio pasiektų tikslumų mediana – 0.9963, o *Attention-2x64* – 0,9947. Tiek *Seq2Seq-2x128*, tiek *Attention-2x64* modelių rezultatai pasižymi siauru ruožu tarp antrojo ir trečiojo kvartilių, o didžiausios ir mažiausios kryžminės patikros metu gautų tikslumų reikšmės yra sąlyginai mažai nutolusios nuo medianų.

Kryžminės patikros rezultatai patvirtina, kad modeliai geba išmokti atpažinti atskirus žodžius iš jų fonemų sekų, o gauti aukšti tikslumo įverčiai liudija šių modelių efektyvumą. Remiantis maža išmatuotų tikslumo reikšmių sklaida, galima teigti, kad šališkumas, kuris galėjo atsirasti atsitiktiniu būdu sudarant mokymosi, testavimo ir patikros imtis, neturėjo didelės įtakos modelių rezultatams.



4.12 pav. *Seq2Seq-2x128* ir *Attention-2x64* kryžminės patikros rezultatai (izoliuotų žodžių rinkinys)

4.4.2 Nuoseklios kalbos rinkinys

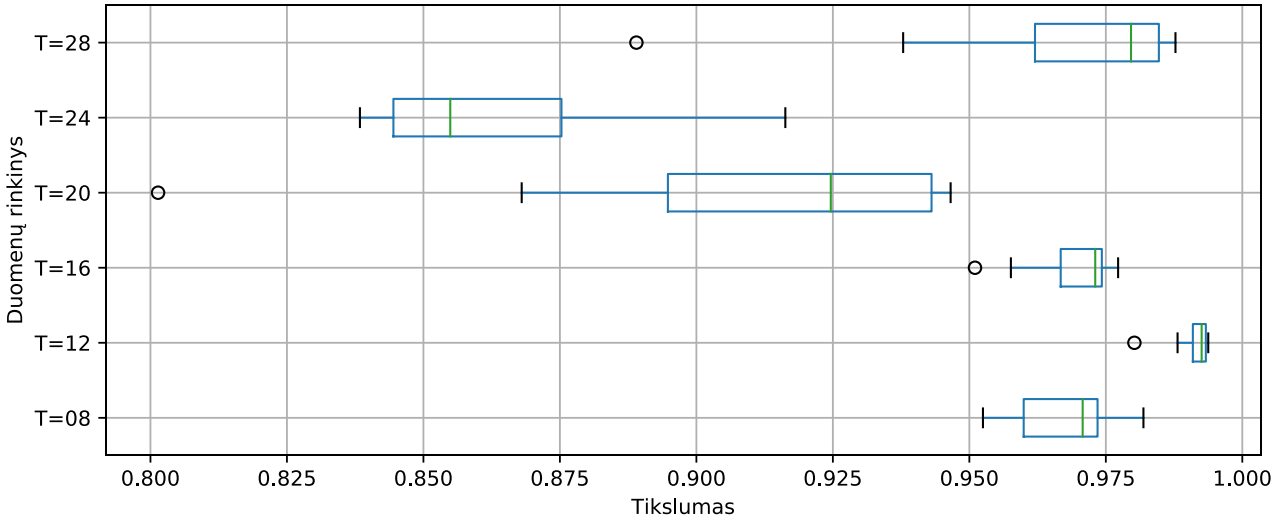
Kryžminės patikros eksperimentas su rišlios kalbos duomenų rinkiniu atliktas remiantis 4.3 skyriuje aprašytu eksperimentu: iš pradinio duomenų rinkinio sudaryti 6 duomenų rinkiniai, kurių kiekvienas skyrėsi ilgiausios fonemų sekos ilgiu.

Seq2Seq-2x128 ir *Attention-2x64* modeliai buvo apmokyti naudojant kiekvieną duomenų rinkinį *k*-lypės kryžminės patikros būdu. Algoritmų pradžioje kiekvienas duomenų rinkinys buvo padalintas į 10 dalių. Kryžminės patikros algoritmas vykdytas 10 kartų (žingsnių), kiekvieno žingsnio metu apmokant modelį 9 dalimis ir įvertinant tikslumą likusia dalimi, tarp žingsnių pasirenkant vis kitą, dar nenaudotą dalį modelių patikrai. Kiekvieno žingsnio metu modelių mokymasis truko po 1 epochą, apmokymui naudojant atbulas sekas ir 0,001 mokymosi žingsnį.

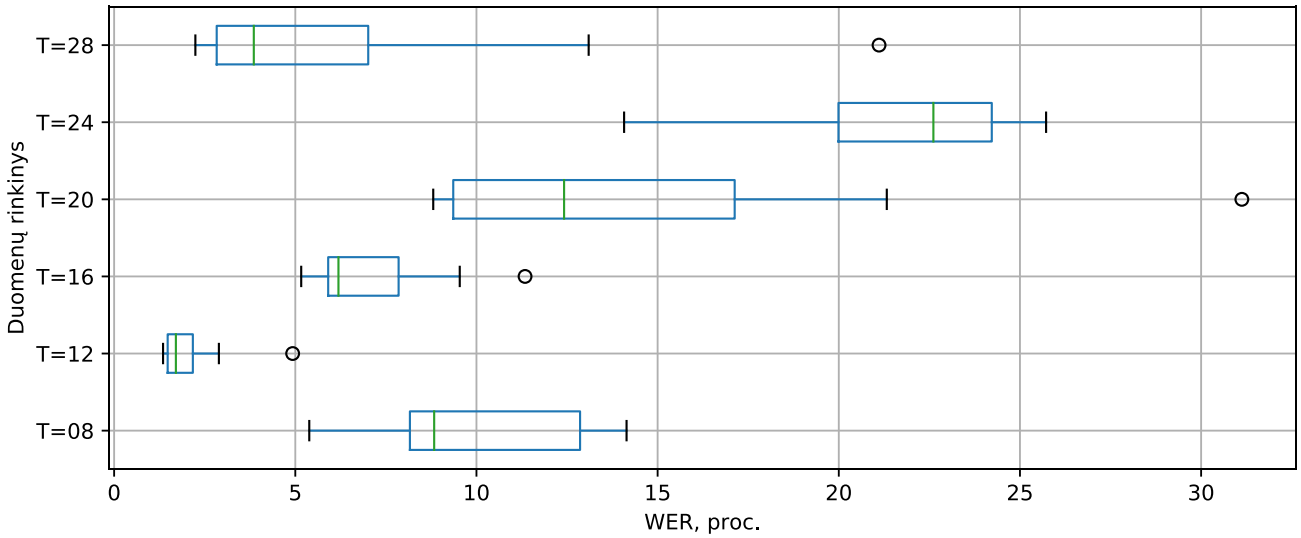
Pasirinkto modelio be dėmesio mechanizmo kryžminės patikros metu gauti tikslumo įverčiai pateikti stačiakampės diagramos formatu 4.13 pav. Grafike diagramos sugrupuotos pagal naudotą duomenų imtį T (T žymi ilgiausios fonemų sekos ilgį rinkinyje). Atitinkamas žodžių atpažinimo klaidos rodiklio grafikas pateiktas 4.14 pav.

Remiantis *Seq2Seq-2x128* kryžminės patikros rezultatais, galima teigti, kad modeliai geba išmokti prognozuoti ištartas frazes pagal jų fonemų sekas, tačiau ilgėjant sekoms jų efektyvumas pradeda kristi. Be

abejo, reikia nepamiršti, kad kryžminės patikros metu modeliai buvo apmokomi tik po vieną epochą, o tai galėjo stipriai paveikti modelių, kurie buvo apmokyti ilgesnėmis fonemų sekomis, pasiektus rezultatus.

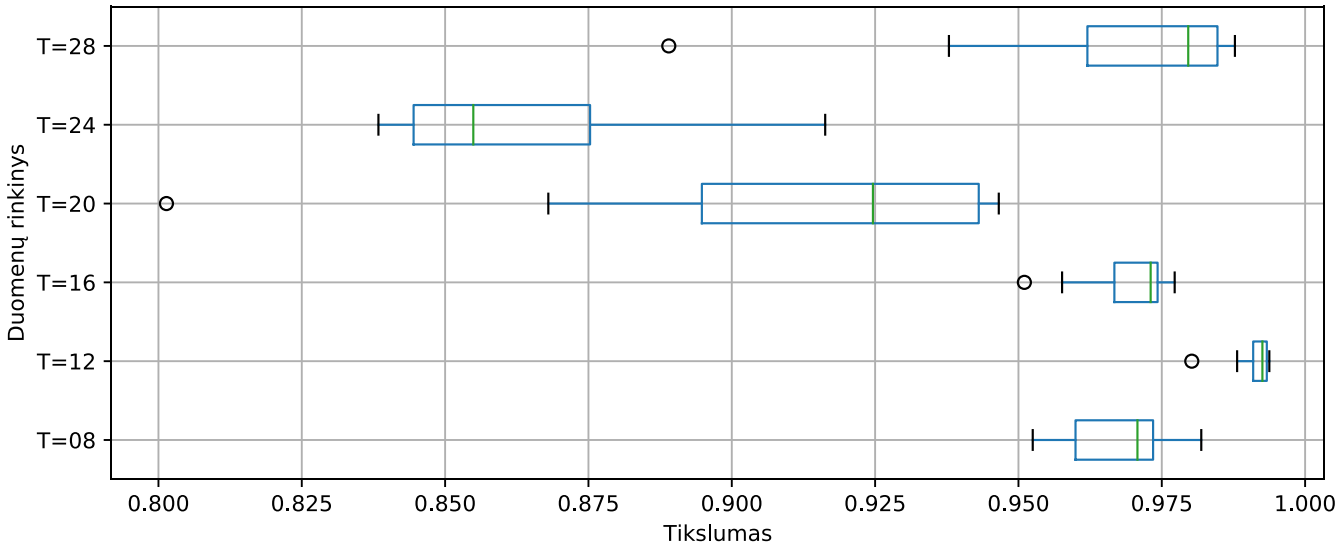


4.13 pav. *Seq2Seq-2x128* modelio kryžminės patikros metu gautų tikslumo įverčių grafikas

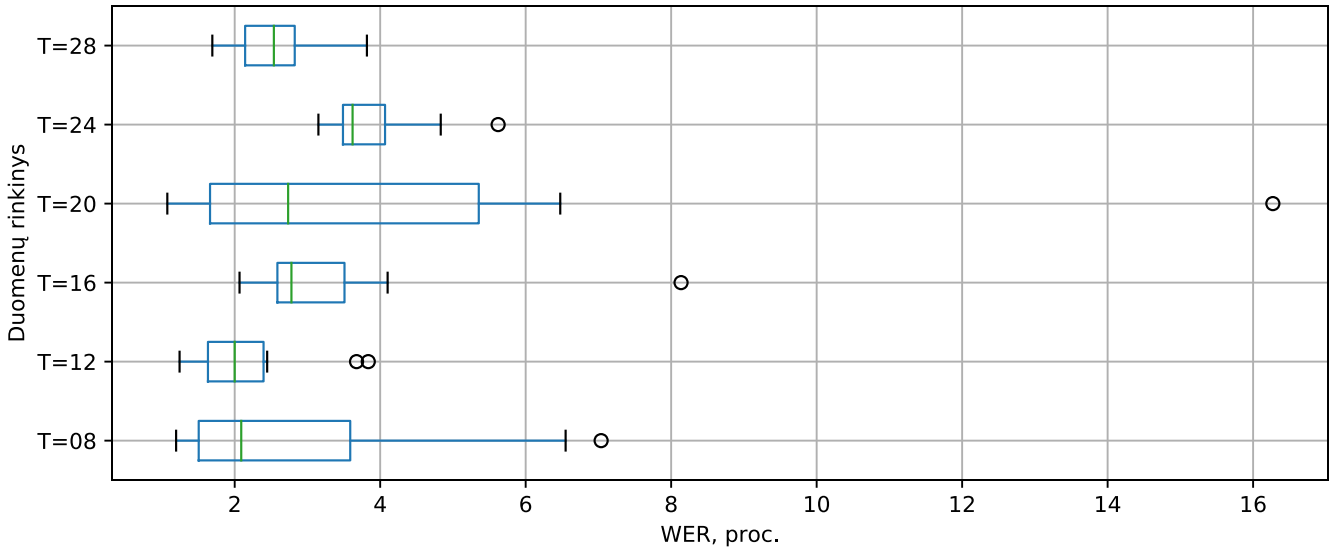


4.14 pav. *Seq2Seq-2x128* modelio kryžminės patikros metu WER įverčių grafikas

Modelio su dėmesio mechanizmu tikslumo įverčiai pateikti 4.15 pav., o WER grafikas – 4.16 pav. *Attention-2x64* rezultatų grafikuose galima matyti tą pačią tendenciją, kaip ir *Seq2Seq-2x128* rezultatuose – modeliai geba išmokti atpažinti sekas, tačiau veikia prasčiau su ilgesnėmis sekomis. Palyginus *Seq2Seq-2x128* ir *Attention-2x64* modelių kryžminės patikros rezultatus, galima pastebėti, kad modelis su dėmesio mechanizmu veikia stabiliau nei tradicinis modelis – gautų tikslumo įverčių sklaida yra daug mažesnė ir beveik visi tikslumo įverčiai viršija 0,90 tikslumo ribą. Panašus kryptingumas pastebimas ir žodžių atpažinimo klaidos rodiklio metrikoje – modelio su dėmesio mechanizmu WER yra mažesni ir mažiau išsisklaidę.



4.15 pav. *Attention-2x64* modelio kryžminės patikros metu gautų tikslumo įverčių grafikas



4.16 pav. *Attention-2x64* modelio kryžminės patikros metu gautų WER įverčių grafikas

5 Išvados

1. Atlikta automatinio kalbos atpažinimo modelio komponenčių analizė parodė, kad automatiniam lietuvių šnekos atpažinimui skirta sistema, turėtų atpažinti ištartus žodžius iš jų fonemų sekų ir gebanti sugeneruoti žodžius atitinkančias grafemų sekas.
2. Atlikta literatūros analizė parodė, kad automatinio lietuvių kalbos atpažinimo sritis yra mažai tyrinėta. Didžioji dalis tyrimų šioje srityje yra susiję su atskirų automatinio šnekos atpažinimo dalių kūrimu ir tobulinimu. Gilaus mokymosi metodai iki šiol buvo naudoti retai, o automatinio šnekos atpažinimo iš fonemų sekų uždavinys iki šiol nebuvo nagrinėtas.
3. Automatinio lietuvių šnekos atpažinimo uždaviniui spręsti pritaikyta speciali giliųjų dirbtinių neuroninių tinklų architektūra – užkoduotojo-dekoduotojo tipo modeliai. Darbe naudotos dvi užkoduotojo-dekoduotojo tipo modelių struktūros: įprastas, tradicinis modelis ir sudėtingesnis modelis su dėmesio mechanizmu. Darbe tyrinėta, kaip įvairių autorių pasiūlyti metodai, tokie kaip mokymasis naudojant atbulas sekas, gradientų „karpymas“ ar reguliarizacijos metodų taikymas, veikia tinklo mokymąsi. Nustatyta, kad pasirinkti modeliai pasiekia geriausius rezultatus naudojant apmokymą su atbulomis sekomis. Naudojant šiuos modelius taip pat yra tikslinga taikyti ankstyvo stabdymo strategiją.
4. Eksperimento metu gauti rezultatai parodė, kad užkoduotojo-dekoduotojo tipo modeliai be didelio vargo gali išmokti atpažinti įvairaus ilgio izoliuotus žodžius iš jų fonemų sekų. Darbe tyrinėti modeliai pasiekė didesnį nei 99 proc. atpažinimo tikslumą su testavimo duomenų rinkiniu, o atliktos kryžminės patikros rezultatai patvirtina tokių modelių efektyvumą. Abu modeliai (modelis su dėmesio mechanizmu ir be jo) pasiekė labai panašius rezultatus, todėl sunku išskirti pranašesnį.
5. Eksperimentais įrodyta, kad užkoduotojo-dekoduotojo tipo modeliai gali būti sėkmingai pritaikyti ir rišlios šnekos atpažinimui (99,8 proc. tikslumas su duomenų rinkiniu, kuriame ilgiausia fonemų seka turi 28 fonemas). Mažas žodžių atpažinimo klaidos rodiklis (1.5 proc. su duomenų rinkiniu, kuriame ilgiausia fonemų seka turi 28 fonemas) liudija, kad užkoduotojo-dekoduotojo tipo modeliai gali ne tik išmokti ryšį tarp fonemų ir jas atitinkančių grafemų sekų, bet ir išskirti žodžių pradžias ir pabaigas fonemų sekose. Nors šiame darbe tyrinėtos sekos su daugiau nei vienu ištartu žodžiu buvo apribotos iki 28 fonemų, modelių pasiekti rezultatai leidžia daryti prielaidą, kad užkoduotojo-dekoduotojo tipo modeliai gali veikti ir su daug ilgesnėmis sekomis. Atliktos kryžminės patikros rezultatai patvirtina, kad užkoduotojo-dekoduotojo tipo modelius tikslinga naudoti sprendžiant automatinio lietuvių šnekos atpažinimo iš fonemų sekų uždavinį.

6 Literatūros sąrašas

[1] R. E. Gruhn, W. Minker, and S. Nakamura, “Statistical Pronunciation Modeling for Non-Native Speech Processing,” 2011, pp. 5–17.

[2] D. Sabaliauskas, “Lietuvių kalbos atpažinimas iOS įrenginiuose.,” *Lietuvių kalbos atpažinimas iOS įrenginiuose: magistro darbas*. Kauno technologijos universitetas. Prieiga per eLABa – nacionalinė Lietuvos akademinė elektroninė biblioteka., Kaunas, 2014.

[3] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, “Deep Speech: Scaling up end-to-end speech recognition,” *CoRR*, vol. abs/1412.5, 2014.

[4] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2016–May, pp. 4945–4949, 2016.

[5] C.-C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, “State-of-the-art Speech Recognition With Sequence-to-Sequence Models,” 2018.

[6] D. Sipavičius and R. Maskeliūnas, “‘Google’ Lithuanian speech recognition efficiency evaluation research.,” *Inf. Softw. Technol. Proc. 22nd Int. Conf. ICIST 2016, Druskininkai, Lith. Oct. 13-15, 2016*, pp. 602–612, 2016.

[7] “Raštija: Apie projektą LIEPA.” [interaktyvus]. Prieiga per: https://www.raštija.lt/liepa. [žiūrėta 2017-12-28].

[8] A. Lipeika, J. Lipeikienė, and L. Telksnys, “Development of Isolated Word Speech Recognition System,” *Informatica*, vol. 13, no. 1, pp. 37–46, 2002.

[9] G. Raškinis and D. Raškinienė, “Building Medium-Vocabulary Isolated-Word Lithuanian HMM Speech Recognition System,” *Informatica*, vol. 14, no. 1, pp. 75–84, 2003.

[10] M. Filipovič and A. Lipeika, “Development of HMM / Neural Network-Based Medium-Vocabulary Isolated-Word Lithuanian Speech Recognition System,” *Informatica*, vol. 15, no. 4, pp. 465–474, 2004.

[11] D. Šilingas, S. Laurinčiukaitė, and L. Telksnys, “Towards Acoustic Modeling of Lithuanian Speech,” 2004.

[12] A. Lipeika and S. Laurinčiukaitė, “Framework for Choosing a Set of Syllables and Phonemes for Lithuanian Speech Recognition,” vol. 18, no. 3, pp. 395–406, 2007.

[13] S. Laurinčiukaitė and A. Lipeika, “Syllable-Phoneme based Continuous Speech Recognition,” *Electr. Eng.*, vol. 6, no. 6, pp. 10–13, 2006.

[14] R. Lileikytė, A. Gorin, L. Lamel, J. L. Gauvain, and T. Fraga-Silva, “Lithuanian Broadcast Speech Transcription Using Semi-supervised Acoustic Model Training,” *Procedia Comput. Sci.*, vol. 81, no. May, pp. 107–113, 2016.

[15] T. Alumäe and O. Tilk, “Automatic speech recognition system for Lithuanian broadcast audio,” *Front. Artif. Intell. Appl.*, vol. 289, pp. 39–45, 2016.

[16] R. Lileikytė, L. Lamel, J.-L. Gauvain, and A. Gorin, “Conversational telephone speech recognition for Lithuanian,” *Comput. Speech Lang.*, vol. 49, pp. 71–82, 2018.

[17] M. J. F. Gales, K. M. Knill, and A. Ragni, “Unicode-based graphemic systems for limited resource languages,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5186–5190.

[18] A. Rudžionis, R. Maskeliūnas, K. Ratkevičius, and V. Rudžionis, “Investigation of voice servers application for Lithuanian language,” *Elektron. Ir Elektrotechnika*, vol. 6, no. 6, 2007.

[19] “Lithuanian Speech Recognition Using the English Recognizer Lithuanian Speech Recognition Using the English Recognizer,” no. January 2008, 2013.

[20] R. Maskeliunas, A. Rudžionis, K. Ratkevičius, and V. Rudžionis, “Investigation of foreign languages models for lithuanian speech recognition,” *Elektron. ir Elektrotechnika*, vol. 3, no. 3, pp. 15–20, 2009.

[21] V. Rudžionis, R. Maskeliūnas, A. Rudžionis, and K. Ratkevičius, “On the adaptation of foreign language speech recognition engines for lithuanian speech recognition,” *Lect. Notes Bus. Inf. Process.*, vol. 37 LNBIP, pp. 113–118, 2009.

[22] R. Maskeliūnas, A. Rudžionis, ir V. Rudžionis, “Analysis of the possibilities to adapt the foreign language speech recognition engines for the lithuanian spoken commands recognition,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5641 LNAI, pp. 409–422, 2009.

[23] V. Rudžionis, G. Raškinis, R. Maskeliūnas, A. Rudžionis, ir K. Ratkevičius, “Comparative analysis of adapted foreign language and native lithuanian speech recognizers for voice user interface,” *Elektron. ir Elektrotechnika*, vol. 19, no. 7, pp. 90–93, 2013.

[24] S. R. Beigaitė, G. Ciganaitė, A. Keršytė, P. Pažėra, S. Jurgita, K. Dzikienė, and T. Krilavičius, “Lithuanian spontaneous speech recognition.”

[25] A. Rudžionis and V. Rudžionis, “Phoneme recognition in fixed context using regularized discriminant analysis,” in *Sixth European Conference on Speech Communication and Technology*, 1999.

[26] K. Driaunys, V. Rudžionis, and P. Žvinys, “Averaged Templates Calculation and Phoneme Classification,” *Inf. Technol. Control*, vol. 36, no. 1, 2007.

[27] A. Rudžionis and V. Rudžionis, “Lithuanian Speech Database LTDIGITS,” in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC02)*, 2002.

[28] A. Vaičiūnas, V. Kaminskas, and G. Raškinis, “Statistical Language Models of Lithuanian Based on Word Clustering and Morphological Decomposition.,” *Informatica, Lith. Acad. Sci.*, vol. 15, pp. 565–580, 2004.

[29] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A System for Large-scale Machine Learning,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[30] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, “The Kaldi Speech Recognition Toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 2011.

[31] MathWorks, “MATLAB - MathWorks,” 2018. [interaktyvus]. Prieiga per: https://se.mathworks.com/products/matlab.html?s_tid=hp_products_matlab. [žiūrėta 2018-01-19].

[32] A. Hannun, “Sequence Modeling with CTC,” *Distill*, 2017.

[33] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, “A Comparison of sequence-to-sequence models for speech recognition,” *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2017–Augus, pp. 939–943, 2017.

[34] V. Ambrazas ir Lietuvių kalbos Institutas, *Dabartinės lietuvių kalbos gramatika*, 2–asis pat ed. Vilnius: Mokslo ir enciklopedijų leidykla, 1996.

[35] A. Girdenis, “Lietuvių kalbos fonologija,” *Visuotinė Lietuvių Enciklopedija*, 2016. [interaktyvus]. Prieiga per: https://www.vle.lt/Straipsnis/lietuviu-kalbos-fonologija-118007. [žiūrėta 2019-04-07].

[36] L. Deng, “Deep learning: from speech recognition to language and multimodal processing,” *APSIPA Trans. Signal Inf. Process.*, vol. 5, p. e1, Jan. 2016.

[37] S. R. Mundra Rohit, “CS 224D: Deep Learning for NLP1 - Lecture Notes: Part III.” 2015.

[38] T. Epelbaum, “Deep learning: Technical introduction,” *arXiv Prepr. arXiv1709.01412*, 2017.

[39] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[40] O. Christopher, “colah’s blog: Understanding LSTM Networks,” 2015. [interaktyvus]. Prieiga per: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [žiūrėta 2018-01-14].

[41] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International Conference on Machine Learning*, 2014, pp. 1764–1772.

[42] A. Graves, N. Jaitly, and A. Mohamed, “Hybrid speech recognition with deep bidirectional LSTM,” in *2013 IEEE workshop on automatic speech recognition and understanding*, 2013, pp. 273–278.

[43] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent Neural Network Regularization,” *CoRR*, vol. abs/1409.2, 2014.

[44] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” *CoRR*, vol. abs/1412.3, 2014.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[46] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” pp. 1–9, 2014.

[47] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” pp. 1–15, 2014.

[48] R. Nallapati, B. Zhou, C. N. dos santos, C. Gulcehre, and B. Xiang, “Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond,” pp. 280–290, 2016.

[49] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12–June, pp. 3156–3164, 2015.

[50] C. Wang, H. Yang, C. Bartz, and C. Meinel, “Image Captioning with Deep Bidirectional LSTMs.”

[51] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks,” 2012.

[52] M.-T. Luong, H. Pham, and C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” 2015.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *CoRR*, vol. abs/1412.6, 2014.

7 Priedai

7.1 Priedas A

Priede pateikti detalūs panaudos atvejų aprašai. Bendrieji PA „Nurodyti parametrus“, „Nurodyti žodynus“ ir „Generuoti duomenų rinkinius“ nėra aprašomi, kadangi juos detalizuoja kiti panaudos atvejai.

7.1 lentelė Panaudos atvejo „Nurodyti hiperparametrų failą“ aprašymas

PA „Nurodyti hiperparametrų failą“	
Tikslas Nustatyti modelio hiperparametrus	
Aprašymas PA skirtas modelio hiperparametrams nustatyti: dirbtinių neuronų skaičiui, apmokymo partijos dydžiui, mokymosi žingsniui ir pan. Nustatymai yra nurodomi konfigūracinio failo pagalba, kuris formatuojamas JSON formatu.	
Prieš sąlyga Kompiuteryje yra vartotojo sukurtas konfigūracinis failas	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas parametrų failo pavadinimas.	1. Visų pirma, sistema patikrina, ar failas egzistuoja. Antra, tikrinama, ar parametrai, nurodyti faile, yra teisingi, t. y. atitinka iš anksto apibrėžtus jų reikalavimus. Pagal nurodytus parametrus sukuriamas modelis su iš anksto numatytais svoriais.
2. Baigiamas PA	
Po sąlyga	Programoje sukurtas modelis su numatytaisiais svoriais.

7.2 lentelė Panaudos atvejo „Nurodyti modelio katalogą“ aprašymas

PA „Nurodyti modelio katalogą“	
Tikslas Nurodyti naujo (arba jau esamo) modelio katalogą parametrų saugojimui	
Aprašymas Modelio katalogas apibrėžia, kur bus saugojamas modelis ir jo parametrai, pvz., dirbtinių neuronų svoriai. Jei nurodomas katalogas, kuriame jau yra saugomas modelis, jo parametrai yra įkeliami į programą tolimesniam naudojimui. Jei kataloge yra išsaugotą keletas to paties modelio parametrų rinkinių – į programą įkeliami paskutiniai (naujausi) modelio parametrai.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodoma modelio katalogo vieta.	1. Jei katalogas jau egzistuoja ir nėra tuščias, tikrinama, ar jame yra saugomas modelis. Jei taip – modelio parametrai yra įkeliami į programą. Jei katalogas neegzistuoja – jis yra sukuriama.
2. Baigiamas PA	
Po sąlyga	Sukurtas naujo modelio katalogas arba įkelti esamo modelio parametrai.

7.3 lentelė Panaudos atvejo „Apmokyti modelį“ aprašymas

PA „Apmokyti modelį“	
Tikslas Apmokyti esamą (arba naują) modelį	
Aprašymas PA skirtas apmokyti modelį vartotojo pateiktu duomenų rinkiniu	
Prieš sąlyga Įvykdytas „Nustatyti parametrus“ PA	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas modelio mokymosi režimas.	1. Sistema nuskaito ir apdoroja vartotojo pateiktą duomenų rinkinį, paruošdama jį modelio mokymuisi. Modelis yra apmokomas pagal vartotojo pateiktus parametrus (partijos dydį, mokymosi epochų skaičių ir pan.) Mokymosi pabaigoje modelio parametrai yra išsaugomi nurodytame modelio kataloge.
2. Baigiamas PA	
Po sąlyga	Modelio kataloge išsaugoti apmokyto modelio parametrai ir mokymosi metu pasiekti rezultatai.

7.4 lentelė Panaudos atvejo „Ištestuoti modelį“ aprašymas

PA „Ištestuoti modelį“	
Tikslas Įvertinti modelio tikslumą nepriklausomo duomenų rinkinio pagalba	
Aprašymas Panaudos atvejis skirtas įvertinti modelio tikslumą pagal numatytas kokybės vertinimo metrikas vartotojo nurodyto duomenų rinkinio pagalba. Jei šis atvejis atliekamas kartu su kitu – „Apmokyti modelį“ – panaudos atveju, modelis yra įvertinimas po kiekvienos mokymosi epochos.	
Prieš sąlyga Įvykdytas „Nustatyti parametrus“ PA	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas modelio testavimo režimas.	1. Sistema nuskaito ir apdoroja vartotojo pateiktą duomenų rinkinį, paruošdama jį modelio mokymuisi. Įkeltas modelis yra ištestuojamas apdoroto duomenų rinkinio pagalba, o pasiekti rezultatai yra išsaugomi modelio kataloge.
2. Baigiamas PA	
Po sąlyga	Modelio kataloge išsaugoti patikros metu pasiekti rezultatai.

7.5 lentelė Panaudos atvejo „Nurodyti mokymosi duomenų rinkinį“ aprašymas

PA „Nurodyti mokymosi duomenų rinkinį“	
Tikslas Nurodyti duomenų rinkinį, kuriuo naudojantis modelis bus apmokytas	
Aprašymas PA skirtas nurodyti duomenų rinkinį, kuriuo bus apmokomas statistinis modelis. Duomenų rinkinį sudaro failas (-ai) su fonemų ir jas atitinkančių grafemų sekomis. Sekų formatas faile: $x_1, x_2 \dots x_T;\ y_1, y_2 \dots y_{T'}$ Kiekviena seka baigiasi eilutės pabaigos simboliu.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas katalogas arba failas su mokymosi duomenimis.	1. Nurodžius failą sistema patikrina, ar failas egzistuoja ir ar failo surašyti duomenys yra atitinkamo formato. Jei buvo nurodytas katalogo pavadinimas – sistema nuskaito visus failus, esančius kataloge.
2. Baigiamas PA	

Po sąlyga	Sistemoje saugomas sąrašas failų, kurių turiniai bus panaudoti mokymosi metu.

7.6 lentelė Panaudos atvejo „Nurodyti testavimo duomenų rinkinį“ aprašymas

PA „Nurodyti testavimo duomenų rinkinį“	
Tikslas Nurodyti duomenų rinkinį, kuriuo naudojantis bus įvertintas modelio efektyvumas	
Aprašymas PA skirtas nurodyti duomenų rinkinį, kuriuo bus patikrintas (ištestuotas) statistinis modelis. Duomenų rinkinį sudaro failas (-ai) su fonemų ir jas atitinkančių grafemų sekomis. Sekų formatas faile: $x_1, x_2 \dots x_T;\ y_1, y_2 \dots y_{T'}$ Kiekviena seka baigiasi eilutės pabaigos simboliu.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas katalogas arba failas su testavimo (arba patikros) duomenimis.	1. Nurodžius failą sistema patikrina, ar failas egzistuoja ir ar failo surašyti duomenys yra atitinkamo formato. Jei buvo nurodytas katalogo pavadinimas – sistema nuskaito visus failus, esančius kataloge.
2. Baigiamas PA	
Po sąlyga	Sistemoje saugomas sąrašas failų, kurių turiniai bus panaudoti testavimo (arba patikros) metu.

7.7 lentelė Panaudos atvejo „Generuoti įprastus rinkinius“ aprašymas

PA „Generuoti įprastus rinkinius“	
Tikslas Sukurti duomenų rinkinius modelio apmokymui, patikrai ir testavimui	
Aprašymas Iš vartotojo nurodytų anotacijų failų sukuriami duomenų rinkiniai modelių apmokymui, patikrai ir testavimui.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodoma: • failų su anotacijomis katalogas; • mokymosi duomenų rinkinio dalis (proc.); • testavimo duomenų rinkinio dalis (proc.); • ar bus sudaromos makro-sekos; • ilgiausios sekos ilgis (matuojamas grafemomis); • grupėje esančių fonemų ilgių skirtumas;	1. Sistema nuskaito visus kataloge esančius failus, kurių plėtinys „.la2“. Pagal vartotojo nurodytus parametrus falai padalinami į mokymosi, patikros ir testavimo duomenų rinkinius. Kiekvienas failas, esantis rinkinyje, yra apdorojamas išgaunant fonemų ir grafemų sekas. Sudaromos makro-sekos (jei nurodyta) arba izoliuotų žodžių fonemų sekos. Sekos yra apdorojamos, atmetant ilgesnes sekas nei buvo nurodyta, ir sugrupuojamos į atskirus failus.
2. Baigiamas PA	
Po sąlyga	Iš failų su anotacijomis sugeneruoti mokymosi, patikros ir testavimo duomenų rinkiniai.

7.8 lentelė Panaudos atvejo „Generuoti kryžminės patikros rinkinius“ aprašymas

PA „Generuoti kryžminės patikros rinkinius“	
Tikslas Sukurti duomenų rinkinius *k*-lypei kryžminei patikrai	
Aprašymas Iš vartotojo nurodytų anotacijų failų sukuriami *k* duomenų rinkinių, kurie gali būti panaudoti kryžminės patikros metu.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
2. Programos paleidimo metu nurodoma: • failų su anotacijomis katalogas; • mokymosi duomenų rinkinio dalis (proc.); • testavimo duomenų rinkinio dalis (proc.); • ar bus sudaromos makro-sekos; • ilgiausios sekos ilgis (matuojamas grafemomis); • grupėje esančių fonemų ilgių skirtumas;	3. Sistema nuskaito visus kataloge esančius failus, kurių plėtinys „.la2“. Pagal vartotojo nurodytus parametrus falai padalinami į nustatytą skaičių duomenų rinkinių. Kiekvienas failas, esantis rinkinyje, yra apdorojamas išgaunant fonemų ir grafemų sekas. Sudaromos makro-sekos (jei nurodyta) arba izoliuotų žodžių fonemų sekos. Sekos yra apdorojamos, atmetant ilgesnes sekas nei buvo nurodyta, ir sugrupuojamos į atskirus failus.
4. Baigiamas PA	
Po sąlyga	Iš failų su anotacijomis sugeneruoti duomenų rinkiniai kryžminei patikrai.

7.9 lentelė Panaudos atvejo „Prognozuoti frazes“ aprašymas

PA „Prognozuoti frazes“	
Tikslas Sugeneruoti grafemų sekas iš duotų fonemų sekų	
Aprašymas Panaudos atvejis skirtas atpažinti ištartus žodžius iš duotų fonemų sekų ir sugeneruoti jas atitinkančias grafemų sekas. Prognozavimo metu tikslumo metrikos nėra skaičiuojamos.	
Prieš sąlyga Įvykdytas „Nustatyti parametrus“ PA	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas frazių prognozavimo režimas.	1. Pagal vartotojo pateiktus parametrus yra sukuriamas statistinis modelis, o iš nurodyto katalogo įkeliami modelio parametrai. Jei kataloge parametrai neegzistuoja – sistema sugeneruoja klaidos pranešimą. Sistemoje esantis statistinis modelis sugeneruoja grafemų sekas iš fonemų sekų, kurios buvo įkeltos į sistemą „Nurodyti fonemų sekų rinkinį“ PA metu. Sugeneruotos sekos (prognozės) yra išsaugomos į „“ PA metu nurodytą katalogą tekstinių failų formatu.
2. Baigiamas PA	
Po sąlyga	Grafemų sekos, gautos statistiniu modeliu apdorojus fonemų sekas, išsaugotos į rezultatų failus nurodytame kataloge.

7.10 lentelė Panaudos atvejo „Nurodyti fonemų sekų rinkinį“ aprašymas

PA „Nurodyti fonemų sekų rinkinį“	
Tikslas Nurodyti failą ar katalogą su fonemų sekomis, kurios bus panaudotos kaip įvestis prognozuojant grafemų sekas	
Aprašymas	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas failas ar katalogas.	1. Sistema patikrina, ar buvo nurodytas failas, ar katalogas. Jei buvo nurodytas katalogas, sistema nuskaito kiekvieno kataloge esančio failo turinį. Jei buvo nurodytas tik failas – nuskaitomas tik to failo turinys. Fonemų sekos įkeliamos į programą tolimesniam apdorojimui.
2. Baigiamas PA	
Po sąlyga	Fonemų sekos įkeltos į sistemą tolimesniam apdorojimui.

7.11 lentelė Panaudos atvejo „Nurodyti rezultatų katalogą“ aprašymas

PA „Nurodyti rezultatų katalogą“	
Tikslas Nurodyti katalogą, kuriame bus saugomi prognozavimo metu gauti rezultatai	
Aprašymas	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas modelio mokymosi režimas.	1. Sistema sukuria katalogą, jei šis dar neegzistuoja, ir išsaugo jame grafemų rezultatus. Rezultatai saugomi tekstinių failų formatu, failo (-ų) pavadinimas (-ai) sutampa su „Nurodyti fonemų sekų rinkinį“ panaudos atvejo metu nurodytu failo (-ų) pavadinimu (-ais).
2. Baigiamas PA	
Po sąlyga	Prognozavimo metu gautos grafemų sekos išsaugotos nurodytame kataloge.

7.12 lentelė Panaudos atvejo „Nurodyti fonemų žodyną“ aprašymas

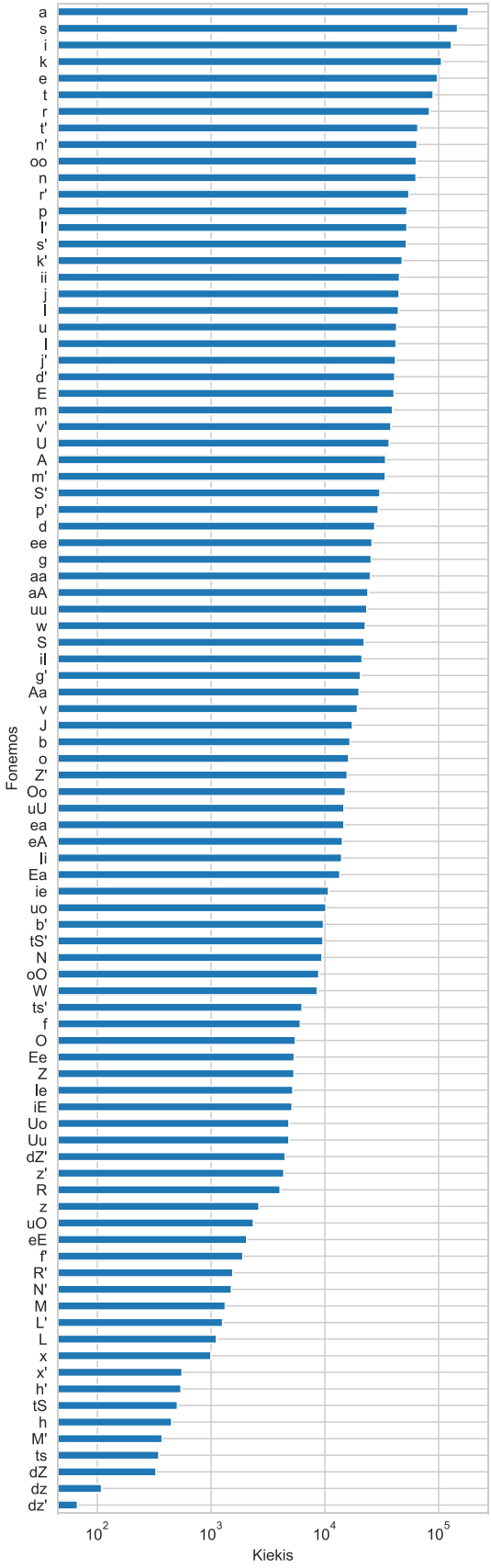
PA „Nurodyti fonemų žodyną“	
Tikslas Nurodyti fonemų žodyną	
Aprašymas Vartotojas nurodo fonemų žodyną, kuriame yra visos apmokymo ir testavimo duomenų rinkiniuose esančios fonemos. Žodynas pateikiamas tekstiniu failo formatu, fonemas atskiriant naujos eilutės simboliu.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas fonemų žodynas.	1. Sistema patikrina, ar failas egzistuoja, ir nuskaito iš jo fonemas. Fonemų žodynas papildomas specialiu simboliu, kuris žymi duomenų rinkiniuose esančias, bet žodyne nenurodytas fonemas.
2. Baigiamas PA	

Po sąlyga	Fonemų žodynas yra įkeltas į sistemą ir paruoštas tolimesniam darbui.

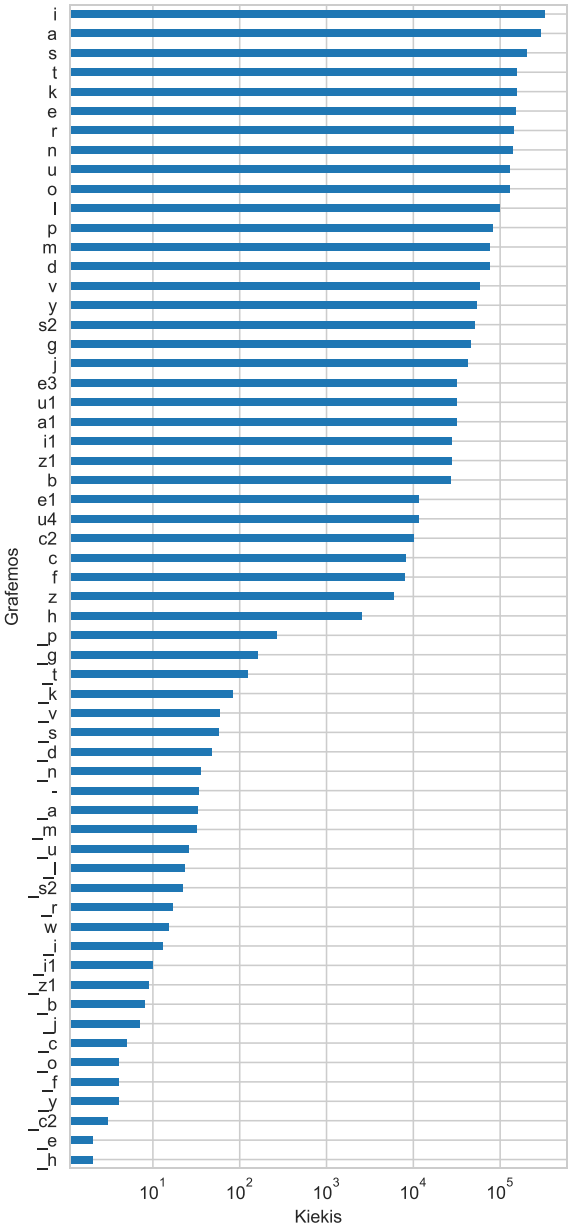
7.13 lentelė Panaudos atvejo „Nurodyti grafemų žodyną“ aprašymas

PA „Nurodyti grafemų žodyną“	
Tikslas Nurodyti grafemų žodyną	
Aprašymas Vartotojas nurodo grafemų žodyną, kuriame yra visos apmokymo ir testavimo duomenų rinkiniuose esančios grafemos (tarp jų ir specialūs simboliai: tarpo simbolis, brūkšnys ir t.t.). Žodynas pateikiamas tekstiniu failo formatu, grafemas atskiriant naujos eilutės simboliu.	
Prieš sąlyga	
Aktorius	Vartotojas
Sužadinimo sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Programos paleidimo metu nurodomas grafemų žodynas.	1. Sistema patikrina, ar failas egzistuoja, ir nuskaito iš jo grafemas. Grafemų žodynas papildomas specialiu simboliu, kuris žymi duomenų rinkiniuose esančias, bet žodyne nenurodytas grafemas.
2. Baigiamas PA	
Po sąlyga	Grafemų žodynas yra įkeltas į sistemą ir paruoštas tolimesniam darbui.

7.2 Priedas B



7.1 pav. Fonemų pasiskirstymo garsyne grafikas.

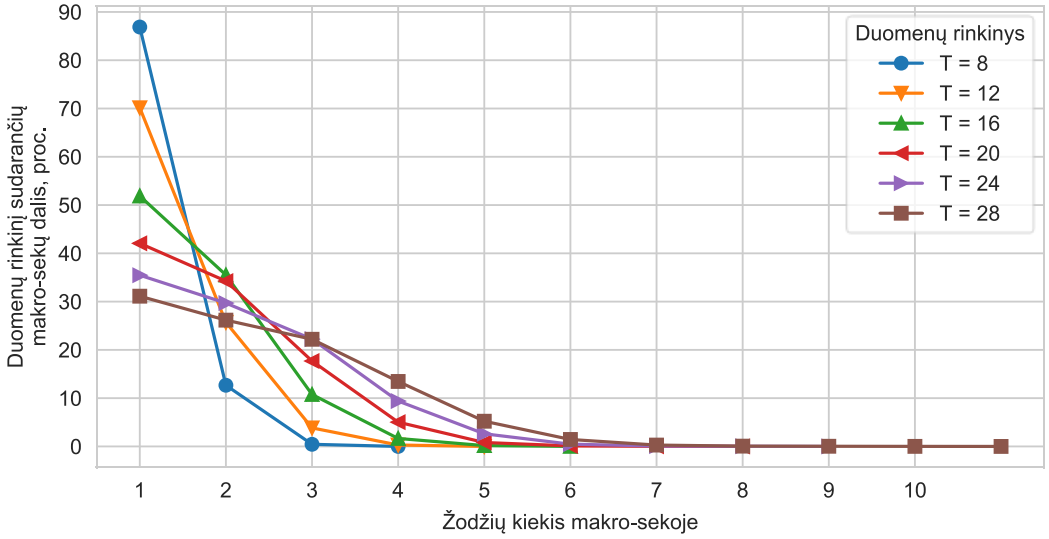


7.2 pav. Grafemų pasiskirstymo garsyne grafikas.

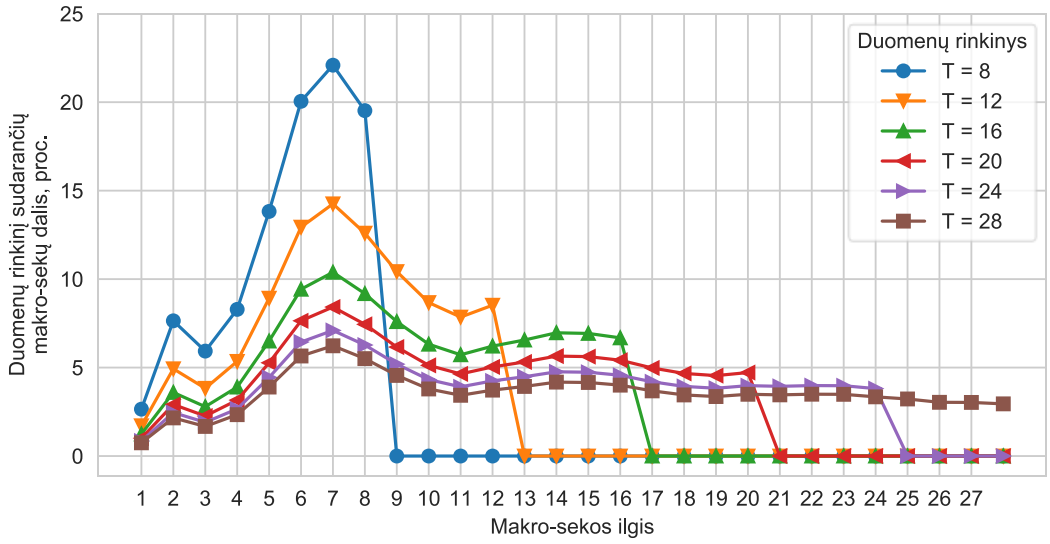
7.3 Priedas C

7.14 lentelė Duomenų rinkinių statistika

Duomenų rinkinys	***T*=8**	***T*=12**	***T*=16**	***T*=20**	***T*=24**	***T*=28**
Žodžių skaičius rinkinyje	417935	765845	1273983	1821439	2473073	3151933
Makro-sekų skaičius rinkinyje	367989	570321	782680	965211	1145086	1304801



7.3 pav. Makro-sekose esančių žodžių kiekio pasiskirstymas duomenų rinkiniuose



7.4 pav. Makro-sekų ilgių pasiskirstymas duomenų rinkiniuose