ELEKTROTEHNIŠKI VESTNIK 92(5): 261–269, 2025
IZVIRNI ZNANSTVENI ČLANEK

Sistem za podnaslavljanje slovenskega govora za gluhe in naglušne

Blaž Kovačič, Janez Brest, Borko Bošković

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Koroška cesta 46, 2000 Maribor, Slovenija

E-pošta: blaz.kovacic@student.um.si

Povzetek. Razvili smo rešitev za gluhe in naglušne osebe v obliki aplikacije, imenovane UHO. Rešitev s pomočjo lokalnega modela razpoznave govora omogoča realnočasovni prikaz podnapisov slovenskega govora, ki se predvaja na napravi. V ta namen smo s pomočjo govornega korpusa Artur 1.0 učili modele za razpoznavo govora. Učenje je potekalo na superračunalniku VEGA, kjer smo na testni množici za večji model *base* dosegli stopnjo napačno razpoznanih besed 11,38 % in 15,19 % za manjši model *tiny*. Realnočasovno izvedbo smo zagotovili z uporabo ustreznih zaledij za sklepanje z modeli in s pristopom optimizacije dekodiranja žetonov, ki ga imenujemo aditivno dekodiranje. Rezultati so spodbudni in kažejo, da realnočasovna razpoznava daje primerljive rezultate kot razpoznava celotnih posnetkov naenkrat.

Ključne besede: Realnočasovna razpoznava govora, okvara sluha, samodejno podnaslavljanje, strojno učenje

Slovenian speech captioning system for deaf and hard of hearing

We developed a solution for deaf and hard of hearing in the form of an application, called UHO. The solution uses a local speech recognition model to display real-time captions of Slovenian speech that is being played on the device. For this purpose, we used the speech corpus Artur 1.0 to train the speech recognition models. We trained the models on the VEGA supercomputer, where we achieved word error rates of 11.38% for the larger *base* model and 15.19% for the smaller *tiny* model. We ensured real-time execution by using appropriate model inference backends and by using a token decoding optimization approach that we call *additive decoding*. Results are promising, indicating that real-time speech recognition can give comparable results to recognition of full-length audio samples.

1 UVOD

Okvara sluha spada med eno izmed najtežjih oblik invalidnosti [2]. V Sloveniji o večjih težavah s sluhom poroča 5,7 % populacije [3]. Prav tako več kot 5 % svetovne populacije zaradi okvare sluha potrebuje rehabilitacijo. Do okvare sluha lahko pride na več načinov. Med te sodijo npr. težave v obdobju pred ali med rojstvom, kronične okužbe ušes, poslabšanje sluha zaradi starosti, izpostavljenost glasnim zvokom in podobno. Gluhim in naglušnim lahko vsakdanje situacije predstavljajo velik stres, gluhost pa vpliva na kakovost življenja in vodi do socialne izolacije. Poleg tega lahko gluhost negativno vpliva na socialno varnost, delozmožnost, proces izobraževanja, zmožnost komunikacije in počutje. Pri starejših z okvaro sluha je prav tako lahko prisotna depresija [4].

Gluhe osebe v Sloveniji imajo sicer pravico do tolmača, vendar žal le v omejenem obsegu ur (t. i. vavčerjev). Na voljo imajo 30 ur letno, razen dijakov in študentov, ki jim pripada 100 ur letno [5]. Gluhe osebe poudarjajo, da je to premalo [6].

Tehnološki napredek na področju umetne inteligence je omogočil razvoj različnih aplikacij za osebe s posebnimi potrebami, kot so [36][37][38]. Razcvet je doživela tudi tehnologija razpoznave govora, ki danes že dosega stopnjo razpoznave, primerljivo s človeško sposobnostjo [12]. Preprostejši sistemi razpoznavajo po eno besedo naenkrat (angl. isolated-word speech recognition). To je uporabno npr. pri zaznavi začetne besede pri razpoznavi govora (angl. wake word detection) in v sistemih z glasovnimi ukazi [39]. Po drugi strani kompleksnejši sistemi govor razpoznavajo kontinuirano (angl. continuous speech recognition). V našem delu gre za kontinuirano razpoznavo govora.

Razpoznava govora na področju podporne tehnologije (angl. Assistive Technology) najde uporabno vrednost pri različnih podpornih sistemih. Primer tega so programske rešitve podjetja Nuance [16], ki osebam s stanji, kot je kvadriplegija, omogočajo glasovni nadzor računalnika. Obstajajo tudi rešitve, ki gluhim uporabnikom omogočajo prikaz strojno generiranih podnapisov za zvok, ki se predvaja na napravi. Primer tega je aplikacija "Live captions" [7] podjetja Microsoft, ki je na voljo v okolju operacijskega sistema Windows. Prav tako za naprave z operacijskim sistemom Android obstaja aplikacija "Samodejni podnapisi" [8] podjetja Google. Žal nobena od omenjenih aplikacij ne podpira slovenskega jezika.

Prejet 31. junij, 2025
Odobren 21. oktober, 2025

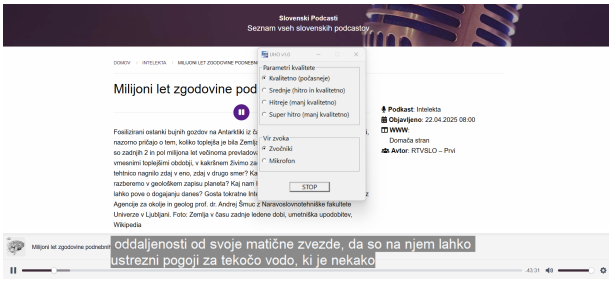
Pogosto so rešitve za razpoznavo govora storitve v oblaku, npr. Google Cloud Speech-to-Text [1]. Razlog za implementacijo razpoznavalnikov govora v oblaku je predvsem računska zahtevnost izvajanj velikih modelov strojnega učenja. Slabosti pa so, da so takšne storitve plačljive, da porajajo vprašanje o zasebnosti, prav tako pa lahko kakovost dobljenih transkriptov variira.

V našem delu smo razvili rešitev* v obliki aplikacije za operacijska sistema Windows in Android, imenovane UHO. Rešitev omogoča realnočasovno podnaslavljanje slovenskega govora in temelji na uporabi lokalnih modelov Whisper [12] podjetja OpenAI brez potrebe po internetni povezavi. Modele smo na superračunalniku VEGA doučili (angl. fine-tuning) s pomočjo slovenskega govornega korpusa Artur 1.0 [9]. V okolju operacijskega sistema Windows smo uporabili večji model Whisper *base*, v okolju operacijskega sistema Android pa najmanjši model Whisper *tiny*. Primer grafičnega izrisa podnapisov ob predvajanju določenega podkasta prikazuje slika 1. Podnapisi se sproti izpisujejo v sivem polju na dnu zaslona, ne glede na to, katero okno je v ospredju.

V aplikaciji UHO za operacijski sistem Windows smo implementirali sklepanje (angl. inference) na centralni procesni enoti (CPE), da bi aplikacija lahko delovala na čim širšem naboru računalnikov. V aplikaciji za operacijski sistem Android smo pri sklepanju omogočili Androidov aplikacijski programski vmesnik za nevronske mreže (Android Neural Networks API – NNAPI), ki sam določi, katere operacije bodo pri sklepanju potekale na CPE in katere na grafični procesni enoti (GPE).

Modeli Whisper na vhodu zahtevajo 30-sekundni posnetek oz. pripadajoč log-mel spektrogram. To je predstavitev zvočnega signala v frekvenčnem prostoru, ki s pomočjo logaritmično razporejenega sklopa filtrov daje večji poudarek signalu pri višjih frekvencah. Če torej v scenariju realnočasovne razpoznave z modelom sklepamo približno vsaki 2 sekundi, še zmeraj obstaja 28-sekundno prekrivanje govora med trenutno in prejšnjo iteracijo sklepanja. To pri dekodiranju žetonov prinaša veliko redundanco, saj je večina žetonov med iteracijama enaka. Za namen zmanjšanja redundance smo implementirali pristop, kjer smo v t. i. poziv (angl. prompt) modela podali dekodirane žetone iz prejšnje iteracije sklepanja. Tako smo izračunali le na novo pridobljene žetone in pri dekodiranju prihranili veliko procesorskega časa. Ta intuitivni pristop imenujemo *aditivno dekodiranje* in ga podrobneje opišemo v nadaljevanju.

V rešitvi UHO za operacijski sistem Windows smo za sklepanje uporabili knjižnico CTranslate2 [21]. CTranslate2 omogoča učinkovito sklepanje z modeli transformerjev in je na voljo za programska jezika C++ in Python. Zaradi potrebe po večji fleksibilnosti pri postopku aditivnega dekodiranja smo skozi analizo knjižnice namesto višjenivojskih rutin uporabili določene nižjenivojske rutine knjižnice. Po drugi strani smo v rešitvi za operacijski sistem Android sklepali s pomočjo knjižnice *ONNX* (angl. Open Neural Network Exchange) [25]. Modele Whisper smo doučili s Pythonovo knjižnico *HuggingFace Transformers* [26], ki pa uporablja modele tipa pytorch [27], zato je bilo treba modele pretvoriti iz tipa pytorch v tip ONNX. Pri tem smo dodatno implementirali pomembno tehniko za pohitritev sklepanja z dekodirnikom, imenovano KV-predpomnjenje (angl. Key-Value Caching – KV Caching – predpomnjenje ključev in vrednosti).



Slika 1 Demonstracija podnaslavljanja naše aplikacije UHO za operacijski sistem Windows, na primeru zvočne vsebine določenega podkasta.

Pri vrednotenju rezultatov smo uporabili znano metriko stopnje napačno razpoznanih besed (angl. Word Error Rate – WER), ki je definirana [14] z enačbo (1). V enačbi I (angl. Insertions), D (angl. Deletions) in S (angl. Substitutions) predstavljajo število vstavljenih, izbrisanih in zamenjanih besed v razpoznanem besedilu. N predstavlja število besed v referenčnem (dejansko izgovorjenem) besedilu.

$$\mathrm{WER} = \frac{I + D + S}{N} \times 100\ [\%] \tag{1}$$

Človek razpoznava z vrednostjo WER okoli $4\,\%$ [23], medtem ko bi bila vrednost WER pri razpoznavi govora v idealnem primeru $0\,\%$. Pregled vrednosti WER, ki jih na različnih govornih korpusih dosegajo različni modeli strojnega učenja, je podan v [30].

Struktura članka je sledeča. V poglavju 2 članka opišemo sorodna dela. Nato v poglavju 3 opišemo, kako uporabljamo modele Whisper, kjer med drugim podrobneje opišemo naš pristop aditivnega dekodiranja. Sledi poglavje 4, kjer opišemo način učenja modelov. Rezultate učenja predstavimo v poglavju 5. V poglavju 6 sledi predstavitev rezultatov realnočasovne razpoznave, kot so meritve časov sklepanja z modeli, točnost realnočasovne razpoznave in primerjava rezultatov s sorodnimi deli. V zadnjem poglavju 7 povzamemo ugotovitve in prispevke našega dela ter opišemo smernice za nadaljnje delo.

*Programska koda rešitve je na voljo na https://github.com/obstino-org/uho, naučeni modeli in učni podatki pa na https://huggingface.co/blko.

2 SORODNA DELA

V delu [18] so razpoznavali slovenski govor z nevronskimi mrežami s časovnim zamikom (angl. Time Delay Neural Network – TDNN) in mrežami z dolgim kratkoročnim spominom (angl. Long Short-Term Memory). Uporabili so tri različne govorne korpuse, med drugim tudi Gos 1.0 [32]. Avtorji so primerjali metode, temelječe na nevronskih mrežah s statistično osnovanim pristopom skritih modelov Markova v kombinaciji z mešanicami Gaussovih porazdelitev (angl. Hidden Markov Models, Gaussian Mixture Models). Najboljše rezultate na testni množici so dosegli s TDNN, in sicer z vrednostjo WER $27{,}16\,\%$. V nasprotju z našim delom so avtorji uporabili prilagojene arhitekture nevronskih mrež, medtem ko v našem delu Whisper temelji na arhitekturi transformer. Prav tako so pri učenju uporabili relativno manjše učne množice v skupnem obsegu približno 140 ur.

Zanimivo je delo [17], v katerem so avtorji razpoznavali slovenski govor za domeno dnevnoinformativnih oddaj. Avtorji so za razpoznavo govora uporabili nevronsko mrežo in ločen trigramski jezikovni model z velikostjo slovarja 250.000 besed. Nevronsko mrežo so učili z manjšo količino zvočnih posnetkov v obsegu 66 ur in dosegli vrednost WER $15{,}71\,\%$. Avtorji so v nasprotju z našim delom uporabljali ločen akustični in jezikovni model. Rešitev avtorjev je dosegla relativno nizko vrednost WER. Obstaja pa verjetnost, da njihov model ne bi deloval tako uspešno na posnetkih zunaj domene dnevnoinformativnih oddaj.

Obstaja tudi delo Online Notes [33], v katerem so za študente iz tujine in študente s posebnimi potrebami razvili sistem za realnočasovno prevajanje in podnaslavljanje govora. Govor so razpoznavali in prevajali v oblaku. Avtorji poročajo o vrednosti WER $20\,\%$, ki so jo dosegli ob prvem predavanju v sklopu semestrskega testiranja. V delu Online Notes v nasprotju z našo rešitvijo govora niso razpoznavali lokalno na napravi, temveč v oblaku. Prav tako so morali svoj jezikovni model sproti prilagajati vsebini predavanj.

Z realnočasovno razpoznavo z modeli Whisper so se ukvarjali v delu Whisper-Streaming [19]. Avtorji so, podobno kot mi, uporabljali pristop, kjer so doslej dekodirane žetone dali modelu na vhodu kot poziv. Implementacija njihove rešitve periodično zajema nov zvok govora. V zadnjih dveh iteracijah zajema zvoka in razpoznave govora obstaja prekrivanje žetonov, ki so bili dekodirani. Prekrivajoči se žetoni se štejejo kot potrjen transkript. To dosežejo s pomočjo t. i. algoritma lokalnega soglasja (angl. Local Agreement) [31]. S pomočjo informacije o časovnih oznakah ob zaznavi ločila nato krajšajo zvočni medpomnilnik do časovne oznake ločila. S tem zagotovijo, da medpomnilnik vselej vsebuje zgolj eno poved. Rešitev Whisper-Streaming v nasprotju z našo za uspešno delovanje potrebuje model, ki vsebuje časovne oznake na nivoju besed in napoveduje ločila. To lahko pomeni oviro pri implementaciji rešitve, saj korpusi, kot je Artur 1.0, ne vsebujejo časovnih oznak na nivoju besed. Prav tako so avtorji uporabljali večje modele in jih izvajali na GPE, medtem ko je naša rešitev zasnovana na uporabi manjših modelov in sklepanja na CPE.

3 MODELI WHISPER

Modele Whisper je podjetje OpenAI predstavilo in odprtokodno dalo na razpolago leta 2022 [12]. Modeli temeljijo na arhitekturi transformerjev in med drugim omogočajo večjezično razpoznavo govora. Za angleški jezik model *whisper-base* dosega vrednost WER $5{,}0\,\%$ na angleškem korpusu LibriSpeech.test-clean, medtem ko vrednost WER na slovenskem korpusu CommonVoice 9 znaša $70{,}3\,\%$. Ker je slednja vrednost za praktično uporabo povsem nesprejemljiva, smo pri našem delu uporabili pristop doučenja modelov. Pri tem smo skripto za učenje osnovali na [15]. Modeli Whisper so na voljo v različnih velikostih [22]. Poznamo *tiny* (najmanjši), *base*, *small*, *medium* in *large* (največji)*.

3.1 Sklepanje z modeli

Modeli Whisper so sestavljeni iz kodirnih in dekodirnih blokov. Z modeli Whisper značilno sklepamo tako, da na vhodu kodirnika podamo log-mel spektrogram 30-sekundnega posnetka. S kodirnikom za dan posnetek sklepamo zgolj enkrat, nato pa z dekodirnikom sklepamo večkrat – enkrat za vsak nov izhodni žeton. Izhodni žetoni so del slovarja Whisper, ki vsebuje 51.865 podbesednih enot (delov besed). Z dekodirnikom sklepamo tako, da mu na vhodu podamo izhod kodirnika in do zdaj dekodirane žetone – t. i. poziv. Poziv daje dekodirniku kontekst za izračun naslednjega izhodnega žetona. Na začetku dekodiranja je v našem primeru poziv sestavljen iz zaporedja žetonov "`<|startoftranscript|><|sl|><|transcribe|>`" [34], ki jim pripadajo numerični indeksi `50258`, `50305`, `50359`†. Tukaj "`<|sl|>`" pomeni oznako, ki pove, da gre za razpoznavo slovenskega jezika.

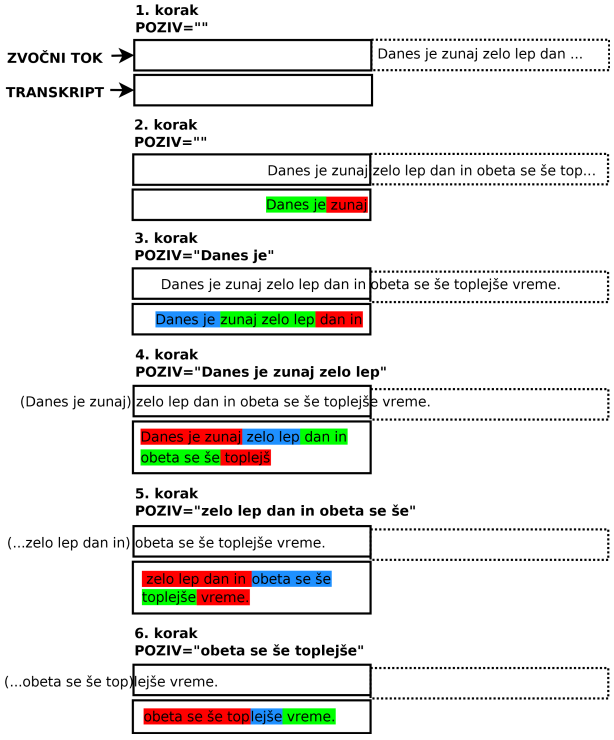
Tipično z namenom pridobivanja točnejših transkripcij uporabljamo pri dekodiranju tudi metodo iskanja z žarki (angl. beam search) [24]. To je pristop, kjer vzporedno preiskujemo več možnih izhodnih zaporedij žetonov. Število preiskovalnih poti je znano tudi kot *število žarkov* (angl. beam size). Na koncu izberemo zaporedje z največjo verjetnostjo žetonov.

3.2 Predpomnjenje ključev in vrednosti

Pri sklepanju na operacijskem sistemu Android smo model ONNX prilagodili tako, da omogoča uporabo

*Na voljo so *large*, *large-v2*, *large-v3* in *large-v3-turbo*, kjer je slednji optimiziran za hitrost sklepanja.

†Indekse žetonov najlažje dobimo s pomočjo leksikalnega analizatorja knjižnice *whisper* [35].



Slika 2 Prikaz postopka aditivnega dekodiranja. Zeleno: na novo dekodirani žetoni; modro: starejši potrjeni žetoni; rdeče: zavrženi žetoni.

metode t. i. predpomnjenja ključev in vrednosti. Ključi in vrednosti so vektorji, ki se računajo v mehanizmu pozornosti arhitekture transformer. Ker tudi modeli Whisper temeljijo na transformerjih, smo s to tehniko dosegli bistveno pohitritev časa sklepanja z dekodirnikom. Prilagojeni model dekodirnika tako zdaj na vhodu dodatno sprejema ključe in vrednosti iz prejšnje iteracije sklepanja. Dekodirnik nato vrne modificiran nabor ključev in vrednosti, ki jih v naslednji iteraciji sklepanja dekodirniku ponovno damo na vhod. Pohitritev dosežemo, saj nam s tem pristopom pri sklepanju ni treba ponovno izračunavati vseh ključev in vrednosti iz prejšnjih iteracij.

3.3 Aditivno dekodiranje

Pri razpoznavi govora značilno zajemamo zvok vsaki 2 sekundi in ga dodajamo v 30-sekundni medpomnilnik. Naivna implementacija realnočasovne razpoznave govora bi zaradi zahteve modelov Whisper po fiksni 30-sekundni dolžini vhodnih posnetkov vsakič znova izvajala razpoznavo govora nad celotnimi posnetki.

Mi smo se odločili uporabiti drugačen, intuitiven pristop, ki ga imenujemo *aditivno dekodiranje*. Ta pri sklepanju z dekodirnikom omogoča uporabo oz. vnos informacije žetonov iz prejšnje iteracije sklepanja v trenutno iteracijo. Efektivno to pomeni, da dekodiramo zgolj nove žetone iz trenutne iteracije, in tako posledično znatno zmanjšamo število žetonov, ki jih moramo dekodirati.

Pristop je ilustriran na sliki 2. Za vsak korak je s pravkotnikom s polno črto in označbo “zvočni tok” prikazan 30-sekundni medpomnilnik. Ko zajamemo 2 sekundi novega zvočnega odseka, ga dodamo na konec medpomnilnika, hkrati pa z začetka medpomnilnika odstranimo 2 sekundi zvoka. Zvok, ki še prihaja, je označen s pravokotnikom s črtkano črto. Zeleno označeni so žetoni, ki smo jih dekodirali v trenutnem koraku. Rdeče označeni so žetoni na začetku in koncu, ki smo jih zavrgli. Žetone na začetku zavržemo zato, ker nam je model zanje vrnil nizke verjetnosti in sklepamo, da v medpomnilniku ni več pripadajočega zvoka. Žetone na koncu pa zavržemo, ker je stavek na koncu zvočnega toka pogosto odrezan sredi zadnje besede in posledično napačno razpoznan. Imamo tudi modro označene žetone, to so žetoni, ki smo jih dekodirali v prejšnjih iteracijah. Poziv v naslednjem koraku je sestavljen iz spajanja modrih in zelenih žetonov prejšnje iteracije.

Ko ob zajemu novega zvoka krajšamo začetek zvočnega medpomnilnika, začetek posnetka morda ne bo več vseboval vsebine govora iz prejšnjih iteracij sklepanja. Kot omenjeno, nam bo dekodirnik v naslednji iteraciji sklepanja za žetone odstranjenega zvoka vrnil zelo nizko verjetnost. Te žetone odstranjujemo na nekoliko podoben način kot avtorji v [12] zaznavajo t. i. halucinacije (angl. hallucinations) [13], s pomočjo izračuna verjetnosti žetonov. Natančneje, pri našem pristopu odstranjujemo žeton na prvem indeksu, dokler je povprečje logaritmov verjetnosti prvih 5 žetonov manjše od določenega praga.

Podobno s pomočjo izračuna logaritma verjetnosti zadnjih 5 žetonov tudi odstranjujemo rdeče označene žetone s konca transkripcij, kar lahko vidimo na sliki 2. Dodatno s konca odstranjujemo žetone, dokler ti vsebujejo neabecedne simbole. To je koristno, saj model zaradi prekinjenega govora kot zadnji simbol pogosto napove ločilo, čeprav se stavek še nadaljuje. Poleg tega s konca odstranimo še $backPopExtra = 3$ žetone, saj je zadnji stavek razpoznanega govora zaradi odsekovne narave zajema zvoka pogosto odrezan sredi besede. Posledično to pomeni, da bi dobili napačne transkripcije, če teh žetonov ne bi odstranili.

3.4 Praktični izzivi in rešitve

Modeli Whisper so dovzetni predvsem za pojav t. i. halucinacij [12], kjer model napoveduje besede, ki jih ni v zvočnem posnetku. Pogosto se to zgodi, ko je na posnetku tišina, lahko pa se tudi sicer. Halucinacije se lahko pojavijo kot naključne besede ali kot neprekinjen niz ponavljajočih se besed (angl. repeat loops [12]) kot npr. “*da je bilo je bilo je bilo...*”. Halucinacije smo v našem delu zaznavali s pomočjo preverjanja verjetnosti žetonov, saj imajo pogosto nižjo verjetnost. Dodatno smo jih zaznavali z merjenjem kompresijskega razmerja izhodnega transkripta, kajti ponavljajoče se besede v halucinacijah imajo višje kompresijsko razmerje. Pri tem smo se zgledovali po pristopu avtorjev [12]. Kadar halucinacije zaznamo, nastavimo žetone poziva na prazen seznam in 30-sekundni zvočni medpomnilnik na ničelne

vrednosti. S tem zagotovimo nadaljevanje dekodiranja, žal pa posledično določen del govora ne bo razpoznan.

Dodaten izziv je tudi situacija, pri kateri pride do zastoja pri dekodiranju in model namesto napovedi novega žetona napove žeton za konec razpoznanega besedila (<|endoftext|>). Opažali smo, da do tega pogosto pride pri menjavi govorca znotraj zajetega zvoka. Videli smo, da je ta neželeni pojav mogoče zmanjšati z odstranjevanjem odsekov daljših tišin z detektorjem glasovne aktivnosti (angl. Voice Activity Detector – VAD)[‡]. V ta namen smo v našem delu uporabili model Silero VAD [20]. Prav tako smo s pomočjo VAD detektirali, kdaj je v posnetku 3-sekundna tišina, ob pojavitvi katere smo resetirali poziv in zvočni medpomnilnik.

4 UČENJE

4.1 Opis korpusa Artur 1.0

Korpus Artur 1.0 [9] vsebuje 1.067 ur slovenskega govora, od katerih je 884 ur transkribiranih. Korpus med drugim vsebuje posnetke branega govora iz korpusa Gigafida 2.0, posnetke parlamentarnega govora ter posnetke javnega in nejavnega govora. Mi smo pri učenju od tega uporabili 832 ur – pri tem nismo vključili studio posnetkov enega govorca.

Korpus za vse posnetke govora vsebuje t. i. standardizirane in pogovorne zapise transkriptov. Primer pogovornega zapisa transkripta iz korpusa je "Res je, da je predlog zakona *naraunan* za tisto kategorijo, ki je *najbol* ogrožena." Pripadajoči transkript v standardiziranem zapisu je "Res je, da je predlog zakona naravnan za tisto kategorijo, ki je najbolj ogrožena." V našem delu smo se pri učenju odločili za uporabo standardiziranih transkripcij.

Korpus sicer nima časovnih oznak na nivoju besed, kjer bi za vsako besedo vedeli, kdaj se je pojavila v posnetku, vsebuje pa časovne oznake na nivoju skupkov besed oz. stavkov. Pri učenju z do 30 sekund dolgimi posnetki smo se odločili za dodajanje žetona s časovno oznako z dolžino trajanja posnetka na konec zaporedja učnih žetonov. Če je bilo trajanje posnetka na primer $17{,}4$ sekunde, smo pripeli žeton z oznako <|17.40|>. To je mogoče, saj imajo modeli Whisper rezervirane žetone za časovne oznake, smiselno pa se nam je zdelo, saj obstaja argument, da vključitev časovnih oznak lahko preprečuje pojav halucinacij. Transkripcije so v formatu .trs orodja za transkribiranje Transcriber 1.5.1 [10]. Primer besedila v datoteki .trs prikazuje transkript 1.

Transkript 1: Zgled odseka datoteke tipa .trs

```
<Sync time="61.768"/>
Res je, da
<Sync time="63.196"/>
je predlog zakona
<Sync time="64.791"/>
naravnan za tisto
```

Transkripcije vsebujejo* tudi okoli 200 neverbalnih in polverbalnih glasov, ki so v besedilu označeni z lojtro (npr. glas *#mhm*). Mi smo pri učenju modelov te oznake v učnih podatkih ohranili.

4.2 Priprava podatkov in okolja za učenje

Ker modeli Whisper delujejo nad 30-sekundnimi posnetki, smo morali pri pripravi učnih podatkov daljše posnetke primerno skrajšati. To je bilo mogoče storiti, saj korpus Artur vsebuje informacijo o časovnih oznakah. Primer časovne oznake v transkriptu 1 je denimo "<Sync time=61.768/>", kar pomeni, da se odsek besedila za to oznako začne pri $61{,}768$ sekundi danega posnetka. Da smo daljše posnetke skrajšali, smo torej s pomočjo opisanih oznak krajše odseke združevali, dokler je bilo njihovo skupno trajanje krajše od 30 sekund. Tako smo razrezali daljše posnetke na krajše, največ 30-sekundne odseke.

Učili smo s pomočjo skripte Python, predvsem z uporabo knjižnic *HuggingFace Transformers* [26], *HuggingFace Datasets* [28], *pytorch* [27] in *evaluate* [29]. Celotni korpus na disku zavzema okoli 300 GB prostora in je pomnilniško zahteven tudi po ekstrakciji značilnic log-mel spektrogramov. To je pomenilo izziv, saj se med učenjem učni podatki navadno hranijo v glavnem pomnilniku, zato smo se odločili za uporabo t. i. pretakanja (angl. dataset streaming) s pomočjo knjižnice *HuggingFace Datasets*. S pomočjo naše skripte smo ustvarili več komprimiranih datotek tipa ".parquet" [11], ki smo jih potem med učenjem pretočno brali z diska.

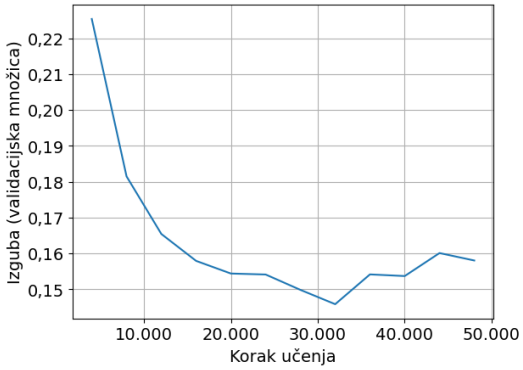
Učili smo na superračunalniku VEGA, na vozliščih z NVIDIA A100 GPE. Skripto Python smo zagnali znotraj vsebnika Singularity. Skripto smo snovali tako, da je bilo omogočeno nadaljevanje učenja tudi po njegovi prekinitvi, do katere pride zaradi časovne omejitve izvajanja procesa. Skupaj je učenje trajalo približno 3 dni. Model *base* smo učili s stopnjo učenja $2{,}50\mathrm{E}{-5}$ in velikostjo paketa 32, model *tiny* pa s stopnjo učenja $3{,}75\mathrm{E}{-5}$ in velikostjo paketa 32. Skupaj smo učili 51.000 korakov oziroma 6 epoh. Število epoh izračunamo s pomočjo enačbe 2.

$$\text{štEpoh} = \frac{\text{štKorakov} \times \text{velikostPaketa}}{\text{štUčnihPrimerkov}} = \frac{51.000 \times 32}{270.000} \approx 6 \tag{2}$$

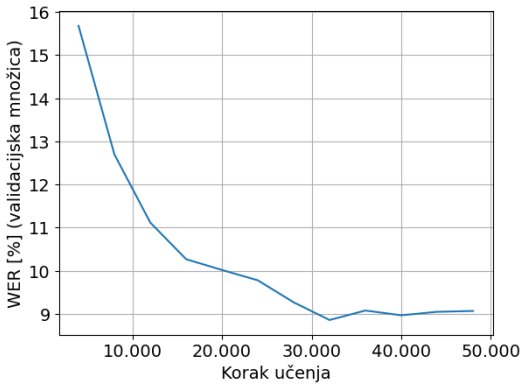
Vmesne rezultate smo shranjevali na vsakih 4.000 korakov. Iz celotnega korpusa smo za testno množico namenili 15 ur posnetkov. Preostali del korpusa smo razdelili na učno in validacijsko množico v razmerju $90:10$.

[‡]Spletna diskusija, od koder smo črpali idejo: https://github.com/openai/whisper/discussions/679.

*Seznam neverbalnih in polverbalnih glasov najdemo v datoteki s transkripti "Artur-DOC/Artur-PogovorniZapis.pdf".



Slika 3 Prikaz izgube na validacijski množici v odvisnosti od učnega koraka med učenjem modela *whisper-base*.



Slika 4 Prikaz vrednosti WER na validacijski množici v odvisnosti od učnega koraka med učenjem modela *whisper-base*.

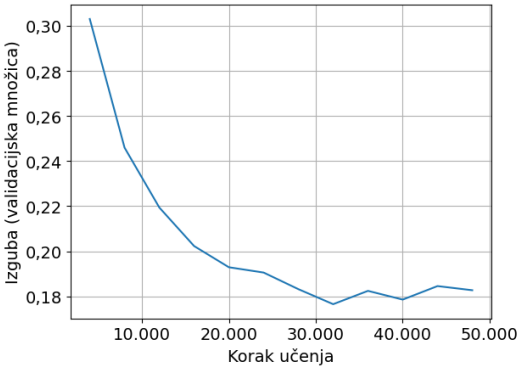
5 Rezultati učenja

Pri vrednotenju rezultatov smo transkripte normalizirali tako, da smo besedila spremenili v male črke, odstranili ločila in druge znake ter dvojne presledke.

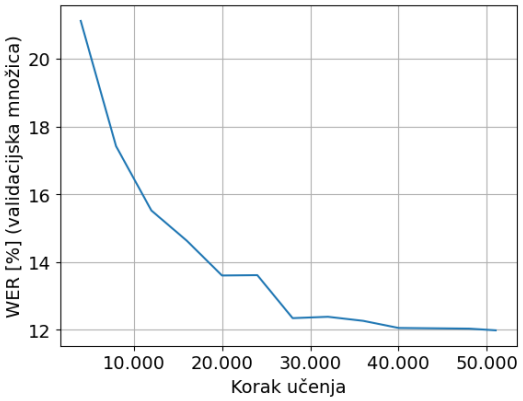
Na sliki 3 je prikazan graf vrednosti izgube na validacijski množici za model *base* med procesom učenja. Z grafa je razvidno, da je vrednost izgube začela naraščati po koraku 32.000. Prav tako je s slike 4 razvidno, da je vrednost WER na validacijski množici pri istem koraku dosegla najnižjo vrednost. Naraščanje vrednosti izgube po tem koraku je mogoče pripisati prekomernemu prileganju.

Na sliki 5 je prikazano gibanje vrednosti izgube na validacijski množici za model *tiny*, medtem ko slika 6 prikazuje graf vrednosti WER na validacijski množici v odvisnosti od učnega koraka. Podobno kot pri modelu *base* je tudi tukaj po 32.000 koraku vrednost izgube začela naraščati, najverjetneje zaradi prekomernega prileganja. Po tem koraku tudi vrednost metrike WER ni bistveno padala.

V končni implementaciji smo uporabili modela *base* in *tiny* pri koraku 32.000, kar ustreza 3,8 epohe. Model *base* je pri tem koraku na testni množici dosegel vrednost WER 11,38 %, model *tiny* pa WER 15,19 %.



Slika 5 Prikaz izgube na validacijski množici v odvisnosti od učnega koraka med učenjem modela *whisper-tiny*.



Slika 6 Prikaz vrednosti WER na validacijski množici v odvisnosti od učnega koraka med učenjem modela *whisper-tiny*.

6 Rezultati realnočasovne razpoznave

6.1 Meritve časov sklepanja

Z doučenima modeloma Whisper *base* in *tiny* smo izvedli meritve časov realnočasovnega sklepanja. Za sklepanje z modeli smo preizkusili različna zaledja, in sicer knjižnico *ONNX*, *CTranslate2* ter knjižnico *HuggingFace Transformers*. Meritve smo povprečili nad minimalno 100 izvajanji kodirnika in dekodirnika. Meritve smo izvajali na CPE na prenosnem računalniku s procesorjem AMD Ryzen 7 5800H in na pametnem telefonu Samsung Galaxy A52S 5G s procesorjem Qualcomm Snapdragon 778G. Na operacijskem sistemu Android smo z namenom pohitritve sklepanja z modeli pri uporabi zaledja *ONNX* omogočili NNAPI.

V tabeli 1 so prikazani časi sklepanja kodirnika pri uporabi različnih zaledij za sklepanje ter modelih *base* in *tiny*. Iz tabele je razvidno, da je čas sklepanja s kodirnikom pri uporabi zaledja *ONNX* najmanjši. V primerjavi z implementacijo *CTranslate2* je bila na operacijskem sistemu Android pri uporabi modela *tiny* prisotna več kot sekunda razlike.

V tabeli 2 so prikazani časi sklepanja enega

Tabela 1 Časi sklepanja s kodirnikom (krepko označeni sta konfiguraciji, uporabljeni v končni rešitvi).

Zaledje za sklepanje	Model	
	base	tiny
Windows – ONNX	253 ms	126 ms
Windows – CTranslate2	**568 ms**	245 ms
Windows – HF Transformers	473 ms	194 ms
Android – ONNX	1.107 ms	**413 ms**
Android – CTranslate2	3.648 ms	1.643 ms

Tabela 2 Časi sklepanja enega dekodirnega koraka pri požrešnem dekodiranju z enim žarkom (krepko označeni sta konfiguraciji, uporabljeni v končni rešitvi).

Zaledje za sklepanje	Model	
	base	tiny
Windows – ONNX (brez KV)	53 ms	26 ms
Windows – ONNX (KV)	31 ms	17 ms
Windows – CTranslate2	**19 ms**	13 ms
Windows – HF Transformers (KV)	13 ms	6 ms
Android – ONNX (brez KV)	193 ms	75 ms
Android – ONNX (KV)	79 ms	**40 ms**
Android – CTranslate2	85 ms	47 ms

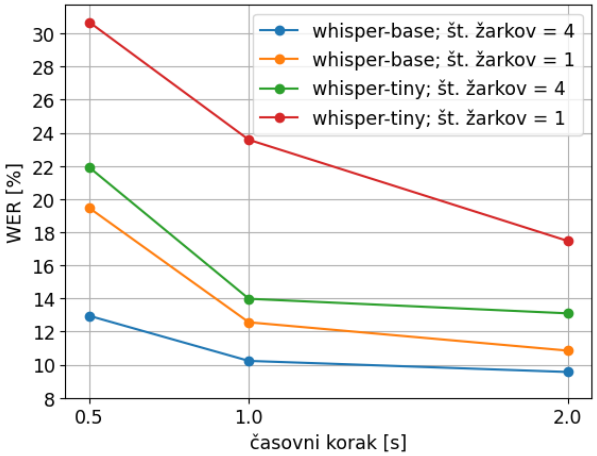
koraka z dekodirnikom. Uporabljena konfiguracija na operacijskem sistemu Windows z zaledjem *CTranslate2* je dosegala čas 19 ms na korak, kar bi pri dekodiranju 100 žetonov ustrezalo približno 2 sekundama. To je precej bolje kot naivna implementacija z zaledjem *ONNX* brez KV-predpomnjenja. Po drugi strani pa je implementacija s KV-predpomnjenjem na operacijskem sistemu Android dosegala najmanjši 40 ms čas sklepanja enega koraka pri uporabi modela *tiny*.

6.2 Meritve vrednosti WER

Za različne konfiguracije smo izvedli meritve vrednosti WER med realnočasovno razpoznavo. Pri vrednotenju smo uporabili posnetek javnega govora "Artur-J-Gvecg-P500001-avd.wav" iz testne množice korpusa Artur 1.0 v trajanju 1 ure in 52 minut. Meritve na sliki 7 prikazujejo vrednosti WER za modela *base* in *tiny* pri različnih časovnih korakih zajema zvočnega toka in različnem številu žarkov.

Nastavitev največje kakovosti v rešitvi UHO za operacijski sistem Windows uporablja 4 žarke in 2-sekundni korak. Pri tej nastavitvi model *base* dosega vrednost WER 9,56 %. Privzeta nastavitev v rešitvi UHO za operacijski sistem Android uporablja 1 žarek in 2-sekundni korak. Pri tej nastavitvi model *tiny* dosega nekoliko višjo vrednost WER 17,47 %.

Iz rezultatov vidimo, da ima velikost časovnega koraka pri zajemu zvočnega toka precejšen vpliv na vrednost WER. Tako je npr. za model *base* pri uporabi 1 žarka in 2-sekundnega časovnega koraka vrednost WER nižja za skoraj 10 odstotnih točk v primerjavi z vrednostjo pri 0,5-sekundnem časovnem koraku. Razvidno je tudi, da lahko večje število uporabljenih žarkov znatno zmanjša vrednost WER. To je vidno predvsem pri modelu



Slika 7 Prikaz vrednosti WER med realnočasovno razpoznavo nad izbranim posnetkom dolžine približno 2 ur; prikazano za modela Whisper *base* in *tiny* pri različnem številu žarkov in različnih časovnih korakih periodičnega zajema zvoka.

whisper-tiny. Po drugi strani pa se zaradi možne omejene procesorske moči uporaba več žarkov na operacijskem sistemu Android ne splača v vseh primerih.

6.3 Primerjava rezultatov WER s sorodnimi deli

V tabeli 3 podamo primerjavo vrednosti WER naše rešitve s sorodnimi deli. Naša rešitev pri uporabi *whisper-base* v primerjavi s slovenskimi sorodnimi deli dosega najnižjo vrednost WER. Podajamo tudi rezultat za rešitev Whisper-Streaming, ki pri realnočasovni razpoznavi dosega nekoliko boljši rezultat od našega. Po drugi strani pa primerjava z Whisper-Streamingom ni najbolj poštena, saj so avtorji uporabili precej večji model *whisper-large-v2*. Prav tako so razpoznavali angleški govor, za katerega so bili modeli učeni z nesorazmerno večjim številom ur kot v naši rešitvi.

7 Zaključek

Predstavili smo rešitev za gluhe in naglušne osebe, imenovano UHO, ki omogoča realnočasovno podnaslavljanje slovenskega govora s pomočjo modelov Whisper podjetja OpenAI. Modele smo doučili na superračunalniku VEGA z govornim korpusom Artur 1.0. Pri tem smo na testni množici za model *whisper-base* dosegli vrednost WER 11,38 %, za model *whisper-tiny* pa vrednost WER 15,19 %.

V aplikaciji UHO za operacijski sistem Windows nam je kljub uporabi nekoliko večjega modela *whisper-base* uspelo zagotoviti realnočasovno izvedbo s pomočjo knjižnice *CTranslate2*. V aplikaciji UHO za operacijski sistem Android smo hitrost sklepanja optimizirali z uporabo metode KV-predpomnjenja in zaledja *ONNX*. Kljub temu smo bili zaradi omejene procesorske moči mobilnih naprav prisiljeni za realnočasovno razpoznavo uporabiti manjši, manj točen model *whisper-tiny*.

Tabela 3 Primerjava naše rešitve s sorodnimi deli. Krepko označeni sta konfiguraciji, ki smo ju uporabili v končni rešitvi.

Rešitev	Model	Vrsta razpoznave	Jezik	WER
Naša rešitev	whisper-base	nad celotnimi posnetki	slo.	11,38%
Naša rešitev	whisper-tiny	nad celotnimi posnetki	slo.	15,19%
Naša rešitev*	whisper-base	realnočasovno	slo.	**10,85%**
Naša rešitev*	whisper-tiny	realnočasovno	slo.	**17,47%**
Sorodno delo s TDNN [18]	TDNN	nad celotnimi posnetki	slo	27,16%
Dnev. inform. oddaje [17]	globoke NM	nad celotnimi posnetki	slo.	15,17%
Online Notes [33]	Kaldi WFST	realnočasovno (oblak)	slo.	20%
Whisper-Streaming** [19]	whisper-large-v2	realnočasovno	angl.	5,80%

* požrešno dekodiranje, časovni korak 2 s
** testirano na korpusu ESIC

Za izboljšavo kakovosti transkriptov smo implementirali metodo iskanja z žarki. Realnočasovno izvedbo smo omogočili z uporabo intuitivnega pristopa, ki ga imenujemo aditivno dekodiranje. Ta pristop v glavnem omogoča pouporabo informacije žetonov iz prejšnje iteracije dekodiranja v naslednjih iteracijah. S tem pristopom znatno zmanjšamo skupni čas sklepanja z modeli.

Izvedli smo meritve vrednosti WER med realnočasovno razpoznavo. Rezultati so spodbudni in kažejo, da so vrednosti WER pri realnočasovni izvedbi primerljive z vrednostmi ob razpoznavi celotnih posnetkov naenkrat.

Iz meritev vrednosti WER pri realnočasovnem izvajanju smo sicer videli, da modeli *tiny* izkazujejo precej višjo vrednost WER kot modeli *base*. Žal povečanje števila žarkov na manj zmogljivejših napravah, kot so sistemi z operacijskim sistemom Android, pri dekodiranju poveča časovni zamik razpoznave govora. Možnosti rešitve za to omejitev vidimo v omejitvi uporabe na nekoliko hitrejše naprave z operacijskim sistemom Android, kot so Google Pixel s procesorji Google Tensor. Dodatno vidimo potencial za uporabo nekoliko večjega modela, kot je *whisper-base*, na hitrejših napravah z operacijskim sistemom iOS.

Nadaljnje delo bi moralo reševati zastoje, do katerih lahko pride pri menjavi govorca. Možna rešitev bi bila preskakovanje žetonov (angl. token suppression), kot je predčasno napovedan žeton za konec transkripta.

Prav tako bi nadaljnje delo moralo reševati trenutno nezmožnost doučenih modelov za večjezično razpoznavo, ki je posledica t. i. efekta katastrofalnega pozabljanja (angl. catastrophic forgetting).

ZAHVALA

J. Brest in B. Bošković priznavata financiranje članka s strani Javne agencije za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije, raziskovalni program P2-0041 – Računalniški sistemi, metodologije in inteligentne storitve.

Avtorji se zahvaljujejo konzorciju HPC RIVR www.hpc-rivr.si in organizaciji EuroHPC JU eurohpc-ju.europa.eu za financiranje raziskave z uporabo zmogljivosti sistema HPC Vega na Institutu informacijskih znanosti (www.izum.si).

LITERATURA

[1] Google , “Speech-to-Text AI: speech recognition and transcription | Google Cloud,” Dostopno na: https://cloud.google.com/speech-to-text?hl=en [6.3.2025]

[2] N. Rems Arzenšek , “Roman Demir: Bolj kot jakost je pri komuniciranju z naglušno osebo pomembna hitrost govora,” Dostopno na: https://vizita.si/leksikon/gluhost.html [23.1.2025]

[3] NIJZ Podatkovni portal , Dostopno na: https://podatki.nijz.si/, 2019

[4] WHO , “Deafness and hearing loss,” Dostopno na: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss, [23.1.2025]

[5] Zavod Združenje tolmačev za slovenski znakovni jezik , “Pravice do tolmača po zakonu - ZZTSZJ” Dostopno na: https://www.tolmaci.si/pravice-do-tolmaca-po-zakonu/ [23.1.2025]

[6] S. Jagodič , “DNEVNA: Da se bo položaj gluhih izboljšal, bodo potrebna desetletja,” Dostopno na: https://maribor24.si/drustva/dnevna-da-se-bo-polozaj-gluhih-izboljsal-bodo-potrebna-desetletja/ [23.1.2025]

[7] Microsoft , “Live Captions | Microsoft Windows,” Dostopno na: https://www.microsoft.com/en-us/windows/tips/live-captions, [23.1.2025]

[8] Google , “Samodejni podnapisi: Prikaz podnapisov pri predstavnosti in klicih v napravi - Pomoč za Funkcije za ljudi s posebnimi potrebami v Androidu,” Dostopno na: https://support.google.com/accessibility/android/answer/9350862?hl=sl, [23.1.2025]

[9] D. Verdonik; idr. , ASR database ARTUR 1.0 (audio), Slovenian language resource repository CLARIN.SI, ISSN 2820-4042, Dostopno na: http://hdl.handle.net/11356/1776, 2023

[10] D. Verdonik; idr. , ASR database ARTUR 1.0 (transcriptions), Slovenian language resource repository CLARIN.SI, ISSN 2820-4042, Dostopno na: http://hdl.handle.net/11356/1772, 2023

[11] Apache , “Parquet” Dostopno na: https://parquet.apache.org/, [24.1.2025]

[12] A. Radford; idr. *Robust speech recognition via large-scale weak supervision*, International conference on machine learning, PMLR. 2008, str. 28492-28518, Dostopno na: https://proceedings.mlr.press/v202/radford23a.html

[13] M. Barański; idr. *Investigation of Whisper ASR Hallucinations Induced by Non-Speech Audio*, ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2025, str. 1-5, Dostopno na: https://doi.org/10.1109/ICASSP49660.2025.10890105

[14] A. Ali, S. Renals *Word error rate estimation for speech recognition: e-WER*, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, str. 20-24, Dostopno na: https://www.research.ed.ac.uk/en/publications/word-error-rate-estimation-for-speech-recognition-e-wer

[15] GitHub , “Fine-Tune Whisper For Multilingual ASR with HuggingFace Transformers” Dostopno na: https://github.com/huggingface/community-events/blob/main/whisper-fine-tuning-event/fine-tune-whisper-non-streaming.ipynb [26.1.2025]
[16] Nuance , “Nuance Dragon Professional v16” Dostopno na: https://www.nuance.com/dragon/business-solutions/dragon-professional.html [27.1.2025]
[17] L. Gril; idr. *Avtomatsko razpoznavanje slovenskega govora za dnevnoinformativne oddaje*, Slovenščina 2.0: Empirične, Aplikativne in Interdisciplinarne Raziskave, 2021, 9(1), str. 60-89, Dostopno na: https://doi.org/10.4312/slo2.0.2021.1.60-89
[18] M. Ulčar; idr. *Razpoznavanje slovenskega govora z metodami globokih nevronskih mrež*, Uporabna informatika, 2019, Letn. 27, Št. 3, Dostopno na: https://doi.org/10.31449/upinf.53
[19] D. Macháček; idr. *Turning Whisper into Real-Time Transcription System*, Dostopno na: https://doi.org/10.48550/arXiv.2307.14743, 2023
[20] Silero Team, *Silero VAD: pre-trained enterprise-grade Voice Activity Detector (VAD), Number Detector and Language Classifier*, Dostopno na: https://github.com/snakers4/silero-vad, 2024
[21] OpenNMT, *CTranslate2: Fast inference engine for Transformer models*, Dostopno na: https://github.com/OpenNMT/CTranslate2 [28.1.2025]
[22] OpenAI, *Model Card: Whisper*, Dostopno na: https://github.com/openai/whisper/blob/main/model-card.md, [29.1.2025]
[23] R. P. Lippmann; idr. *Speech recognition by machines and humans*, Speech communication 22.1, 1997, str. 1-15
[24] Free On-Line Dictionary of Computing, *beam search*, Dostopno na: https://foldoc.org/beam+search, [23.4.2025]
[25] Microsoft, *ONNX Runtime*, Dostopno na: https://onnxruntime.ai/, [24.4.2025]
[26] HuggingFace, *Transformers*, Dostopno na: https://huggingface.co/docs/transformers/en/index [24.4.2025]
[27] The Linux Foundation, *PyTorch*, Dostopno na: https://pytorch.org/, [24.4.2025]
[28] HuggingFace, *Datasets*, Dostopno na: https://huggingface.co/docs/datasets/en/index [24.4.2025]
[29] HuggingFace, *Evaluate*, Dostopno na: https://huggingface.co/docs/evaluate/en/index [24.4.2025]
[30] HuggingFace, *Open ASR Leaderboard - a Hugging Face Space by hf-audio*, Dostopno na: https://huggingface.co/spaces/hf-audio/open_asr_leaderboard, [25.4.2025]
[31] D. Liu; idr. *Low-Latency Sequence-to-Sequence Speech Recognition and Translation by Partial Hypothesis Selection*, Dostopno na: https://doi.org/10.48550/arXiv.2005.11185, 2020
[32] V. Zwitter; idr. , Spoken corpus Gos 1.0, Slovenian language resource repository CLARIN.SI, ISSN 2820-4042, http://hdl.handle.net/11356/1040, 2013
[33] T. Šoltes; idr. , ONLINE NOTES: sistem za razpoznavo govora in strojno prevajanje v realnem času na ravni univerzitetnih predavanj, Uporabna informatika. 32, 1 (jul. 2024), Dostopno na: https://doi.org/10.31449/upinf.225, 2024
[34] HuggingFace, *openai/whisper-base*, Dostopno na: https://huggingface.co/openai/whisper-base [30.4.2025]
[35] GitHub , “openai/whisper: Robust Speech Recognition via Large-Scale Weak Supervision” Dostopno na: https://github.com/openai/whisper [30.4.2025]
[36] Google Play , “Seeing AI” Dostopno na: https://play.google.com/store/apps/details?id=com.microsoft.seeingai [28.7.2025]
[37] Google Play , “Project Relate” Dostopno na: https://play.google.com/store/apps/details?id=com.google.research.projectrelate [28.7.2025]
[38] Google Play , “Face Control” Dostopno na: https://play.google.com/store/apps/details?id=com.obstino.facecontrol [28.7.2025]
[39] B. Kovačič; B. Bošković , Slovenian command word speech recognition using transfer learning, Proceedings of the 9th Student Computing Research Symposium (SCORES’23). University of Primorska Press; Faculty of Electrical Engineering and Computer Science; Faculty of Computer and Information Science, str. 43-46, Dostopno na: https://doi.org/10.26493/scores23.04, 2023

Blaž Kovačič je leta 2020 diplomiral s področja elektronike na Univerzi v Mariboru. S področja računalništva in informacijskih tehnologij je leta 2025 magistriral na isti univerzi. Njegovi raziskovalni interesi vključujejo razvoj podporne tehnologije za osebe s posebnimi potrebami, interakcijo med človekom in računalnikom ter umetno inteligenco.

Janez Brest je na Univerzi v Mariboru diplomiral, magistriral in doktoriral s področja računalništva in informacijskih tehnologij, in sicer v letih 1995, 1998 in 2000. Trenutno je kot redni profesor zaposlen na Fakulteti za elektrotehniko, računalništvo in informatiko ter je tudi vodja Laboratorija za računalniške arhitekture in jezike na Inštitutu za računalništvo. Njegovi raziskovalni interesi vključujejo evolucijsko računalništvo, umetno inteligenco, optimizacijske metode, programske jezike ter vzporedno in porazdeljeno računalništvo. Je pridruženi urednik revije Swarm and Evolutionary Computation.

Borko Bošković je v letih 2004 in 2010 na Univerzi v Mariboru pridobil diplomo in doktorat iz računalništva. Trenutno je docent na Fakulteti za elektrotehniko, računalništvo in informatiko. Z Laboratorijem za računalniške arhitekture in jezike sodeluje od leta 2000. Njegovi raziskovalni interesi vključujejo evolucijsko računalništvo, optimizacijo, obdelavo naravnega jezika in vzporedno računanje.