LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

RUNAS VALODAS NOTEIKŠANA

MAĢISTRA DARBS

Autore: **Veronika Hromčenkova**
Stud. Apl. Nr.: vh19010
Darba vadītājs: Dr. dat. Askars Salimbajevs

RĪGA 2021

ANOTĀCIJA

Dabiskās valodas apstrādes jomā valodas noteikšana vai valodas uzminēšana nozīme noteikt satura dabisko valodu. Runas valodas noteikšanas uzdevums ir noteikt runas audio ieraksta valodu.

Runas valodas noteikšanai ir daudz pielietojumu. Tas ir svarīgs pirmais solis vairākās runas apstrādes sistēmās. Ja valoda tiek identificēta nepareizi, jebkura turpmāka apstrāde sniegs kļūdainus rezultātus. Tas ir īpaši svarīgi daudzvalodu kopienās, kur daudzas vairākas valodas tiek izmantotas salīdzinoši bieži.

Šī darba mērķis ir izpētīt runājamās valodas noteikšanas metodes, koncentrējoties uz 3 Latvijā runājamām valodām: latviešu, krievu, angļu.

Atslēgvārdi: runas valodas noteikšana, mašīnmācīšanās, dziļas mācīšanās, wav2vec

ABSTRACT

Spoken Language Identification

In natural language processing, language identification or language guessing is the problem of determining which natural language given context is in. Spoken Language Identification is the task to identify the language of a spoken utterance.

This is an important first step in several speech processing systems. If language is identified incorrectly any further processing will give erroneous results. This is especially important in bi-lingual and multi-lingual communities, where many several languages are used comparably frequently.

The aim of this work is exploring methods for spoken Language Identification focusing on 3 languages spoken in Latvia: Latvian, Russian, English.

Keywords: spoken language identification, machine learning, deep learning, wav2vec

AUTOREFERĀTS

Šī maģistra darba mērķis ir izpētīt runājamās valodas noteikšanas problēmu un to risinājumus, koncentrējoties uz 3 Latvijā runājamām valodām: latviešu, krievu, angļu.

Maģistra darba ietvaros tika veikts literatūras pārskats, izpētītas runas valodas noteikšanas pieejamas metodes un gatavi risinājumi. Darbā ir izmantoti 49 literatūras avoti, no kuriem 28 ir recenzēti zinātniski raksti un 6 ir no *arXiv* krātuves.

Darba praktiskajā daļā tika implementēti dažādi algoritmi un apmācīti vairāki modeli, izmantojot wav2vec2-xlsr, lai noteikt runas valodu no trim Latvijā runājamām valodām: latviešu, krievu un angļu.

Darba novitāte it tas, ka tika izveidots viens no pirmajam runas valodas noteikšanas risinājumiem, kas atbalsta latviešu valodu. Veicot literatūras izpēti, netika atrasts neviens pētnieciskais darbs, kurā būtu aprakstīts šī uzdevuma risinājums latviešu valodai. Tikai kad maģistra darba lielāka daļa jau bija paveikta, tika atrasts 2021. gada runas valodas noteikšanas pētījums, kas ietver latviešu valodu. Turklāt darba izstrādātajā risinājumā pamatā ir runas dziļas reprezentācijas metode un wav2vec2-xlsr modelis, kas ir jauni un aktuālie pētniecības virzieni, kuru izmantošana runas valodas noteikšanā vēl nav plaši izpētīti.

Darba ietvaros iegūtie rezultāti ir salīdzināti ar Google Cloud Speech-to-Text un VOXLINGUA107 runas valodas noteikšanas sistēmām. Tika konstatēts, ka šī darba ietvaros izstrādāts modelis ir salīdzināms ar tiem, vai pat pārspēj tos.

SATURS

APZĪMĒJUMU SARAKSTS

ASR (*Automatic Speech Recognition*) **-** automātiska runas atpazīšana

CNN (*Convolutional Neural Networks)* - konvolūciju neironu tīkli

Fonēma (*phoneme***) -** atšķirīga skaņu vienība valodā, kas veido katru valodu

Fonotaktika (*phonotactic*) – lingvistikas nozare, kas nodarbojas ar ierobežojumiem valodā attiecībā uz pieļaujamajām fonēmu kombinācijām

GMM (Gaussian mixture model) - Gausa sadalījumu maisījuma modelis

LID (*Language Identification*) – valodas noteikšana

Ne-skaitļošanas (*non-computational*) – pieejas, kas balstās uz ekspertu zināšanām, nevis statistiku

Pašpārraudzītā *(self-supervised)* **pieeja** – mašīnmācīšanās pieeja, kurā modelis pats izveido datu marķējumus

Pietrenēšana (fine-tuning) - mašīnmācīšanās pieeja, kur iepriekš apmācīts modelis tiek pielāgots jaunam uzdevumām

SLID (*Spoken Language Identification*) – runas valodas noteikšana

Runas dziļas reprezentācijas (*Speech embeddings*)

RNN (*Recurrent Neural Networks*) – rekurentie neironu tīkli

KNN (*k-nearest neighbors*)– k-tuvāko kaimiņu algoritms

MLP (*multilayer perceptron*) – daudzslāņu perceptrons

IEVADS

Valodas noteikšana (*Language Identification*, LID) ir uzdevums, kura mērķis ir noteikt dota satura dabisko valodu. Saturs varētu būt rakstīts teksts, attēls vai runas audio ieraksts.

Runas valodas identifikācijai (*Spoken Language Identification*, SLID) ir vairāki pielietojumi. Piemēram, tas palīdz noteikt runas valodu telefona zvanā un pārsūtīt ienākošo zvanu attiecīgajam speciālistam tādu telefonisku informācijas pakalpojumu jomā kā klientu apkalpošana, preču pasūtīšana pa tālruni, interaktīvi balss atbildes pakalpojumi un citu zvanu centru pakalpojumi. Kā arī SLID var izmantot, lai strukturēt iepriekš ierakstītu zvanu arhīvu un arī citus audio datus pēc valodas.

Runas asistentos vai citās lietotnēs ar balss saskarni valodas noteikšana darbojas kā pirmais solis, kas izvēlas atbilstošo valodu no pieejamo valodu saraksta. Piemēram, publicējot savu pakalpojumu, lietotni vai produktu valstī, kurā ir vairākas oficiālās vai runājamas valodas, pastāv iespēja no lietotājiem saņemt audio dažādās valodās. Tas var ievērojami apgrūtināt šo pieprasījumu apstrādi.

Valodas identifikācija iedalās divās pieejās: ne-skaitļošanas (*non-computational*) un skaitļošanas (*computational*). Ne-skaitļošanas pieeja prasa pietiekami dziļas zināšanas par identificējamo valodu, piemēram, diakritiskas zīmes, visbiežāk lietotie vārdi, rakstzīmju kombinācijas utt. Bet skaitļošanas pieeja balstās uz statistikas paņēmieniem, nevis uz valodu zināšanām. Katras valodas identificēšanai ir nepieciešams liels apmācību datu apjoms. Statistiskās pieejas ietver sevī divus soļus: apmācību un klasifikāciju. Apmācības solī tiek veikta pazīmju izgūšana no dotās apmācības datu kopas, ko sauc par korpusu. No šim pazīmēm tiek izveidoti valodu profili. Klasifikācijas solī tiek noskaidrota apmācības profila un klasificējamu datu profila līdzība, un līdzīgākā valoda tiek izvēlēta kā dokumenta valoda.

Teksta valodas noteikšanai var izmantot gan skaitļošanas, gan ne-skaitļošanas metodes. Tomēr runas valodas noteikšanai izmanto tikai skaitļošanas metodes, runas signāla sarežģītības dēļ. Līdz ar to, šajā darbā tiek aplūkotas tikai skaitļošanas metodes.

Šī maģistra darba mērķis ir izpētīt runājamās valodas noteikšanas problēmu un to risinājumus, koncentrējoties uz 3 Latvijā runājamām valodām: latviešu, krievu, angļu. Tiks pētīti gan eksistējoši gatavi pakalpojumi, gan vairākas valodas noteikšanas pieejas. Īpašā uzmanība tiks

pievērsta, lai šo metodi varētu implementēt bez eksperta līmeņa zināšanām mašīnmācīšanās un lingvistiskas jomās.

Darbā ir ievads, 2 nodaļas, secinājumi, bibliogrāfiskais saraksts un viens pielikums. Darba struktūra ir sekojoša:

- Pirmā nodaļa ir veltīta iepazīšanai ar valodas noteikšanas metodēm. Tiek apskatīta arī esoša situācija ar pieejamiem mākoņpakalpojumiem un pieejamiem apmācības datiem valodas noteikšanai mūs interesējošām valodām.
- Otra nodaļa ir veltīta izvēlētas pieejas aprakstam. Tiek apkopoti visu paveikto eksperimentu rezultāti, salīdzināti savā starpā un ar citiem pieejamiem līdzīgiem risinājumiem.

Darba noslēgumā ir apkopoti darba rezultāti un izdarīti secinājumi par paveikto darbu.

1. VALODAS NOTEIKŠANA

1.1 Pieejamie gatavi risinājumi

Sākot pētījumu, tika nolemts vispirms izpētīt vai eksistē gatavi risinājumi, piemēram, mākoņpakalpojumi un API, kuri jau izpilda maģistra darba ievadā uzstādītus mērķus.

Pētot situāciju ar esošajiem servisiem, kuri nodarbojas tieši runas valodas identifikāciju, tika atrasts tikai viens serviss *Translated Labs* [45]. Tiek apgalvots, ka modelis atbalsta 8 valodas (tā starpā arī angļu un krievu valodas). Tomēr demo versija, kas atrodas servisa mājaslapā, nav strādājošā. Kā arī piedāvātie API nav pieejami.

Pārējos gadījumos SLID tika piedāvāts tikai kā sastāvdaļa no runas atpazīšanas (ASR, *Automatic Speech Recognition*) pakalpojumiem. Tika atrasti 3 ASR servisi, kur iekšā ir iebūvēts SLID: Amazon Transcribe [3], Google Cloud Speech [15], Azure Speech SDK [4] (sk. 1.1.tabula).

1.1. tabula

ARS servisu salīdzinājums

	Atbalstītas valodas	Angļu ID	Krievu ID	Latviešu ID	Izmaksas
Amazon	31	Ir	Ir	Nav	~ $1.44 / h
Azure	31	Ir	Ir	Nav	~ $1 / h
Google	71	Ir	Ir	Ir	~ $1 / h

Sūtot audio transkripcijas pieprasījumu uz Google Cloud Speech API , var norādīt sarakstu ar valodām, kuras dotais audio varētu saturēt. Pakalpojums mēģina transkribēt audio, pamatojoties uz valodu, kas vislabāk atbilst paraugam no piedāvātajiem variantiem. Tad transkripcijas rezultātus iezīmē ar paredzamo valodas kodu. Lai gan Google Cloud Speech atbalsta mūs interesējošas valodas, API ir domāts runas atpazīšanai, kas šī maģistra darba gadījumā ir lieka darbība. Rezultātā varētu būt liekas izmaksas un lēnāka apstrāde.

Amazon Transcribe runas atpazīšanas pakalpojums atbalsta 31 valodas, tā starpā angļu un krievu valodas. Amazon apgalvo, ka valodas identifikācija notiek daudz ātrāk nekā pati transkripcija. Lai valodas noteikšana būtu ātrāka, var ierobežot valodu sarakstu no kuriem serviss

mēģina to identificēt. Iespēja veikt tikai valodas noteikšanu netiek piedāvāta. Turklāt, latviešu valoda, diemžēl, nav atbalstīta. Tātad, šīs API maģistra darba mērķiem neder.

Microsoft piedāvā valodas noteikšanas pakalpojumus savā Azure Cognitive Services API. Līdzīgi kā iepriekš aprakstītos pakalpojumos, valodas noteikšana ir daļa no kopēja runas atpazīšanas pakalpojuma un tiek izmantota, lai noteiktu, kurā no piedāvātajam valodām visvairāk atbilst dotiem audio datiem. Viena pieprasījuma var padot līdz 4 iespējamam valodām. Valodas noteikšanas pakalpojums pašlaik nav pieejams latviešu valodai.

Svarīgi arī piebilst, ka mākonpakalpojumu izmantošana var izraisīt iespējamas privātuma problēmas, ja audio ieraksti satur konfidenciālu vai īpaši jutīgu informāciju.

Meklējot *GitHub* vietnē pēc atslēgvārdiem "spoken language identification" var atrast 51 repozitoriju. Tomēr nevar atrast neviena gatava modeļa latviešu valodai. Arī citām valodām bieži tiek piedāvāts tikai kods bez gataviem modeļiem.

1.2 Mašīnmācīšanās metodes

Mašīnmācīšanās ir mākslīgā intelekta apakšnozare, kas vērsta uz tādu lietojumprogrammu veidošanu, kuras mācās no datiem un laika gaitā uzlabo to precizitāti bez ieprogrammēšanas to darīt. Mašīnmācīšanās algoritms vai modelis ir statistikas apstrādes darbību secība. Mašīnmācībā modeli tiek “apmācīti”, lai atrastu šablonus, likumsakarības un funkcijas lielos datu apjomos, lai pieņemtu lēmumus un veiktu prognozes, pamatojoties uz jauniem datiem. Jo vairāk datu ir apmācībā, jo labāks būs modelis, jo precīzāki kļūs lēmumi un prognozes. Mašīnmācību lietojumprogrammas (vai modeļa) veidošanai ir četras pamatdarbības:

1. Apmācības datu kopas izvēlēšana un sagatavošana.
2. Modeļa un algoritma izvēlēšana apmācībai uz datu kopas.
3. Modeļa apmācība.
4. Modeļa izmantošana un uzlabošana.

Izmantojot mērķa valodas kā klases, runas valodas noteikšanu var uzskatīt arī par klasifikācijas problēmu. Pazīmes, kuras var izmantot klasifikatora modeļa izveidošanai, varētu būt frāze, vārds, zilbe, fonēma, vai paša runas signāla frekvences izmaiņas. SLID metodes var atšķirties pēc klasifikatora modeļa izveidošanai izmantoto pazīmju sarežģītības. No viszemākā līdz

augstākajam abstrakcijas līmenim ir akustiskās, prozodiskās, fonotaktiskās, leksiskās un sintaktiskās metodes.

Lielākā daļa LID sistēmu darbojas divos posmos: apmācība un klasifikācija [29]. Apmācības posmā tipiskā sistēma tiek iepazīta ar runas piemēriem dažādās valodās. Dažām sistēmām nepieciešami tikai digitalizēti runas izteikumi un attiecīgas valodu patiesās identitātes. Sarežģītākām LID sistēmām var būt nepieciešama fonētiskā transkripcija un/vai ortogrāfiskā transkripcija katram apmācības izteikumam kopā ar izrunu vārdnīcu. Šo transkripciju un vārdnīcu izgatavošana ir dārgs un laikietilpīgs process, kuram parasti ir nepieciešams lingvists, kurš brīvi pārvalda interesējošo valodu. Katrai valodai tiek analizēti apmācības dati un tiek izveidots viens vai vairāki modeļi. Šie modeļi ir paredzēti atspoguļot dažas valodas pamatīpašības, kurus pēc tam var izmantot valodas noteikšanā.

Akustiskās, cilvēka balss radītās, gaisa spiediena svārstības ir grūti apstrādājamas un satur daudz liekas un nevajadzīgas informācijas, kas tikai traucē valodas noteikšanai. Tāpēc tiek veikta pazīmju izgūšana. Pazīmju izgūšanas laikā no gaisa spiediena svārstībā tiek iegūti pazīmju vektori, kuri satur valodai un runai specifisku informāciju. Pazīmju vektoru izmantošana palīdz atbrīvoties no fona trokšņa, augstfrekvences skaņām un citām iezīmēm, kuras nav raksturīgas cilvēka balsij, saglabājot nepieciešamo informāciju kompaktā vektorā, kurš ir daudz piemērotāks statistiskai apstrādei [16,9]. Populāras audio signāla pazīmju vektora izvilkšanas metodes ir LPCC jeb linear predictive cepstral coefficients [40], PLP jeb perceptual linear predictive [18], MFCC jeb Mel-frequency capstral coefficients [5].

1.2.1 Fonotaktiska pieeja

Fonotaktika (Phonotactic) ir lingvistikas nozare, kas nodarbojas ar ierobežojumiem valodā attiecībā uz pieļaujamajām fonēmu kombinācijām. Katru valodu veido fonēmas, kas ir atšķirīga skaņu vienība šajā valodā. Uz fonēmām balstās vairākas prozodiskās un akustiskās pazīmes, kas kļūst par pamata pazīmēm, no kurām atkarīga statistiskā modeļa veiktspēja [38,33]. Ja divām valodām ir daudz fonēmu, kas pārklājas, to identificēšana klasifikatoram kļūst par sarežģītu uzdevumu.

Indijas nacionālā zinātņu un tehnoloģiju institūta (NIST) 1996. gadā veiktie oficiālie novērtējumi parādīja, ka visveiksmīgākajā pieejā automātiskai valodas noteikšanai tiek izmantots

runas signāla fonotaktiskais saturs. Uz fonēmām balstītās sistēmās parasti izmanto (1) fonēmu atpazinējus, kuri izdot fonēmu virknes, un (2) n-gramu valodas modeļus, kuri aptver fonēmu savstarpējas sakarības (jeb fonotaktiku). Alternatīva LID pieeja balstās uz Gausa maisījuma modeļiem (GMM), kas klasificē valodas, izmantojot runas signālu akustiskās īpašības.

1.1. att. **PRLM metodes darbošanās princips** [29]

SLID sistēma ar PRLM (*Phone Recognition Followed by Language Modelling*) [29] pieeju vispirms atpazīst fonēmu no runas signāla straumes un pēc tam klasificē fonēmu pēc mērķa valodām (sk. 1.1. att.), izmantojot statistisko modelēšanu. Šajā sistēmā tiek izmantots viens fonēmas atpazinējs, neatkarīgi no tā, kādā valodā ir runas signāls. Veidojot fonēmu n-gramu statistisko modeli, šo fonēmu atpazinēju izmanto kā universālo (valodai neatkarīgo) atpazinēju. Tas tiek darīts, aprēķinot universāla atpazinēja fonēmu secību parādīšanās varbūtību katrai valodai. Klasificēšanas posma tiek meklēts n-gramu modelis, kurš atpazītai fonēmu virknei piešķir vislielāko varbūtību, un šī modeļa valoda tiek izvēlēta kā audio ieraksta valoda.

Galvenā atšķirība starp PRLM un PPRLM (*Parallel Phone Recognition Followed by Language Modelling)* [29] ir izmantoto fonēmu atpazinēju skaitā. PPRLM metodē ir vairāki fonēmu atpazinēji. Katrs fonēmu atpazinēji ir apmācīts noteikt kādas konkrētas valodas fonēmas. Statistikas n-gram modelis katrai valodai tiek veidots izmantojot fonēmu virknes no šis valodas fonēmu atpazinēja. Tad klasificēšanas laikā audio signāls tiek padots uz vairākiem fonēmu atpazinējiem. Katrai fonēmu virknei no atpazinējā tiek aprēķinātas logaritmiskās varbūtības izmantojot šis valodas fonēmu n-gramu modeļi. Šīs varbūtības tiek salīdzinātas ar visu citu valodu varbūtības vērtībām. Maksimālā vērtība nosaka, kuru valodu lieto runātājs.

Ir veikti vairāki pētījumi SLID jomā, kuros savā starpā tika salīdzinātas dažādas fonotaktikas metodes. Piemēram, Pāvels Matejka, et al [36] ir izmantojuši PRLM un PPRLM, lai atšķirtu divpadsmit valodas, izmantojot 3-gramu modeļus. Viņi atklāja, ka PRLM sistēmas ar vairāk apmācības datiem pārspēj PPRLM sistēmas.

1.2.2 Runas signāla akustiska satura klasificēšana

Lai gan uz fonēmām balstītas sistēmas nodrošināja labāko valodas identifikācijas veiktspēju, to lielās skaitļošanas prasības var neļaut to izmantot zemu izmaksu reāllaika lietojumprogrammās. Alternatīva pieeja LID izmanto Gausa maisījuma modeļus (GMM), lai klasificēt valodas, izmantojot runas signāla akustisko saturu.

Valodas noteikšanas pieejai pēc GMM metodes pamatā ir tas, ka dažādām valodām ir atšķirīgas skaņas un skaņas frekvences. GMM sistēmu ir viegli apmācīt, jo tai nav nepieciešama mācību ortogrāfiskā un fonētiskā marķēšana. Šis idejas pieeja ir izmantot GMM, lai tiešā veidā modelēt pazīmju vektoru varbūtības sadalījumus. Apmācības gaitā, katrai valodai izveido GMM modeli. Atpazīšanas laikā nezināms runas izteikums tiek klasificēts, vispirms pārveidojot digitalizēto viļņu formu par vektoriem un pēc tam aprēķinot varbūtību, ka valodas specifiskais GMM radīja nezināmu runas izrunu. Maksimālā vērtība nosaka, kuru valodu lieto runātājs.

Lai gan GMM sistēmas ir diezgan efektīvas, tās nenodrošina tādu izcilu veiktspēju, kā uz fonēmu balstītas LID sistēmas[29]. Tika piedāvāta fonotaktiskās pieejas variācija [35], kurā ienākošās runas apzīmēšanai tika izmantots GMM, nevis fonēmu atpazinējs. Šī pieeja radīja GMM LID sistēmu, kuras veiktspēja bija konkurētspējīga ar fonēmām balstītām pieejām, bet kuras darbība bija daudz ātrāka.

Literatūrā ir aprakstītas ir vairākas citas metodes, kuri veic valodas klasificēšanu no akustiska satura. Gazeau et al. [46] savā pētījumā parādīja, kā neironu tīklus, atbalsta vektoru mašīnu (*Support Vector Machine*) un slēpto Markova modeli (HMM) var izmantot, lai identificētu franču, angļu, spāņu un vācu valodas. Datu kopa tika sagatavota, izmantojot balss paraugus no Youtube News un VoxForge [47] datiem. Slēptie Markova modeļi pārveido runu vektoru secībā, tika izmantoti, lai uztvertu runas laika iezīmes. HMM, kas apmācīti VoxForge datu kopā, vislabāk darbojās salīdzinājumā ar citiem modeļiem, kurus viņš piedāvāja tajā pašā VoxForge datu kopā. Precizitāte bija 70,0%.

Valodas atkarīgo pazīmju ieguve bija populāra pieeja runāto valodu klasificēšanai [20,11,27]. Sekojot šiem panākumiem, tika ierosināts izmantot i-vektorus kā iezīmes dažādos klasifikācijas tīklos. I-vektori ir uz datiem-bāzēta pieeja pazīmju iegūšanai, kas nodrošina elegantu un vispārīgu sistēmu audio klasifikācijai un identificēšanai. Tas ir fiksēta garuma mazdimensionāls vektors, kas tiek iegūts katram izteikumam. Priekšrocība ir tāda, ka i-vektora modeli var apmācīt

bez uzraudzības (nenorādot runātāja/valodas identitāti runātāja/valodas atpazīšanai). No otras puses, i-vektors satur informāciju gan par klasi, gan par kanālu.

I-vektora pieejas koncepcija vispirms tika ieviesta runātāju verifikācijas kontekstā [31,32] un pēc tam veiksmīgi piemērota daudziem citiem ar runu saistītiem uzdevumiem. Viens no tiem ir runātas valodas noteikšana, kur pirms dziļas mācīšanas veiksmes, i-vektori kļuva par dominējošo pieeju.

1.3 Dziļas mācīšanās metodes

Dziļa mācīšanās ir mašīnmācīšanās apakškopa. Visa dziļā mācīšanās ir mašīnmācīšanās, tomēr ne visa mašīnmācīšanās ir dziļa mācīšanās. Dziļas mācības algoritmi definē mākslīgu neironu tīklu, kas paredzēts, lai uzzinātu, kā cilvēka smadzenes mācās. Dziļa mācīšanās modeļiem ir nepieciešams liels datu daudzums, kas iziet cauri vairākiem aprēķinu slāņiem, katrā secīgajā slānī piemērojot svaru un novirzes, lai pastāvīgi pielāgotu un uzlabotu rezultātus.

Gadu gaitā pētījumos ir izmantotas daudzas prozodiskās un akustiskās pazīmes, lai izveidotu mašīnmācīšanās modeļus LID sistēmām (sk. iepriekšējo sadaļu). Pēdējo gadu laikā LID problēma tika risināta, izmantojot i-vektora shēmu, kas arī atbilst vismodernākajām pieejām runātāju atpazīšanas uzdevumos. Tomēr i-vektora pieeju veiktspēja ievērojami samazinās, strādājot ar īsiem testa izteikumiem [22]. Mūsdienās lielākā daļa runas valodas identificēšanas mēģinājumu paļaujas uz dziļiem neironu tīkliem. Sistēmas, kuru pamatā ir tādas dziļas mācīšanās pieejas, kā padziļināti neironu tīkli (*Deep Neural Networks*) vai garās īslaicīgas atmiņas (*Long Short-Term Memory*) rekurentais neironu tīkls (RNN), ir pierādījuši, ka pārspēj i-vektora pieejas [24].

Revay et al. [41] valodu klasificēšanai izmantoja ResNet50 [26] arhitektūru, ģenerējot katra neapstrādātā audio log-Mel spektrus. Modelis izmanto ciklisku mācīšanās ātrumu, kur mācīšanās ātrums lineāri palielinās un pēc tam samazinās. Maksimālais cikla apguves ātrums tiek noteikts, atrodot optimālo mācību ātrumu, izmantojot *fastai* bibliotēku. Modelis klasificēja sešas valodas - angļu, franču, spāņu, krievu, itāļu un vācu - un sasniedza 89,0% precizitāti.

Bartz et al. [7] valodas identificēšanai piedāvāja divus dažādus hibrīdus konvolūciju rekurentus neironu tīklus (*Convolutional Recurrent Neural Networks*). Viņi ierosināja jaunu arhitektūru telpisko pazīmju iegūšanai no neapstrādāta audio log-Mel spektriem, izmantojot CNN,

un pēc tam RNN, laika pazīmes fiksēšanai, lai identificētu valodu. Šis modelis *Youtube* ziņu datu kopā sasniedza 91,0% precizitāti.

Konvolūciju neironu tīkli (*Convolutional Neural Networks,* CNN) satur vienu vai vairākus konvolūcijas slāņus un *max-pooling* slāņus. Konvolūcijas slānis sastāv no filtriem, kas apstrādā daļu no ieejas signāla. CNN ir parādījuši perspektīvus rezultātus dažādu fonēmu atpazīšanas un atslēgvārdu noteikšanas (KWS) uzdevumos [34,19].

Bartz et al. otrajā arhitektūrā izmantoja arhitektūru Inception-v3, lai iegūtu telpiskas pazīmes, kuras pēc tam tika izmantotas kā ievade divvirzienu LSTM (*Long Short-Term Memory*) neironu tīklam, lai precīzi paredzētu valodu. Šis modelis sasniedza 96,0% precizitāti četrās valodās - angļu, vācu, franču un spāņu.

Montavons [17] sava pētījumā izmantoja laika aiztures neironu tīklu (*time-delay neural network*). Šajā tīklā bija divdimensiju konvolūciju slāņi pazīmju ieguvei. Šajā pētījumā tika sniegta detalizēta analīze par to, kā dziļas arhitektūras pārspēj savus mazākus analogus. Šajā darbā tika konstatētas arī grūtības klasificēt uztveres ziņā tādas līdzīgas valodas, kā vācu un angļu. Ar šo metodi datu kopai, kas sastāv no 3 valodām - angļu, franču un vācu, bija iespējams sasniegt 91,2% precizitāti.

Arvien populārākā kļūst runas valodas noteikšana, izmantojot x-vectors [14]. Tas ir dziļās mācīšanās pieeja, kas bija paredzēta [13] runātāja atpazīšanai, bet paradīja sevi labi arī SLID uzdevumiem. X-vectors var uzskatīt par vienkāršu runas dziļas reprezentācijas veidu, kas satur informāciju par runātāju un valodu.

Tomēr uz dziļas mācīšanas balstītām sistēmām nepieciešamas milzīgas apmācību datu kopas, lai gūtu panākumus. Turklāt to apmācība ir dārga, un tiem joprojām ir daudz apmācāmo parametru.

1.4 Runas dziļas reprezentācijas

Mūsdienu runas atpazīšanas modeļiem ir nepieciešams liels daudzums transkribētu audio datu, lai sasniegtu labu veiktspēju. Nesen neironu tīklu priekšapmācība (*pre-training*) ir kļuvusi par efektīvu metodi gadījumos, kad marķēto datu ir maz. Galvenā ideja ir iemācīties vispārīgas reprezentācijas citos uzdevumos, kur ir pieejams ievērojams daudzums marķētu vai nemarķētu datu, un izmantot apgūtas reprezentācijas, lai uzlabotu turpmāko uzdevumu veiktspēju, kuriem ir ierobežots datu apjoms.

Viens no modeļiem, kas izmanto iepriekš aprakstīto metodi runas audio datiem ir wav2vec [42], kuru izgudroja *Facebook Research* pētnieki. Wav2vec ir pašpārraudzīts algoritms, kas izmanto neapstrādātu, nemarķētu audio bez rakstiskas transkripcijas, lai apmācītu automātiskās runas atpazīšanas (ASR) modeļus. *Facebook* pētnieciskā darbā ietvaros šī pašpārraudzītā pieeja pārspēj tradicionālās ASR sistēmas, kas paļaujas tikai uz transkribēto audio, izmantojot divreiz mazāk marķētus datus. Ir svarīgi samazināt nepieciešamību pēc manuāli anotētiem datiem, lai izstrādātu sistēmas, kas saprot ne tikai angļu un citas populāras valodas, bet arī tos, kuriem ir ierobežoti transkribēts runas apmācības kopas.

Wav2vec apmāca modeļus, liekot tiem izvēlēties starp oriģinālajiem runas piemēriem un modificētajām versijām. Katrā 10 sekunžu ilgajā audio klipā, kurā modelis tiek apmācīts, wav2vec ģenerē vairākus traucējošos piemērus, kas maina 10 ms sākotnējā audio ar sekcijām no citas vietas klipā. Pēc tam modelim jānosaka, kura versija ir pareiza. Algoritms atkārto šo uzdevumu vairākas reizes katrai audio sekundei un prognozē pareizas audio milisekundes nākotnē.

Wav2vec pārvērš neapstrādātus runas piemērus par reprezentācijām (kodu), kurus var nodot esošai ASR sistēmai. Wav2vec attēlojumu izmantošana kā ievade ļauj algoritmam strādāt ar plašu esošo runas atpazīšanas modeļu klāstu.

Nesen Facebook ierosināja jaunu modeli wav2vec 2.0 [1]. Galvenās šī modeļa atšķirības no wav2vec 1.0 ir:

- Jaunais modelis apgūst pamata runas vienības, kas tiek izmantotas, lai risinātu pašpārraudzītu uzdevumu.
- Wav2vec 2.0 ir daudzvalodu variants, kas ir apmācīts 12 valodās no *Mozilla Common Voice* [10].
- Modelis ir apmācīts prognozēt pareizu runas vienību maskētām audio daļām, līdzīgi kā ļoti veiksmīgs *BERT* [23] valodas modelis.

Wav2vec gan 1.0, gan 2.0 var izmantot ne tikai runas atpazīšanas uzdevumiem. Pēdējie pētījumi parāda, ka wav2vec2 var veiksmīgi izmantot runas valodas noteikšanai, sasniedzot ļoti augstu precizitāti [8, 23].

Kad wav2vec2 parādīja izcilo sniegumu, *Facebook AI* prezentēja wav2vec2-xlsr [2]. XLSR apzīmē daudzvalodu runas reprezentācijas (*cross-lingual speech representations*) un attiecas uz

wav2vec2-xlsr spēju iemācīties runas reprezentācijas vairākās valodās. Daudzvalodu mācīšanās ir vērsta uz tādu modeļu veidošanu, kuros veiktspējas uzlabošanai tiek izmantoti citu valodu dati. Wav2vec2-xlsr-53 modelis ir iemācīts uz 53 valodām un 56 tūkstošu stundu runas datiem no *CommonVoice* [10], daudzvalodu *LibreSpeech* [28] un *BABEL*. Šī liela modeļa mērķis ir stimulēt zemu resursu runas izpratnes izpēti.

Autori parāda, ka automātiskas runas atpazīšanas (ASR) modeļa iepriekšēja apmācība ar daudzvalodu nemarķētiem runas datiem, kam seko pietrenēšana (*fine-tuning*) ar konkrētas valodas nelielu marķētu datu daudzumu, ļauj sasniegt konkurētspējīgus rezultātus.

1.5 Datu pieejamība

Lai apmācīt un novērtēt jebkurus mašīnmācīšanās modeļus ir nepieciešami dati. SLID gadījumā ir nepieciešami interesējošas valodas runas ieraksti. Audio ierakstiem jābūt "sabalansētiem", jābūt apmēram vienāds ierakstu apjoms katrai valodai. Ierakstiem arī jābūt "no vienas jomas". Ja vienas valodas ieraksti būs no audio grāmatām, bet citas valodas no telefonierakstiem, tad modelis iemācīsies nevis identificēt valodu, bet atšķirt telefonu zvanu no audio grāmatas. Kā arī, ja modelis ir apmācīts uz audiogrāmatām, bet to ir paredzēts izmantot zvanu centra arhīvā, tad precizitāte būs ļoti zema.

Tālāk tiek apskatītas publiski pieejamas runas valodas datu kopas, kuras satur ierakstus mūs interesējošas valodās – angļu, latviešu un krievu. Tika atrastas lielākas datubāzes ar audio ierakstiem – *Common Voice* [10], *LibriSpeech* [28] un *OpenSTT* [39] (sk. 1.2.tabula).

1.2.tabula

Publiski pieejamas valodas datu kopas salīdzinājums.

	Angļu	Latviešu	Krievu
Common Voice	1`700 h	6 h	115 h
LibriSpeech	1`000 h	0	0
OpenSTT	0	0	20`000 h

Kopējā balss (Common Voice) ir Mozilla projekts, lai izveidotu bezmaksas datu bāzi runas atpazīšanas programmatūrai. Tas ir atvērtā koda daudzvalodu datu kopa, kuru ikviens var izmantot, lai apmācītu runas lietojumprogrammas. Projektu atbalsta brīvprātīgie, kuri ieraksta teikumu

paraugus ar mikrofonu un pārskata citu lietotāju ierakstus. Katrs datu kopas ieraksts sastāv no unikāla MP3 faila un atbilstoša teksta. Atšifrēti teikumi tiks apkopoti balss datu bāzē, kas ir pieejama ar CC0 (Tiesības nav rezervētas) licenci. Šī licence ļauj ar autortiesībām vai ar datubāzēm aizsargāta satura īpašniekiem atteikties no šīm interesēm savos darbos un tādējādi ievietot tās publiskā domēnā, lai citi varētu brīvi izmantot darbus jebkādiem mērķiem bez ierobežojumiem saskaņā ar autortiesību vai datubāzes likumiem. Datubāze satur datus apmēram 60 valodām un kopējais pārbaudītu ierakstu apjoms ir vairāk par 7000 stundām. No tām: 6 pārbaudītas stundas kopa latviešu valodai, 115 pārbaudītas stundas kopa krievu valodai un 1,7k stundas – angļu valodai.

LibriSpeech ir lasītas angļu valodas runas korpuss, kas piemērots runas atpazīšanas sistēmu apmācībai un novērtēšanai. LibriSpeech korpuss ir iegūts no audiogrāmatām, kas ir daļa no LibriVox projekta, un satur 1000 stundas runas, kas atlasīta 16 kHz frekvencē. Šis korpuss ir brīvi pieejams lejupielādei.

OpenSTT ir lielākā publiskā krievu valodas runas pārvēršana tekstā (Speech-to-Text) datu kopa, kas ir piemērota ASR sistēmu apmācībai. Datubāzei ir plašs domēnu klāsts, tas ir radio, audiogrāmatas, *YouTube*, telefona zvani u.c. *OpenSTT* ir pārrakstītu aptuveni 20 000 stundu transkribēto audio datu kopums wav formātā.

Eksistē arī citi gan publiski, gan privāti audio datu avoti dažādam valodām. Var izmantot ierakstus no audiogrāmatām, *YouTube*, radio u.c.. Latviešu valodai ir iespējams izmantot Latvijas republikas Saeimas sēžu audio un video ierakstus, kuri ir publiski pieejami Saeimas mājaslapā. Lai izmantotu šos datus, vajag implementēt programmatūru šādu datu savākšanai. Turklāt vajag nodrošināt, lai savāktie dati ir "sabalansēti" un ir no vienas jomas. Tādu rīku izstrāde ir ārpus šī darba konteksta.

1.6 Kopsavilkums

Šīs darba sadaļas mērķis bija izpētīt SLID metodes, koncentrējoties uz 3 Latvijā runājamām valodām: latviešu, krievu, angļu. Tika izpētītas vairākas pieejas: esošu komerciālo pakalpojumu API izmantošana, mašīnmācīšanās, dziļa mācīšanas un runas dziļu reprezentāciju vektori.

No pieejamiem API vai servisiem, kas veic valodas noteikšanu, visas trīs valodas atbalsta tikai *Google Cloud Speech*, tomēr serviss veic runas atpazīšanu, kas varētu būt lieka darbība, kā arī mākonpakalpojumu izmantošana var izraisīt iespējamas privātuma problēmas, ja audio ieraksti satur konfidenciālu vai īpaši jutīgu informāciju.

Mašīnmācīšanās pieeja ir pārāk sarežģīta, dārga un laikietilpīga, kurai bieži ir nepieciešams lingvists, kurš brīvi pārvalda interesējošo valodu. Kā arī šī metode ir novecojusi un tai ir sliktākā precizitāte.

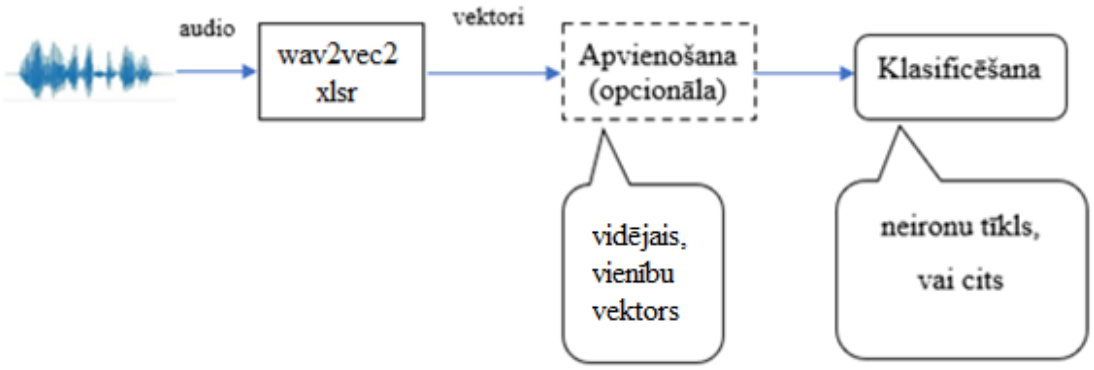
Dziļas mācīšanas pieejai ir nepieciešams liels datu apjoms. Tomēr latviešu valodai ir pieejams tikai viena publiska datu kopa, ar 6h runas ierakstiem. Papildus datus latviešu valodai ir iespējams savākt no saeimas video ierakstiem vai meklēt latviešu valodas runu *YouTube* lietotnē, kas prasa papildus laika ieguldījumus. Iespējams, ka šī metodei būs arī nepieciešams dārgais GPU (grafiskais procesors), lai apmācītu modeļus.

Ņemot vērā darba izvirzītos mērķus, pieejamus datu un skaitļošanas resursus, kā arī nepieciešamu kompetenci, tika secināts, ka vispiemērotākais risinājums ir klasificēšanas modelis, kas darbojas uz daudzvalodu runas dziļas reprezentācijām.

2. IEROSINĀTĀ PIEEJA

2.1 Pieejas apraksts

Ierosinātas pieejas ideja ir apmācīt parasto mašīnmācīšanās klasificēšanas modeli uz wav2vec2-xlsr modeļa daudzvalodu runas dziļām reprezentācijām. Risinājuma ideja ir paradīta 2.1. attēlā.



2.1. att. **Ierosinātā risinājuma ideja**

Audio ieraksts wav formātā tiek padots uz daudzvalodu wav2vec2-xlsr, kurš izdod daudzvalodu dziļas runas reprezentācijas jeb vektorus. Tālāk šos vektorus var apvienot vienā vektorā, kas reprezentē visu ierakstu. Tas var būt normalizētu vektoru summa vai vidējais aritmētiskais. Tad seko klasiskais klasificēšanas modelis, kas noteic ieraksta valodu.

Izvirzītas pieejas priekšrocības ir sekojošas:

- Nevajag lingvistiskas vai fonotaktiskas zināšanas.
- *Facebook* [1] apgalvo, ka 10 min ir pietiekami priekš automātiskas runas atpazīšanas modeļa apmācīšanas ar wav2vec2. Tas nozīme, ka mums ir pietiekami daudz datu katrai valodai (sk. sadaļu 1.5), lai uztrenētu labu valodas noteikšanas modeli.
- Internetā ir brīvi pieejami vairāki iepriekš apmācīti wav2vec2 modeļi un piemēri.
- Mūsdienās lai uzprogrammēt parastu mašīnmācīšanās klasifikatoru nevajag lielu iepriekšējo pieredzi.

- Nav nepieciešamas dārgas GPU iekārtas.

2.2 Eksperimenta dizains

Praktiskā daļā tika lemts izmantot audio ierakstus no *CommonVoice* datubāzes, lai visām valodām ieraksti būtu sabalansēti un no vienas jomas. Tas ir nepieciešams lai modelis neiemācītos pareģot vispārstāvētāko valodu vai atpazīt jomas nevis valodas.

Tā kā latviešu valodai ir pieejama 7 stundu datu kopa, lai visām valodām dati būtu sabalansēti, no krievu un angļu datu kopām tika nejauši izvēlētas 7 stundas. Tālāk dati tiek nejauši sadalīti pēc sekojoša principa:

- 500 audio ieraksti no katras valodas tiks izmantota novērtēšanas vai testēšanas nolūkiem,
- Visi atlikuši audio dati ir atvēlēti modeļu apmācībai.

CommonVoice lūdz ikvienu, kurš lejupielādē datu kopu, ievērot līdzautoru privātumu, lai aizsargātu visus ieguldītājus drošību. Tas nozīmē, ka dalot datus nav iespējams nodrošināt lai runātāji apmācības un novērtēšanas kopās nepārklājās. Tomēr, darbā izmantotas datu kopas ir pietiekami lielas un daudzveidīgas, kas teorētiski minimizē šo problēmu.

Rezultātā apmācības datu kopa satur sekojošos datus:

- Latviešu valodā ir 7147 audio faili ar kopīgo garumu 06:44:23 un vidējo audio faila garumu 3.4 sekundes.
- Angļu valodā ir 5273 audio faili ar kopīgo garumu 06:09:26 un vidējo audio faila garumu 4.2 sekundes.
- Krievu valodā ir 4367 audio faili ar kopīgo garumu 06:31:03 un vidējo audio faila garumu 5.4 sekundes.

Savukārt, novērtēšanas vai testa kopa sastāv no 1500 audio failiem:

- Latviešu valodā ir 500 failu ar kopīgo garumu 00:27:13.
- Angļu valodā ir 500 failu ar kopīgo garumu 00:35:52.
- Krievu valodā ir 500 failu ar kopīgo garumu 00:44:29.

Lai novērtētu kā modelis strādā rēķināsim precīzumspēju (*precision*), pārklājums (*recall*), F1 un akurātumu (*accuracy*) katrai valodai. Precizitāte tiek definēta kā:

$$Precizitāte = \frac{TP}{TP + FP}$$

kur TP ir pareiza atbilsme (*true positive*) un FP ir kļūdaina atbilsme (*false positive*). Pārklājums tiek definēts kā:

$$Pārklājums = \frac{TP}{TP + FN}$$

kur TP ir pareiza atbilsme (*true positive*) un FN ir kļūdaina neatbilsme (*false negative*). F1 tiek definēts kā:

$$F1 = 2 \times \frac{Precizitāte * Pārklājums}{Precizitāte + Pārklājums}$$

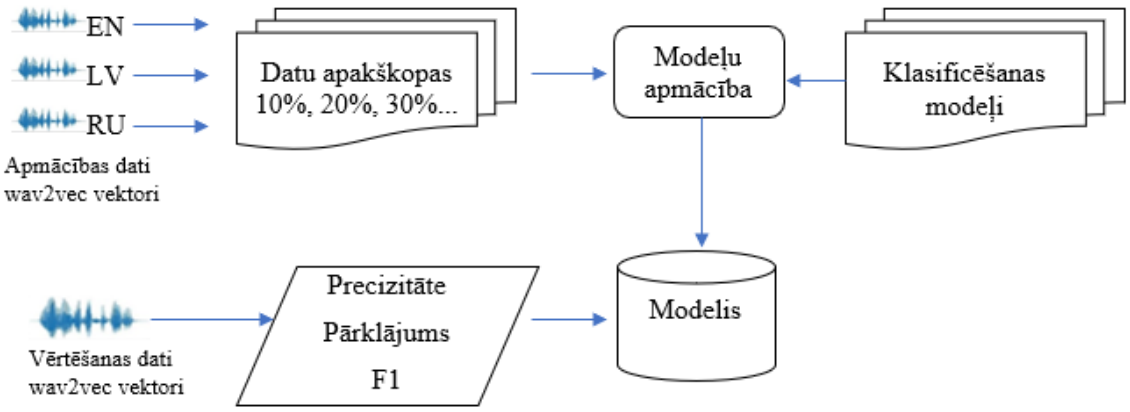
Beidzot, akurātums tiek definēts kā:

$$Akurātums = \frac{TP + TN}{TP + FP + TN + FN}$$

kur TN ir pareiza neatbilsme (*true negative*), FP ir kļūdaina atbilsme (*false positive*) un FN ir kļūdaina neatbilsme (*false negative*).

Attēlā 2.2 ir prezentēta maģistra darba praktiskā daļā veiktu eksperimentu shēma, kuru mērķis ir eksperimentāli noskaidrot:

- Vai ir iespējams veikt runas valodas noteikšanu izvēlētām valodām izmantojot wav2vec2-xlsr runas dziļas reprezentācijas?
- Kā labāk apvienot vairākas runas dziļas reprezentācijas vienā vektorā?
- Kāds ir minimālais datu apjoms, kas ir nepieciešams lai iegūtu pietiekami precīzu modeli?
- Kāds klasificēšanas modelis dotajā scenārijā strādā vislabāk?



att. 2.2. **Eksperimentu shēma**

Lai ietaupīt laiku, eksperimenti ar dažādiem apmācības datu apjomiem tika veikta tikai metodei, kura sasniedz labākos rezultātus.

2.3 Google Cloud Speech-to-Text API

Viens no pieejamiem gataviem risinājumiem ir Google Cloud Speech-to-Text API. Tas ir primārais uzdevums ir pārvērst runu tekstā, bet kā beta versija ir pieejama arī runas valodas noteikšanas funkcija.

Lai notestētu Google valodas noteikšanas funkciju tika izmantota ir Google Cloud Speech-to-Text API v1p1beta1 versija. API izsaukuma parametros var noradīt primāro valodas kodu un alternatīvas valodas (bet ne vairāk kā 4). Tad .wav fails tiek sūtīts uz mākoņservisu un tiek atgriezta runas atpazīšanas rezultāts, kas satur arī noteiktas valodas kodu.

Testējot šo API, tika atklāts, kaut arī Google apliecina, ka latviešu valoda ir viena no atbalstītajām valodām Speech-to-Text API, tomēr tā netiek pareizi identificēta. Norādot kaut vienā vietā, kā primāro valodu, vai kā alternatīvo, latviešu valodu, runas valodas noteikšanas funkcija pārstāj strādāt un API vienmēr atgriezīs primāro valodas kodu.

Tādā veidā kā alternatīvs risinājums tika izmēģināts latviešu valodas vietā izmantot lietuviešu valodas kodu alternatīvo valodu sarakstā, jo tās ir radniecīgas valodas. Šis risinājums deva negaidīti labu rezultātu, kas ir attēlots 2.1. tabulā. Tomēr latviešu valodai ir diezgan mazs pārklājuma rezultāts, kas norāda uz to, ka neliela daļa no failiem latviešu valodā tika pareizi atpazīti un tika bieži sajaukti ar angļu valodu.

2.1. tabula

Google Cloud Speech-to-Text API valodas noteikšanas rezultāti

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
LV	117	15	88	383	88.64%	23.4%	37.03%	73.47%
RU	481	108	892	19	81.66%	96.2%	88.33%	91.53%
EN	480	269	731	20	64.08%	96%	76.86%	80.73%
Vidējais					78.13%	71.87%	67.41%	81.91%

No iegūtiem rezultātiem var secināt, ka valodas noteikšana labi strādā krievu un angļu valodai – tiem ir augsts pārklājuma un F1 rezultāts. Vislabākais F1 rezultāts, kā arī akurātums šinī eksperimentā ir krievu valodai.

2.4 Kosinusa līdzība

Šīs klasificēšanas metodes pamatideja ir salīdzināt līdzību starp failu un valodu vektoriem. Lai noteikt faila runas valodu, ir jāsaskaita faila vektorus un jāsalīdzina to ar vairākiem valodas vektoriem. Kādā valoda ir tuvāk, tāda arī tiek piešķirta failam.

Viens no izplātītākajiem veidiem kā mērīt attālumu starp vektoriem ir kosinusa līdzība (*Cosine similarity*). Kosinusa līdzība ir metrika, kas palīdz noteikt, cik līdzīgi ir vektori neatkarīgi no to lieluma. To mēra pēc leņķa kosinusa starp diviem vektoriem un nosaka, vai divi vektori ir vērsti aptuveni vienā virzienā. Jo mazāks leņķis, jo lielāka ir kosinusa līdzība un vektori ir līdzīgāki viens otram.

Formula, lai atrastu kosinusa līdzību starp diviem vektoriem, ir:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Kā jau bija minēts, lai iegūtu audio faila vektorus, izmantojam wav2vec2-xlsr modeli. Tas izdod dažādu vektoru daudzumu, atkarīgi no dota audio faila garuma. Šajā darbā tika apskatītas 3 metodes, kā ar iegūtiem vektoriem darboties:

1. Izmantojot vienības vektorus (*unit vector*) iegūtos no failu vektoriem. Vienības vektors normētā vektoru telpā ir vektors ar garumu 1, ko dažreiz sauc arī par virziena vektoru. Lai iegūtu valodas vektoru, visi vienas valodas failu vektori apmācības datu kopā tika saskaitīti un rezultāts tika normalizēts. Testēšanas datiem vektori tika summēti un normalizēti tikai viena faila ietvaros. Tad katrs testēšanas faila normalizēts vektors tiek salīdzināts ar valodas vektoru. Līdzīga veidā ir implementēta kosinusa līdzība tekstiem *Shorttext* [43] rīkkopā.

2. Otrā metodē tiek izmantots vienības vektoru vidējais vektors (*average vector of unit vectors*). Apmācības gaitā sākumā tiek izrēķināti vienības vektori katram failam, tad katrai valodai tika saskaitīts vidējais vektors no visiem vienību vektoriem. Savukārt, testa datu vektori tika saskaitīti un normalizēti tādā pati veidā kā ir aprakstīts pirmajā metodē. Šī metode var būt potenciāli laba, jo valodas un testa failu vektori tiek iegūti līdzīgā veidā. Gan apmācības, gan testēšanas datu kopās vienības vektors tiek iegūts no katra faila, nevis visiem valodas failiem kopā.

3. Vēl viena alternatīva metode, kas tika izpētīta, ir vidēja vektora izmantošana. Vidējais vektors tika saskaitīts vispirms katram failam, gan apmācības, gan testēšanas failiem. Tad no apmācības datu vidējiem vektoriem tiek rēķināts vidējais vektors katrai valodai.

2.2. tabula

Kosinusa līdzības metodes rezultāti latviešu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	117	15	88	383	88.64%	23.4%	37.03%	73.47%
Vienību vektors	208	247	753	292	45.71%	41.6%	43.56%	64.07%
Vidējais vienību vektors	208	247	753	292	45.71%	41.6%	43.56%	64.07%
Vidējais vektors	211	238	762	289	46.99%	42.2%	44.47%	64.87%

Tabulā 2.2. ir redzami rezultāti latviešu valodas noteikšanai ar kosinusa līdzības metodi. Labākais rezultāts ir eksperimentam ar vidējiem vektoriem. Vienības vektoriem un vidējiem

vienības vektoriem rezultāts ir identisks. Visos trīs eksperimentos pārklājums un F1 rezultāts ir labāks nekā Google API.

2.3. tabula

Kosinusa līdzības metodes rezultāti krievu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	481	108	892	19	81.66%	96.2%	88.33%	91.53%
Vienību vektors	117	76	924	383	60.62%	23.4%	33.76%	69.4%
Vidējais vienību vektors	117	76	924	383	60.62%	23.4%	33.76%	69.4%
Vidējais vektors	120	83	917	380	59.11%	24%	34.13%	69.13%

Tabulā 2.3. ir redzami rezultāti krievu valodas noteikšanai ar kosinusa līdzības metodi. Vienības vektoru un vidējo vienības vektoru metodēm rezultāts ir identisks, un tiem precizitāte un akurātums ir labāk, nekā vidējo vektoru metodei. Tomēr pārklājums un F1 ir nedaudz labāks metodei ar vidējiem vektoriem. Visos trīs eksperimentos rezultāts nav labāks, salīdzinājumā ar Google API rezultātiem.

2.4. tabula

Kosinusa līdzības metodes rezultāti angļu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
EN	480	269	731	20	64.08%	96%	76.86%	80.73%
Vienību vektors	322	530	470	178	37.79%	64.4%	47.63%	52.8%
Vidējais vienību vektors	322	530	470	178	37.79%	64.4%	47.63%	52.8%
Vidējais vektors	322	526	474	178	37.97%	64.4%	47.77%	53.07%

Angļu valodas noteikšanai ar kosinusa līdzības metodi labākais rezultāts ir eksperimentam ar vidējiem vektoriem. Tabulā 2.4. ir redzams, ka pārklājums visām trim metodēm neatšķiras, tomēr visi pārējie rezultāti ir labāki vidējiem vektoriem. Kā arī citām valodām, vienības vektoru un vidējo vienības vektoru metodēm rezultāts ir identisks. Salīdzinot ar Google API, kosinusa līdzības metodei rezultāts nav labāks angļu valodas noteikšanai.

2.5. tabula

Kosinusa līdzības metodes vidēji rezultāti visām valodām

	Precizitāte	Pārklājums	F1	Akurātums
Google API	78.13%	71.87%	67.41%	81.91%
Vienību vektors	48.04%	43.13%	41.65%	62.09%
Vidējais vienību vektors	48.04%	43.13%	41.65%	62.09%
Vidējais vektors	48.03%	43.53%	42.13%	62.36%

Eksperimentā ar kosinusa līdzības metodi labāko F1 rezultāti izdevās sasniegt angļu valodai. Vidējie rezultāti visām trim metodēm ir līdzīgi, kā var redzēt 2.5. tabulā. 1. un 2. metodēm rezultāts ir identisks, bet 3. metodes rezultāts ir nedaudz labāks par citiem. Tāpēc tālākos eksperimentos tiks izmantoti vidējie vektori.

Pēc iegūtiem rezultātiem var secināt, ka kosinusa līdzības metode dod labāko rezultātu nekā vienkārši minēt vai paredzēt vienmēr vienu valodu. Tomēr rezultāti ievērojami atpaliek no Google API rezultātiem.

2.5 K-tuvāko kaimiņu algoritms

K-tuvāko kaimiņu algoritms *(k-nearest neighbors algorithm)* ir uzraudzīts mašīnmācīšanās algoritms, ko var izmantot gan klasifikācijas, gan regresijas problēmu risināšanai. Uzraudzīts mašīnmācīšanās algoritms ir tāds, kas paļaujas uz iezīmētiem ievades datiem, lai iemācīties funkciju, kas rada atbilstošu izvadi, kad tiek doti jauni nemarķēti dati. Gan klasifikācijas, gan

regresijas gadījumos ievadi veido K tuvākie apmācības piemēri datu kopā. Bet izvade ir atkarīga no tā, vai KNN izmanto klasifikācijai vai regresijai.

Klasifikācijas problēmas izvade ir diskrēta vērtība, kas apzīmē objekta piederību klasei. Standarta prakse izvadi attēlot kā veselu skaitli, piemēram, 1, -1 vai 0. Objektu klasificē pēc tā kaimiņu balsojuma, objektu piešķirot vērtībai, kas ir visizplatītākā starp K tuvākajiem kaimiņiem.

Regresijas problēmas iznākums ir reāls skaitlis. Objektam tiek piešķirta vidējā vērtība no tuvāk esošajiem K objektiem, kuru vērtības jau ir zināmas.

KNN algoritms pieņem, ka līdzīgas lietas pastāv tiešā tuvumā. Tas uztver līdzības ideju, aprēķinot attālumu starp punktiem vektoru telpā.

KNN algoritma darbības princips:

1. Datu ielādēšana,
2. K inicializēšana atbilstoši izvēlētajam kaimiņu skaitam,
3. Katram datu piemēram:
    a. Attālumu aprēķināšana starp vaicājuma piemēru un pašreizējo piemēru,
    b. Attāluma un piemēra rādītāja pievienošana kopai.
4. Kopas sakārtošana pēc attāluma, no mazākajiem līdz lielākajiem,
5. Pirmos K ierakstu izvēlēšana no sakārtotās kopas,
6. Atlasīto K ierakstu marķējuma iegūšana,
7. Rezultātu atgriešana:
    a. Regresijas gadījumā, objektam tiek piešķirta vidējā vērtība, kas tiek sarēķināta no tuvāk esošajiem K objektiem, kuru vērtības jau ir zināmas,
    b. Klasifikācijas gadījuma, objekts tiek piešķirts klasei, kas ir visizplatītākā starp dotā elementa K kaimiņiem, kuru klases jau ir zināmas.

Lai atrastu datiem piemēroto K, KNN algoritms tika palaists vairākas reizes ar dažādām K vērtībām. Tādā veidā ir iespēja izvēlēties to K, kas samazina sastopamo kļūdu skaitu, vienlaikus saglabājot algoritma spēju precīzi prognozēt, kad tam tiek padoti iepriekš neredzēti dati.

KNN modeļa apmācībai tika izmantoti katras valodas datu kopas vidējie vektori. Modeļa testēšanai tika izmantoti katra faila vidējie vektori. Šī vektoru apvienošanas metode tika izvēlēta, jo tā sasniedza labāko rezultātu iepriekšējā eksperimentā.

Apmācības modeļa implementēšanai tika izmantota mašīnmācīšanās bibliotēka *Python* valodai *scikit-learn* un tās KNN klasifikatora implementācija - *KNeighborsClassifier*. Tika veikti vairāki eksperimenti, katru reizi mainot *n_neighbors* (kaimiņu skaits) parametru klasifikatorā.

2.6. tabula

KNN metodes rezultāti latviešu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	117	15	88	383	88.64%	23.4%	37.03%	73.47%
K=3	419	243	757	81	63.29%	83.8%	72.12%	78.4%
K=5	431	283	717	69	60.36%	86.2%	71%	76.53%
K=10	439	314	686	61	58.3%	87.8%	70.07%	75%
K=100	436	379	621	64	53.5%	87.2%	66.31%	70.47%

Latviešu valodai labākais rezultāts ir, kad k = 3 (sk. tabula 2.6). Un jo lielāks ir k, jo sliktāk ir rezultāts. Tomēr labāko pārklājumu izdevās sasniegt ir pie k =10. Salīdzinot ar Google API F1 rezultāts latviešu valodai ir ievērojami labāks.

2.7. tabula

KNN metodes rezultāti krievu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	481	108	892	19	81.66%	96.2%	88.33%	91.53%
K=3	390	202	798	110	65.88%	78%	71.43%	79.2%
K=5	391	188	812	109	67.53%	78.2%	72.47%	80.2%
K=10	384	181	819	116	67.96%	76.8%	72.11%	80.2%
K=100	346	187	813	154	64.91%	69.2%	66.98%	77.27%

Krievu valodai labākais rezultāts sanāca pie k = 5 (sk. tabula 2.7). Līdzīgi kā latviešu valodas rezultātos, šeit arī ir redzama tendence, ka jo lielāks ir k, jo sliktāk ir rezultāts. Turklāt, KNN metodes rezultāti krievu valodai nav tik pat labi, kā Google API rezultāti.

2.8. tabula

KNN metodes rezultāti angļu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	480	269	731	20	64.08%	96%	76.86%	80.73%
K=3	193	53	947	307	78.46%	38.6%	51.74%	76%
K=5	169	38	962	331	81.64%	33.8%	47.8%	75.4%
K=10	156	26	974	344	85.71%	31.2%	45.74%	75.33%
K=100	118	34	966	382	77.63%	23.6%	36.19%	72.27%

Angļu valodai labākais rezultāts, tā pat kā latviešu valodai, ir ar k = 3 (sk. tabula 2.8). Līdzīgi kā iepriekšējām valodām, šeit arī ir redzama tāda tendence – jo lielāks ir k, jo sliktāk ir rezultāts. Salīdzinot ar Google API rezultātiem, precizitāte angļu valodai ir daudz labāka un akurātums ir ļoti tuvs Google rezultātiem, tomēr pārklājums un F1 ir ievērojami mazāki.

2.9 tabula

KNN metodes vidējais aprēķins visām valodām

	Precizitāte	Pārklājums	F1	Akurātums
Google API	78.13%	71.87%	67.41%	81.91%
K=3	69.21%	66.8%	65.1%	77.87%
K=5	69.85%	66.07%	63.76%	77.38%
K=10	70.66%	65.27%	62.64%	76.84%
K=100	65.35%	60%	56.5%	73.33%

Galu galā labākais vidējais rezultāts ir ar k = 3. Ar KNN metodi sanāca iegūt ievērojami labāko rezultātu, nekā ar kosinusa līdzības metodi. Bet joprojām rezultāts nav tik labs kā Google API, lai gan ir jau diezgan tuvu.

2.6 Daudzslāņu perceptrons

Nākamais klasificēšanas modelis, kas tika pārbaudīts šinī maģistra darbā, ir daudzslāņu perceptrona klasifikators jeb MLP (*multilayer perceptron*). Perceptrons ir smadzeņu informācijas

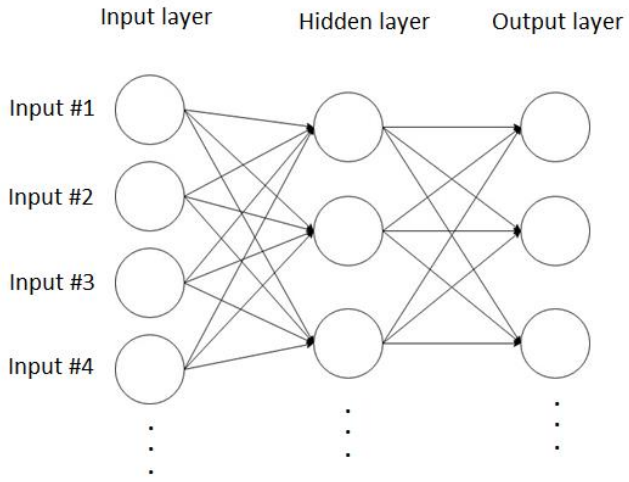
uztveres matemātiskais modelis. Mašīnmācīšanās jomā perceptrons ir bināro klasifikatoru uzraudzītas mācīšanās algoritms. Binārais klasifikators ir funkcija, kas var izlemt, vai ievade, kuru attēlo skaitļu vektors, pieder kādai konkrētai klasei vai nē [37].

Daudzslāņu perceptrons ir mākslīgā neironu tīkla visnekrasākais veids. To veido vairāk nekā viens perceptrons. Vienkāršākais MLP sastāv no vismaz trim mezglu slāņiem (sk. 2.3.att): ievades slāņa signāla saņemšanai, slēpta slāņa, kas ir skaitļošanas dzinējs, un izvades slāņa, kas prognozē ievadi. MLP var saturēt patvaļīgo slēpto slāņu skaitu. Izņemot ievades mezglus, katrs mezgls izmanto nelineāru aktivācijas funkciju [30].

Vēsturiski izplatītās aktivizācijas funkcijas ir hiperboliskais tangenss (tanh) un loģistiskā funkcija (σ):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Jaunākajās dziļās mācīšanās tendencēs visbiežāk tiek izmantota ReLU (*Rectified Linear Unit*) funkcija.



2.3. att. **MLP ar vienu slēptu slāni**

Daudzslāņu perceptroni trenējas uz ieejas-izejas pāru kopam un iemācās modelēt korelāciju vai atkarības starp šīm ieejām un izejām. Daudzslāņu tīklos tiek izmantotas dažādas mācību metodes, no kurām vispopulārākā ir atpakaļizplatīšana (*back-propagation*), kur izejas vērtības tiek salīdzinātas ar pareizo atbildi, lai aprēķinātu kādas iepriekš definētas kļūdas funkcijas vērtību. Pēc

tam, izmantojot dažādas metodes, kļūda tiek atgriezta tīklā. Izmantojot šo informāciju, algoritms pielāgo katra savienojuma parametrus, vai svaru un nobīdi, lai mazinātu kļūdu. Pēc šī procesa atkārtošanas ar daudziem apmācības cikliem tīkls parasti nonāk stāvoklī, kurā skaitļošanas kļūda ir maza.

Šī maģistra darba ietvaros tika izmantots MLP ar vienu slēpto slāni. Vektoriem, ko izdod wav2vec2-xlsr modelis ir 1024 dimensijas, tātad ievades slānī MLP klasifikatorā jābūt 1024 neironiem. Lai pārbaudīto kāds slēpto slāņu izmērs dod labāku rezultātu, tika pārbaudītas dažas vērtības no 100 līdz 1200. Savukārt, izvades slānī ir 3 neironi, kas veido 3 dimensiju vektoru, kur katra dimensija apzīmē katras valodas varbūtību. Eksperimentos tika izmēģinātas 3 dažādas aktivācijas funkcijas: ReLU, hiperboliskais tangenss (tanh) un loģistiskā funkcija (logistic).

MLP modeļa apmācībai tika izmantota mašīnmācīšanās bibliotēka *Python* valodai *scikit-learn* un tās MLP klasifikatora implementācija - *MLPClassifier*. Maksimālais iterāciju skaits tika iestatīts uz 300, bet klasifikators var apstāties arī agrāk, ja apmācības rezultāts neuzlabojas starp iterācijām. Apmācības koeficients (0.001) un citi apmācības parametri tiek izmantoti tādi, kādi tie ir *scikit-learn* pēc noklusējuma.

Sākumā eksperimenta gaitā tika notestētas dažādas aktivācijas funkcijas ar slēpto slāņu izmēru 100. Rezultāti ir apkopoti tabulās 2.10 – 2.13.

2.10. tabula

MLP metodes rezultāti latviešu valodai ar dažādam aktivizācijas funkcijām

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	117	15	88	383	88.64%	23.4%	37.03%	73.47%
ReLU	395	55	945	105	87.78%	79%	83.16%	89.33%
Logistic	412	138	862	88	74.91%	82.40%	78.48%	84.93%
Tahn	387	59	941	113	86.77%	77.40%	81.82%	88.53%

Tabulā 2.10 ir redzami eksperimenta rezultāti latviešu valodai ar dažādam aktivizācijas funkcijām. Labākie rezultāti ir sasniegti ar ReLU funkciju. Bet labākais pārklājums iznāca ar loģistisko funkciju. Pārklājums, F1 un akurātums latviešu valodas datu kopai iznāca daudz labāks,

salīdzinot ar Google API rezultātiem. Precizitāte MLP metodes ir zemāka par Google API rezultātu.

2.11. tabula

MLP metodes rezultāti krievu valodai ar dažādam aktivizācijas funkcijām

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	481	108	892	19	81.66%	96.2%	88.33%	91.53%
ReLU	454	166	834	46	73.22%	90.8%	81.07%	85.87%
Logistic	432	336	664	68	56.25%	86.40%	68.14%	73.07%
Tahn	467	251	749	33	65.04%	93.40%	76.68%	81.07%

Tabulā 2.11 ir atrodami eksperimenta rezultāti krievu valodai ar dažādam aktivizācijas funkcijām. Labākie rezultāti, tā pat kā eksperimentā ar latviešu valodas datu kopu, ir sasniegti ar ReLU aktivizācijas funkciju. Bet labākais pārklājums ir sasniegts ar hiperbolisko tangensu. Salīdzinot ar Google API, MLP metodei rezultāti sanāca sliktāki.

2.12. tabula

MLP metodes rezultāti angļu valodai ar dažādam aktivizācijas funkcijām

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	480	269	731	20	64.08%	96%	76.86%	80.73%
ReLU	401	29	971	99	93.26%	80.2%	86.24%	91.47%
Logistic	173	9	991	327	95.05%	34.60%	50.73%	77.60%
Tahn	326	10	990	174	97.02%	65.20%	77.99%	87.73%

Tabulā 2.12 ir atrodami eksperimenta rezultāti angļu valodai ar dažādam aktivizācijas funkcijām. Labākie rezultāti, joprojām, ir sasniegti ar ReLU aktivizācijas funkciju. Bet labākā precizitāte ir sasniegta ar hiperbolisko tangensu. Salīdzinot ar Google API rezultātiem angļu valodai, MLP metodei precizitāte ir labākā ar visām aktivizācijas funkcijām. Ar ReLU funkciju ir sasniegts labākais F1 un akurātums. Pārklājums MLP metodei ir sliktāk ar visām aktivizācijas funkcijām.

2.13. tabula

MLP metodes vidējais aprēķins visām valodām ar dažādam aktivizācijas funkcijām

	Precizitāte	Pārklājums	F1	Akurātums
Google API	78.13%	71.87%	67.41%	81.91%
ReLU	84.75%	83.33%	83.49%	88.89%
Logistic	75.4%	67.8%	65.78%	78.53%
Tahn	82.95%	78.67%	78.83%	85.78%

Rezumējot, pirmajā eksperimentā ar dažādām aktivācijas funkcijas labākais rezultāts ir sasniegts ar ReLU funkciju (sk. 2.13. tabula). Turklāt, šie rezultāti ir labāki nekā Google API vidējais rezultāts visām valodām.

Noskaidrojot, ka ReLU aktivācijas funkcija dod labākus rezultātus, tālāk tika veikti eksperimenti ar dažādiem slēpta slāņa izmēriem. Rezultāti ir apkopoti tabulās 2.14 – 2.17.

2.14. tabula

MLP metodes rezultāti latviešu valodai ar dažādu slēpto slāņu izmēru

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	117	15	88	383	88.64%	23.4%	37.03%	73.47%
100	395	55	945	105	87.78%	79%	83.16%	89.33%
200	432	96	904	68	81.82%	86.40%	84.05%	89.07%
500	390	45	955	110	89.66%	78.00%	83.42%	89.67%
1200	468	166	834	32	73.82%	93.60%	82.54%	86.80%

Eksperimentā ar dažādiem slēptiem slāņiem izmēriem latviešu valodai labākais rezultāts, pēc F1, ir ar slēptā slāņa izmēru 200 (sk. 2.14. tabula). Labākais pārklājums ir sasniegts ar slēptā slāņa izmēru 1200. Salīdzinot ar Google API rezultātiem latviešu valodai, pārklājums, F1 un akurātums ir labāks MLP metodei ar visām slēpto slāņu izmēriem. Precizitāte ir labāka tikai ar slēptā slāņa izmēru 500.

2.15. tabula

MLP metodes rezultāti krievu valodai ar dažādu slēpto slāņu izmēru

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	481	108	892	19	81.66%	96.2%	88.33%	91.53%
100	454	166	834	46	73.22%	90.8%	81.07%	85.87%
200	437	134	866	63	76.53%	87.40%	81.61%	86.87%
500	468	188	812	32	71.34%	93.60%	80.97%	85.33%
1200	415	150	850	85	73.45%	83.00%	77.93%	84.33%

Eksperimentā ar krievu valodu labākais rezultāts ir, tā pat kā latviešu valodai, ar slēptā slāņa izmēru 200 (sk. 2.15. tabula). Labākais pārklājums ir sasniegts ar slēptā slāņa izmēru 500. Tomēr salīdzinot ar Google API rezultātiem krievu valodai, šī eksperimenta rezultāti nav labāki par to.

2.16. tabula

MLP metodes rezultāti angļu valodai ar dažādu slēpto slāņu izmēru

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	480	269	731	20	64.08%	96%	76.86%	80.73%
100	401	29	971	99	93.26%	80.2%	86.24%	91.47%
200	383	18	982	117	95.51%	76.60%	85.02%	91.00%
500	392	17	983	108	95.84%	78.40%	86.25%	91.67%
1200	296	5	995	204	98.34%	59.20%	73.91%	86.07%

Tabulā 2.16. ir apkopoti eksperimenta rezultāti ar angļu valodu. Labākais rezultāts ir sasniegts ar slēptā slāņa izmēru 500, jo ir vislabākie F1 un akurātuma radītāji. Labākais pārklājums ir sasniegts ar slēptā slāņa izmēru 100. Salīdzinot ar Google API, šajā eksperimentā precizitāte un akurātums angļu valodai ir labāki ar visām slēpto slāņu izmēriem.

2.17. tabula

MLP metodes vidējais aprēķins visām valodām ar dažādu slēpto slāņu izmēru

	Precizitāte	Pārklājums	F1	Akurātums
Google API	78.13%	71.87%	67.41%	81.91%
100	84.75%	83.33%	83.49%	88.89%
200	84.62%	83.47%	83.56%	88.98%
500	85.61%	83.33%	83.55%	88.89%
1200	81.87%	78.6%	78.13%	85.73%

Rezumējot, otrajā eksperimentā ar dažādiem slēpta slāņa izmēriem labākais rezultāts ir sasniegts ar izmēru 200. Tabulā 2.17. var redzēt, kā vidējais aprēķins visām valodām ir labāks par Google API rezultātu ar visām slēpto slāņu izmēriem, kas tika pārbaudīti šajā eksperimentā.

2.7 Modeļu pietrenēšana

Precīza pieregulēšana vai pietrenēšana (*Fine-tuning*) ir ļoti cieši saistīta ar tādu uzdevumu kā zināšanu pārnešana (*transfer learning*). Neironu tīklu zināšanu pārnešana nozīme, ka zināšanas (neironu tīklu modeļa svari), kas iegūtas, risinot vienu problēmu, tiek pielietotas jaunai, bet saistītai problēmai.

Precīzāk, pietrenēšana ir process, kurā tiek izmantots modelis, kas jau ir apmācīts vienam konkrētam uzdevumam. Šo modeļi modificē vai pielabo, lai tas veiktu otru līdzīgu uzdevumu. Piemēram, zināšanas, kas iegūtas, iemācoties atpazīt runu (pārvaidot runu tekstā), varētu izmantot runas valodas atpazīšanas problēmā.

Pieņemot, ka sākotnējais uzdevums ir līdzīgs jaunajam uzdevumam, jau izstrādāta un apmācīta mākslīgā neironu tīkla izmantošana ļauj izmantot to visu, ko modelis jau ir apguvis, neizstrādājot to no nulles.

Izplatītākā prakse ir apgriezt iepriekš apmācītā tīkla pēdējo slāni un aizstāt to ar jauno slāni, kas attiecas uz jauno problēmu. Ir arī ierasta prakse iesaldēt iepriekš apmācītā tīkla dažu pirmo slāņu svarus, pieņemot, ka daži pirmie slāņi uztver tādas universālas funkcijas un īpašības, kas attiecas arī uz jauno problēmu un tos vajag saglabāt neskartus. Tā vietā tīkls koncentrēsies uz jauna uzdevumu raksturīgo funkciju apguvi nākamajos slāņos.

Pašlaik mašīnmācīšanās jomā modeļu pietrenēšana ir populāra pieeja, kā izstrādāt klasifikatorus. Tiek ņemts liels vispārīgs gatavs apmācīts modelis, ko maza komanda vai mazs uzņēmums nav spējīgi apmācīt savām uzdevumam, un tiek pietrenēts, lai tas atbilstu jauna uzdevuma vajadzībām.

Tādā veidā šī darbā tika paņemts un pietrenēts wav2vec2-xlsr, lai to izmantotu runas valodas noteikšanas uzdevumām. Tika apgriezts pēdējais slānis gatavam modelim un nomainīts uz slāni, kas izdod varbūtības valodām. Pirmā wav2vec2-xlsr daļa tika iesaldēta, lai izvairīties no pārmērīgas pielāgošanas.

Implementācijai tika izmantota *Hugging Face Transformers* bibliotēka, kas satur gatavu apmācīts wav2vec2-xslr modeli un to kodu [48]. Pietrenēšanas kods tika rakstīts *Python* valoda izmantot *PyTorch* rīkkopu. Maģistra darba izstrādātas implementācijas ierobežojums ir tāds, ka tā paņem tikai pirmās 3 sekundes no katra audio faila. Tas tika darīts, lai vienkāršotu kodu. Tomēr lielāka daļa no audio failiem, kas ir datu kopā, iekļaujas šajā parametrā.

Pietrenēšanai tika izmantots eksponenciālais apmācības koeficients, kuram sākuma vērtība ir 0.00005 un gamma 0.9. Neironu tīklu svaru optimizācijai tika izmantots Adam optimizācijas algoritms [12]. Pietrenēšanas epohu skaits – 20.

Atbilstoši sadaļā 2.2 aprakstītam eksperimentu plānam, pietrenēšanas eksperimenti tika veikti ar dažādiem apmācības datu apjomiem. Tabulā 2.18. ir noradīts kādas apmācības datu apakškopas tika izmantotas.

2.18. tabula

Apmācības datu kopu apraksts

%	Kopējais audio failu apjoms	Aptuvens apjoms vienai valodai
6%	1h 10m	23m
12%	2h 20m	46m
25%	4h 51m	1h 37m
50%	9h 42m	3h 14m
75%	14h 34m	4h 51m
100%	19h 24m	6h 28m

Tālāk tabulās 2.19. – 2.22. ir attēloti rezultāti eksperimentiem visām trim valodām ar dažādiem apmācības datu apjomiem.

2.19. tabula

Pietrenēšanas rezultāti latviešu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	117	15	88	383	88.64%	23.4%	37.03%	73.47%
6%	463	252	748	37	64.76%	92.60%	76.21%	80.73%
12%	488	82	918	12	85.61%	97.60%	91.21%	93.73%
25%	490	60	940	10	89.09%	98.00%	93.33%	95.33%
50%	495	9	991	5	98.21%	99.00%	98.61%	99.07%
75%	498	14	986	2	97.27%	99.60%	98.42%	98.93%
100%	497	9	991	3	98.22%	99.40%	98.81%	99.20%

Tabulā 2.19. ir rezultāti latviešu valodas noteikšanai ar pietrenēšanas metodi. Var redzēt tendenci, ka jo lielāka ir apmācības datu kopā, jo labāks ir modeļa rezultāts. Līdz ar to vislabākais rezultāts tika sasniegts ar pilnu 100% datu kopas apjomu. Tomēr pārklājums ir nedaudz labāks modelim, kas ir apmācīts uz 75% datu apjoma. Visos gadījumos modeļu rezultāts ir ievērojami labāks nekā Google API rezultāts latviešu valodai.

2.20. tabula

Pietrenēšanas rezultāti krievu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	481	108	892	19	81.66%	96.2%	88.33%	91.53%
6%	314	39	961	186	88.95%	62.80%	73.62%	85.00%
12%	417	3	997	83	99.29%	83.40%	90.65%	94.27%
25%	445	5	995	55	98.89%	89.00%	93.68%	96.00%
50%	490	5	995	10	98.99%	98.00%	98.49%	99.00%
75%	486	1	999	14	99.79%	97.20%	98.48%	99.00%
100%	491	2	998	9	99.59%	98.20%	98.89%	99.27%

Arī krievu valodai labākais rezultāts tika sasniegts ar pilnu 100% datu kopas apjomu (sk. 2.20. tabula). Šeit, tā pat kā latviešu valodas noteikšanas eksperimentā, ir tendence - jo lielāka ir datu kopā, jo labāks ir rezultāts. Tomēr nedaudz labāka precizitāte ir sasniegta izmantojot 75% datu apakškopu. Pietrenēšanas pieejas rezultāts krievu valodai ir labāks par Google API rezultātu jau sākot ar 12% datu apjomu, salīdzinot pēc F1 parametra un akurātuma.

2.21. tabula

Pietrenēšanas rezultāti angļu valodai

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
Google API	480	269	731	20	64.08%	96%	76.86%	80.73%
6%	342	90	910	158	79.17%	68.40%	73.39%	83.47%
12%	490	20	980	10	96.08%	98.00%	97.03%	98.00%
25%	490	10	990	10	98.00%	98.00%	98.00%	98.67%
50%	498	3	997	2	99.40%	99.60%	99.50%	99.67%
75%	499	2	998	1	99.60%	99.80%	99.70%	99.80%
100%	498	3	997	2	99.40%	99.60%	99.50%	99.67%

Atšķirībā no eksperimentiem ar latviešu un krievu valodām, angļu valodai labākais rezultāts tika sasniegts ar modeli, kas ir apmācīts ar 75% datu kopas apjomu (sk. 2.21. tabula). Identisks rezultāts iznāca 50% datu kopai un 100% datu kopai. Kopumā, rezultāts uzlabojas līdz tam brīdim, kad datu kopa sasniedz 75% apjomu. Pietrenēšanas rezultāts angļu valodai ir labāks par Google API rezultātu jau sākot ar 12% datu apjomu, salīdzinot pēc F1 parametra un akurātuma.

2.22. tabula

Pietrenēšanas vidējais rezultāts visām valodām

	Precizitāte	Pārklājums	F1	Akurātums
Google API	78.13%	71.87%	67.41%	81.91%
6%	77.56%	74.6%	74.41%	83.07%
12%	93.66%	93%	92.97%	95.33%
25%	95.33%	95%	95.01%	96.67%
50%	98.87%	98.87%	98.87%	99.24%
75%	98.89%	98.87%	98.87%	99.24%
100%	99.07%	99.07%	99.07%	99.38%

Analizējot visu valodu vidējus rezultātus, var redzēt ka labākais runas valodas noteikšanas rezultāts ir sasniegts ar 100% datu kopu. Tabulā 2.22. var redzēt, ka rezultāts uzlabojas, palielinot datu apjomu. Rezultāts ar 6% datu apakškopu ievērojami atpaliek no lielākām datu kopām. Atšķirībā starp 6% un 12% datu kopām ir daudz lielākā, nekā atšķirībā starp 12% un 25%. Var secināt, ka 6% datu kopa vēl nav pietiekama, lai sasniegtu labu rezultātu. Tomēr salīdzinot ar Google API, pietrenēti modeļi sasniedz labāku F1 un akurātumu jau ar 6% datu kopu. Sākot ar 12% datu kopas apjomu pietrenēti modeli pārspēj Google rezultātu pēc visām metrikām.

2.8 Rezultātu kopsavilkums

Praktiskajā daļā tika veikti eksperimenti ar 4 metodēm: kosinusa līdzība, KNN, MLP un pietrenēšana. Visi iegūti rezultāti tika salīdzināti ar Google Cloud Speech-to-Text API rezultātiem, ņemot tos kā bāzlīniju. Kopsavilkumā tika paņemti visu eksperimentu labākie vidējie rezultāti un apkopoti 2.23. tabulā augošā secībā.

2.23. tabula

Rezultātu kopsavilkums ar visām metodēm

	Precizitāte	Pārklājums	F1	Akurātums
Kosinusa līdzība	48.03%	43.53%	42.13%	62.36%
KNN	69.21%	66.8%	65.1%	77.87%
Google API	78.13%	71.87%	67.41%	81.91%
MLP	84.62%	83.47%	83.56%	88.98%
Pietrenēšana 12%	93.66%	93%	92.97%	95.33%
Pietrenēšana 100%	99.07%	99.07%	99.07%	99.38%

No visām aplūkotam un izmēģinātam metodēm labāko rezultātu sanāca sasniegt ar pietrenēšanas metodi. Jau ar MLP metodi sanāca sasniegt labākos vidējos rezultātus, nekā Google. Turklāt ar pietrenēšanu pietiek ar 12% (2h 20m) datu kopu, lai sasniegt labākos rezultātus nekā visi citi, izmantot šajā darbā.

2.9 Salīdzinājums ar VOXLINGUA107

2021. gadā, kad šīs maģistra darbs jau bija aktīva izstrādes procesā, Tallinas tehnoloģiju universitātes pētnieki prezentēja pētījumu [25], kurā tika pētīts automātiski savāktu tīmekļa audio datu izmantošanu runas valodas noteikšanas uzdevumā. Darba mērķis bija izpētīt, vai no tīkla automātiski savāktus un iezīmētus runas datus var izmantot runas valodas noteikšanas sistēmu veidošanai. Tika izveidotas daļēji nejaušas meklēšanas frāzes no valodai specifiskiem *Wikipēdia* datiem, kas pēc tam tiek izmantoti, lai izgūtu videoklipus no *YouTube* 107 valodās. Iegūtas apmācību datu kopas izmērs ir vairāk nekā 6000 stundas un tā ir publiski pieejama. Šie dati tika izmantoti, lai izveidotu valodas atpazīšanas modeļus vairākiem runas valodas noteikšanas uzdevumiem. Eksperimenti parādīja, ka, izmantojot automātiski iegūtus apmācības datus, tiek iegūti konkurētspējīgi rezultāti, salīdzinot ar pašu manuāli apzīmētu datu kopu izmantošanu. Tiešsaistē ir pieejams runas noteikšanas demonstrācijas rīks, kas ir apmācīts uz šīs datu kopas.

Diemžēl, par šo darbu autore uzzināja pārāk vēli, kad vairs nebija laika veikt modeļu apmācību un pilnvērtīgus eksperimentus. Tomēr, izmantojot demonstrācijas rīku tika veikts provizorisks salīdzinājums ar šī maģistra darba izstrādātu modeli. Tika paņemti 20 audio faili katrai

valodai, jo rīkā var augšupielādēt failus tikai pa vienam un nav iespējas notestēt uzreiz lielu datu kopu. Iegūtie rezultāti atrodas 2.23. tabulā.

2.24. tabula

Nelielas datu kopas testēšanas rezultāti uz demo rīka

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
LV	10	0	40	10	100%	50 %	66.67%	83.33%
RU	17	0	40	3	100%	85%	91.89%	95%
EN	7	0	40	13	100%	35%	51.85%	78.33%
Vidēji					100%	56.67%	70.14%	85.56%
Google API					64.62%	65%	57.57%	79.44%
Pietrenēšana					100%	100%	100%	100%

Tā kā šis rīks tika apmācīts uz daudzām valodām, kā rezultāts tam var būt jebkura valoda no tām, nevis viena no trim valodām, kuras tika izmantots šī maģistra darba ietvaros. Līdz ar to arī precizitāte (sk. 2.24. tabula) sanāca ideāla, jo latviešu, krievu un angļu valodas nav radnieciskas un pastāv maza varbūtība, ka rīks izdos kādu no šim trim valodām kā aplami pozitīvo rezultātu (FP).

Spriežot pēc F1, labākais rezultāts sanāca krievu valodai. Tai ir labākais pārklājums un akurātums. Bet sliktākais rezultāts ir angļu valodai. Rezumējot, VOXLINGUA107 demonstrācijas rīks pārspēj Google, salīdzinot pēc F1 un akurātuma, bet atpaliek no šī maģistra darba izstrādāta runas noteikšanas modeļa.

Pētot VOXLINGUA107 datu kopas tika lejupielādētas 3 valodu apakškopas no – latviešu, krievu un angļu. Diemžēl tika konstatēts, ka lejupielādētās datu kopās nav sadalījuma uz apmācības datiem un testēšanas datiem. Tapāt pētījumā [25] nav minētas tādas metrikas kā F1, pārklājums vai akurātums, kuras tiek izmantotas šī maģistra darbā ietvaros. Līdz ar to, nav iespējams veikt salīdzinājumu.

Tomēr ir iespējams novērtēt, kā šī maģistra darbā izstrādāts modelis darbojas uz cita veida datiem, kas nav *CommonVoice*. No lejupielādētam apakškopām tika nejauši izvēlēti 500 faili katrai valodai un tiek veikta modeļa testēšana. Rezultāti ir apkopoti 2.25. tabulā.

2.25. tabula

Rezultāti uz VOXLINGUA107 datu kopas

	TP	FP	TN	FN	Precizitāte	Pārklājums	F1	Akurātums
LV	379	16	984	121	95.95%	75.80%	84.69%	90.87%
RU	455	41	959	45	91.73%	91.00%	91.37%	94.27%
EN	499	110	890	1	81.94%	99.80%	89.99%	92.60%
Vidēji					89.87%	88.87%	88.68%	92.58%
CommonVoice					99.07%	99.07%	99.07%	99.38%

Rezultāti ar VOXLINGUA107 datu kopu sanāca nedaudz sliktāki, nekā ar *CommonVoice.* Bet tie joprojām ir diezgan precīzi un parāda modeļa potenciālu. Ja apvienot abas datu kopas un izmantot pietrenēšanās metodē, ir iespēja, ka rezultāts būs vēl labāks.

SECINĀJUMI

Darba pirmajā daļā tika pētītas runas valodas noteikšanas metodes. Tika apskatīti pašlaik pieejami rīki un API, un noskaidrots, ka nav gatava laba risinājuma, kas derētu 3 Latvijā runājamām valodām: latviešu, krievu, angļu. Visas trīs valodas atbalsta tikai *Google Cloud Speech,* turklāt serviss veic runas atpazīšanu, kas varētu būt lieka un dārga darbība.

Tika aplūkota literatūra par 3 valodas noteikšanas pieejam: mašīnmācīšanās, dziļa mācīšanās un runas dziļas reprezentācijas. Tika noskaidrots, ka mašīnmācīšanās ir nepieciešamas labas gan lingvistiskas, gan mašīnmācīšanās zināšanas, turklāt precizitāte šai pieejai atpaliek no jaunākām metodēm.

Savukārt, dziļas mācīšanās metodē ir nepieciešamas lielas apmācību datu kopas, lai sasniegtu labu veiktspēju, bet latviešu valodai brīvi pieejamu datu apjoms ir neliels. Gan papildus datu savākšana, gan dziļo neironu tīklu implementēšana nav vienkāršs uzdevums.

Tādā veidā tika secināts, ka vispiemērotākā pieeja turpmākiem eksperimentiem ir daudzvalodu runas dziļas reprezentāciju izmantošana. Šī metode labi darbojas runas atpazīšanas uzdevumos, īpaši zemu resursu gadījumos. Tika izvirzīta hipotēze, ka tas labi darbosies arī runas valodas noteikšanas uzdevumā. Līdz ar to tika nolemts izmantot wav2vec2-xlsr daudzvalodu modeli. Šis modelis jau ir apmācīts un ir brīvi pieejams, kas būtiski vienkāršoja jauna valodas noteikšanas risinājuma izstrādi.

Darba otrajā daļā tika notestēti Google Cloud Speech-To-Text API rezultāti visām trim valodām. Tika konstatēts, ka API nedarbojas pareizi ar latviešu valodu, tāpēc tā vietā tika izmantota radnieciskā lietuviešu valoda.

Tālākos eksperimentos tika pārbaudīta kosinusa līdzības metode un pēc iegūtiem rezultātiem tika secināts, ka kosinusa līdzības metode dod labāko rezultātu nekā vienkārši minēt vai paredzēt vienmēr vienu valodu. Tādā veidā tika veiksmīgi pārbaudīta hipotēze, ka wav2vec2-xlsr runas dziļas reprezentācijas satur informāciju par runas valodu.

KNN metode sasniedza labākus rezultātus, nekā kosinusa līdzība, bet sliktākus, nekā Google. Bet šīs rezultāts tika uzlabots izmantojot daudzslāņu perceptrona modeli, kuram izdevās pārspēt Google rezultātu.

Vislabākais rezultāts tika sasniegts ar pietrenēšanas metodi. Izmantojot 6h apmācības datu kopu katrai valodai, sanāca sasniegt 99.07% precizitāti, 99.07% pārklājumu, 99.07% F1 un 99.38% akurātumu. Eksperimenti tika veikti arī ar mazākiem apmācības datu apjomiem. Sākot ar 12% (46min katrai valodai) datu kopas apjomu pietrenēti modeli sasniedz labus rezultātus un pārspēj Google un visus citus darbā izmēģinātas metodes pēc visām metrikām. Tādā veidā eksperimenti paradīja, ka labus rezultātus var sasniegt jau ar 46 min apmācības datu kopu katrai valodai.

Darba rezultāti tika salīdzināti ar VOXLINGUA107 pētījuma [25] rezultātiem, kas paradījās 2021.gadā, jau pēc literatūras izpētes maģistra darba ietvaros. Šī maģistra darba ietvaros izstrādāts modelis pārspēja VOXLINGUA107 runas valodas noteikšanas demonstrācijas rīku uz nelielas testa datu apakškopas. Autores izstrādāts modelis tika pārbaudīts arī uz 1500 audio failiem no VOXLINGUA107 un sasniedza konkurētspējīgus rezultātus – 89.87% precizitāti, 88.87% pārklājumu, 88.68% F1 un 92.58% akurātumu.

Rezumējot, tika paradīts, ka sākumā ierosināta pieeja sasniedz ļoti labus un konkurētspējīgus rezultātus. Eksperimentiem tika izmantotas nelielas 6 stundas datu kopas katrai valodai, kas ir publiski un brīvi pieejams. Turklāt, tika pieradīts, ka ar šo datu apjomu pietiek, lai iegūtu labus rezultātus, kas arī pārspēj Google Cloud Speech-To-Text API rezultātus. Tā kā izvelētajiem trim valodām netika veikti kādi specifiski modeļu pielāgojumi, var pieņemt, ka šī pieeja tā pat labi darbosies uz jebkurām citām trim valodām, un varbūt arī vairāk nekā trim.

Šī darba ietvaros izstrādāts modelis ir viens no pirmajiem tāda veida risinājumiem latviešu valodai un var tikt pielietots reālās runas apstrādes sistēmās.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. A. Baevski, Y. Zhou, A. Mohamed, M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Neural Information Processing Systems (NeurIPS)*, 2020.
2. A. Conneau et al. "Unsupervised cross-lingual representation learning for speech recognition.", *arXiv preprint arXiv:2006.13979*, 2020.
3. Amazon Transcribe Now Supports Automatic Language Identification. [tiešsaiste]. – [atsauce 14.01.2021]. Pieejams: https://aws.amazon.com/blogs/aws/amazon-transcribe-now-supports-automatic-language-identification/
4. Automatic language detection for speech to text. [tiešsaiste]. – [atsauce 15.01.2021]. Pieejams: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-automatic-language-detection?pivots=programming-language-csharp
5. B. S. Atal, "Automatic recognition of speakers from their voices". *Proceedings of the IEEE*, 64(4), pp.460-475, 1976.
6. BABEL Projekts. [tiešsaiste]. – [atsauce 10.05.2021]. Pieejams: http://www.reading.ac.uk/AcaDepts/ll/speechlab/babel/
7. C. Bartz, T. Herold, H. Yang, C. Meinel, "Language identification using deep convolutional recurrent neural networks". In: *International Conference on Neural Information Processing*. pp. 880–889, 2017.
8. C. Korkut, A. Haznedaroglu, L. Arslan, "Comparison of Deep Learning Methods for Spoken Language Identification". In *International Conference on Speech and Computer* (pp. 223-231), 2020.
9. C. Y. Fook, H. Muthusamy, L. S. Chee, S. B. Yaacob, A. H. B. Adom, "Comparison of speech parameterization techniques for the classification of speech disfluencies". *Turkish Journal of Electrical Engineering & Computer Sciences*, 21(Sup. 1), pp.1983-1994, 2013.
10. Common Voice dataset. [tiešsaiste]. – [atsauce 20.01.2021]. Pieejams: https://commonvoice.mozilla.org/lv/datasets

11. D. Martinez, O. Plchot, L. Burget, O. Glembek, P. Matejka, “Language recognition in ivectors space”. In: *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

12. D. P. Kingma, J. Ba, “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980*, 2014.

13. D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, S. Khudanpur. “Spoken language recognition using x-vectors.”, *In Odyssey*, The Speaker and Language Recognition Workshop, 2018, pp. 105-111.

14. D. Snyder, D. Garcia-Romero, G. Sell, D. Povey and S. Khudanpur, “X-Vectors: Robust DNN Embeddings for Speaker Recognition” *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329-5333.

15. Detecting language spoken automatically. [tiešsaiste]. – [atsauce 14.01.2021]. Pieejams: https://cloud.google.com/speech-to-text/docs/multiple-languages

16. F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, D. A. Reynolds, “A tutorial on text-independent speaker verification”. *EURASIP journal on applied signal processing*, 2004(4), pp.1-22.

17. G. Montavon, “Deep learning for spoken language identification”. In: *NIPS Workshop on deep learning for speech recognition and related applications*. pp. 1–4, 2009.

18. H. Hermansky, “Perceptual linear predictive (plp) analysis of speech”. *the Journal of the Acoustical Society of America*, 87(4):1738-1752, 1990.

19. H. Soltau, H.K. Kuo, L. Mangu, G. Saon, and T. Beran, “Neural Network Acoustic Models for the DARPA RATS Program” in *Interspeech*, 2013, pp. 3092 - 3096.

20. H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, “mixup: Beyond empirical risk minimization”, in *arXiv preprint arXiv:1710.09412*, 2017.

21. Huggin Face Transformers. [tiešsaiste]. – [atsauce 14.05.2021]. Pieejams: https://huggingface.co/

22. I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, “Automatic Language Identification using Deep Neural Networks” In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5337-5341.2014

23. J. Devlin, M.W. Chang, K. Lee, K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”. In *arXiv preprint arXiv:1810.04805*, 2018.

24. J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno, “Automatic language identification using long short-term memory recurrent neural networks,” in *INTERSPEECH 2014*, pp. 2155–2159, 2014.

25. J. Valk, and T. Alumäe. “Voxlingua107: a dataset for spoken language recognition.”, in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 652-658.

26. K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, *CVPR* 2016, pp. 770–778.

27. L. Ferrer, N. Scheffer, E. Shriberg, “A comparison of approaches for modeling prosodic features in speaker recognition” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 4414–4417.

28. LibriSpeech ASR corpus. [tiešsaiste]. – [atsauce 20.01.2021]. Pieejams: http://www.openslr.org/12

29. M. A. Zissman, “Comparison of Four Approach to Automatic Language Identification of Telephone Speech”, *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31-44, 1996.

30. Multilayer perceptron. Wikipedia. [tiešsaiste]. – [atsauce 19.05.2021]. Pieejams: https://en.wikipedia.org/wiki/Multilayer_perceptron

31. N. Dehak, P. Kenny, R. Dehak, P. Ouellet, P. Dumouchel, “Front End Factor Analysis for Speaker Verification,” *IEEE Transactions on Audio, Speech and Language Processing*, 2011.

32. N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, P. Dumouchel, “Support Vector Machines versus Fast Scoring in the Low Dimensional Total Variability Space for Speaker Verification,” *in Interspeech, Brighton*, 2009.

33. N.E. Safitri, A. Zahra, M. Adriani, “Spoken language identification with phonotactics methods on minangkabau, sundanese, and javanese languages”. *Procedia Computer Science* 81, pp. 182–187, 2016.

34. O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition,” in *ICASSP. IEEE*, 2012, pp. 4277– 4280.

35. P. A. Torres-Carrasquillo, D. A. Reynolds, J. R. Deller Jr., “Language identification using Gaussian Mixture Model Tokenization”. In *ICASSP, Orlando, Fl., USA*, 2002.

36. P. Matejka, P. Schwarz, J. Cernocky and P. Chytil, "Phonotactic Language Identification using High Quality Phoneme Recognition," Eurospeech 2005, September 2005.

37. Perceptron. Wikipedia. [tiešsaiste]. – [atsauce 19.05.2021]. Pieejams: https://en.wikipedia.org/wiki/Perceptron

38. R. Tong, B. Ma, D. Zhu, H. Li, E.S. Chng, “Integrating acoustic, prosodic and phonotactic features for spoken language identification”. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. vol. 1, pp. I–I, 2006.

39. Russian Open Speech to Text (STT/ASR) Dataset. [tiešsaiste]. – [atsauce 20.01.2021]. Pieejams: https://github.com/snakers4/open_stt

40. S. Davis, P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357-366, 1980.

41. S. Revay, M. Teschke, “Multiclass Language Identification using Deep Learning on Spectral Images of Audio Signals”. *arXiv preprint arXiv:1905.04348,* 2019.

42. S. Schneider, A. Baevski, R. Collobert, M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” in *Proc. ISCA Interspeech, 2019*, pp. 3465–3469, 2019.

43. Shorttext homepage. [tiešsaiste]. – [atsauce 14.05.2021]. Pieejams: https://shorttext.readthedocs.io/en/latest/

44. Spoken Language Identification Demo. [tiešsaiste]. – [atsauce 13.05.2021]. Pieejams: https://bark.phon.ioc.ee/lid_demo

45. TranslatedLabs - Spoken Language Identifier. [tiešsaiste]. – [atsauce 14.01.2021]. Pieejams: https://translatedlabs.com/spoken-language-identifier

46. V. Gazeau, C. Varol, "Automatic spoken language recognition with neural networks". *Int. J. Inf. Technol. Comput. Sci. (IJITCS)* 10(8), 11–17, 2018.

47. voxforge.org: Free speech recognition (Linux, Windows and Mac) - voxforge.org. [tiešsaiste]. - [atsauce 23.01.2021.]. Pieejams: http://www.voxforge.org/

48. XLSR-Wav2Vec Transformers. [tiešsaiste]. – [atsauce 18.05.2021]. Pieejams: https://huggingface.co/transformers/model_doc/xlsr_wav2vec2.html

49. Z. Fan, M Li, S. Zhou, B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification". *arXiv preprint arXiv:2012.06185*, 2020.

PIELIKUMI

1. Pielikums. Instrukcija modeļa izmēģināšanai

Darba ietvaros vislabākais rezultāts tika sasniegts ar pietrenēšanas metodi. Kods wav2vec2-xlsr pietrenēšanai runas valodas noteikšanai, kā arī kods pietrenēta modeļa darbināšanai tika izvietots *GitHub:*

https://github.com/veronikahromchenkova/spokenLanguageID

Maģistra darba ietvaros labāko pietrenēto modeli var lejupielādēt šeit:

https://drive.google.com/file/d/1f_1ifkPgMMl4d2Jd4X-hn1v71QwHZ94h/view?usp=sharing

Instrukcija modeļa izmēģināšanai:

1. Lejupielādējiet kodu no iepriekšminētā *GitHub* repozitorija.
2. Lejupielādējiet pietrenēto modeli un ievietojiet to kopā ar Python kodu /FineTuning mapē.
3. Ieinstalējiet atkarības no requirements.txt: `pip install -r requirements.txt`
4. Sagatavojiet mapi ar testa piemēriem wav formātā.
5. Izpildiet komandu `python score.py PATH_TO_TEST_FILES_FOLDER`

Maģistra darbs "**Runas valodas noteikšana**" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 23.05.2021.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autore: Veronika Hromčenkova, 23.05.2021.
(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / ~~n e p i e m ē r o t u~~** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: Dr. dat. Askars Salimbajevs
(Vadītāja paraksts un datums)

Darbs iesniegts **maģistratūras sekretariātā** _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: Dr.sc.comp. Normunds Grūzītis
(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)