Baltic J. Modern Computing, Vol. 10 (2022), No. 3, pp. 409–421
https://doi.org/10.22364/bjmc.2022.10.3.14

Estonian Speech Recognition and Transcription Editing Service

Aivo OLEV, Tanel ALUMÄE

Institute of Software Science, Tallinn University of Technology, Estonia

aivo.olev@taltech.ee, tanel.alumae@taltech.ee

Abstract. This paper describes the latest iteration of our Estonian speech recognition system and the publicly available transcription editing service. The system is now based on an end-to-end wav2vec2.0 model. It achieves a word error rate of 6.9% on a test set of broadcast conversations. Besides recognition it performs speaker diarization, speaker identification, Estonian language detection, and punctuation restoration. The service consists of a speech processing pipeline, web server and a web-based user interface for end-users, offering transcript editing and speaker annotation functionality. The core components of the service have been made open-source and deployed internally by multiple public and private institutions.

Keywords: Speech recognition, Estonian, wav2vec2.0, Nextflow

1 Introduction

Automatic speech recognition (ASR) has evolved rapidly in the past decade and has gone through waves of significant innovations. The most significant trend in designing and deploying ASR systems has been the adoption of deep learning methods for training speech-to-text models, using ever larger amounts of training data. Innovations in model training architectures as well as advances in hardware, namely graphical processing units (GPUs), have allowed this amount of data to be utilized in reasonable time-frames.

The most recent architectures support training of models that are not anymore highly specialized but are useful for many tasks and applications. With the use of a large amount of well-balanced training data a system using a single model can serve a variety of domains, such as broadcast speech from radio and TV, interviews conducted for scientific research and meeting recordings. Even on tasks that were not feasible for an ASR system before, such as transcribing spontaneous dialogue, it now achieves acceptable word error rate and is generally useful for such a task. This has made ASR systems practical for a significantly larger audience.

We present the latest evolution of our ASR system for transcribing Estonian speech and describe how we have deployed it as a freely available service for the general public. Compared to the previous version (Alumäe et al., 2018), the ASR system now also performs language identification and the speech-to-text model is now an end-to-end system that is based on a large pretrained wav2vec2.0 (Baevski et al., 2020) model and is fine-tuned on annotated Estonian speech data. The performance of the ASR system is compared to the earlier version that used a Kaldi-based (Povey et al., 2011) model.

The publicly available service consists of a scalable full speech transcription pipeline for batch processing of long audio recordings[1,2] and a web-based graphical user interface (GUI)[3,4] that allows end-users to submit audio files for transcription, see the execution progress, edit the speaker-segmented transcripts in a user-friendly interface and download the transcripts in various formats. The system is also available in open source form for installing on-premise, and is used in such form by several Estonian companies. The system is being developed within the Estonian national language technology program.

2 Speech-to-text pipeline

Our speech-to-text pipeline consists of the following steps:

- Speech activity detection
- Speaker diarization
- Language identification (Estonian *vs.* other)
- Speech-to-text, including generating word timemarks
- Punctuation restoration
- Inverse text normalization
- Speaker recognition: identification of known public figures

The speaker diarization, punctuation restoration and speaker recognition components in the pipeline are identical to those of our 2018 system (Alumäe et al., 2018) and will not be described in detail in this paper. For speech activity detection, we replaced the earlier system based on Gaussian mixture models (GMMs) with an off-the-shelf neural model Silero VAD (Silero Team, 2021). We've found the model to perform well in all different conditions. The other components of the pipeline will be described in the rest of this section.

2.1 Language identification

In order to avoid transcribing utterances in other languages using the Estonian model, we apply a spoken language identification model that aims to filter out speech that is not in Estonian. The language identification system is based on a model pretrained on

[1] `https://github.com/taltechnlp/est-asr-pipeline`
[2] `https://github.com/taltechnlp/est-asr-backend`
[3] `https://github.com/aivo0/est-asr-frontend`
[4] `https://github.com/aivo0/est-asr-web-api/`

Table 1: Training data for speech recognition.

(a) Acoustic model training data.

Source	Amount (h)
Broadcast speech	591
Spontaneous speech (Lippus, 2011)	53
Elderly speech corpus (Meister, 2021)	49
Talks, lectures	38
Parliament speeches	31
Total	761

(b) Language model training data.

Source	Tokens (M)
ENC19 Web Scrape	526
ENC19 Ref. Corpus	185
ENC19 Wikipedia	35
OpenSubtitles	98
Speech transcripts	6.1
Subtitles from ETV	3.8
Total	854

the VoxLingua107 corpus (Valk and Alumäe, 2021) that contains automatically scraped data from YouTube for 107 languages. Pretraining is implemented using the Speech-Brain framework (Ravanelli et al., 2021) and the model is publicly available[5]. However, we don't use the model's predictions directly for language classification. We use it only for extracting utterance embeddings, which are then classified into Estonian or non-Estonian using a binary logistic regression model. Training data for the regression model comes from the VoxLingua107 corpus (1000 randomly selected Estonian utterances and 3000 randomly selected utterances from other languages) and from the corpus of Estonian L2 speech (Meister and Meister, 2015).

2.2 Speech recognition

2.2.1 Data. Speech data that is used for training the speech recognition models is summarized in Table 1a. Only the duration of the segments containing transcribed speech is shown, i.e., segments containing music, long periods of silence and untranscribed data are excluded.

Most of the training data has been transcribed by our lab in the last 15 years (Meister et al., 2012), except the Corpus of Estonian Phonetic Corpus of Spontaneous Speech that originates from the University of Tartu (Lippus, 2011). Compared to our 2018 system (Alumäe et al., 2018), the amount of acoustic training data has increased from 268 to 761 hours. Most of the increase is due to the addition of ERR2020 corpus[6] that contains 389 hours of broadcast speech, TV talkshows and press conferences and was transcribed in 2020. In addition, 49 hours of speech produced by elderly (aged 60+ years) speakers of Estonian (Meister, 2021) has been added to the training data.

Textual data used for training the language model (LM) is listed in Table 1b. Most of the data originates from the subcorpora of the Estonian National Corpus 2019 (ENC2019) (Kallas and Koppel, 2019): Estonian web, a reference corpus containing balanced data from the web, newspapers and books, and Estonian Wikipedia. We also use all available Estonian data from the OpenSubtitles corpus (Lison and Tiedemann, 2016) and scraped subtitles for the deaf and hard of hearing (SDH) from the Estonian national television

[5] `https://huggingface.co/TalTechNLP/voxlingua107-epaca-tdnn`

[6] `http://bark.phon.ioc.ee/lw/korpused/ERR2020.html`

(ETV). Compared to our 2018 system, the amount of textual data has increased from 690 million to 854 million tokens.

Before using the text data for LM training, text normalization is performed. Texts are tokenized, split into sentences and recapitalized, i.e., converted to a form where names and abbreviations are correctly capitalized while normal words at the beginning of sentences are written in lower case. This is done with the help of the EstNLTK morphological analyzer (Laur et al., 2020). Numbers and other non-standard words are expanded into words using hand-written rules.

2.2.2 End-to-end system. We experiment with two end-to-end speech recognition models that are fine-tuned from the XLS-R-300M and XLS-R-1B models (Babu et al., 2021) using the connectionist temporal classification (CTC) objective. XLS-R is a family of large wav2vec2.0 (Baevski et al., 2020) models pretrained on unlabeled multilingual data. A wav2vec2.0 model is trained by jointly solving a contrastive task over masked latent speech representations and learning a quantization of the latents shared across languages. The model contains a convolutional feature encoder that maps raw audio to latent speech representations which are fed to a Transformer network that outputs context representations. XLS-R models are trained on 436K hours of unannotated speech data in 128 languages, including around 11K hours of Estonian data originating from the VoxPopuli (Wang et al., 2021), VoxLingua107 (Valk and Alumäe, 2021) and Mozilla Common Voice (Ardila et al., 2020) corpora.

We fine-tune the XLS-R models on all of our annotatated speech data (see Table 1a) using a standard procedure: transcripts are mapped to sequences of case-sensitive characters, using the orthographic form of the words, with numbers expanded and punctuation marks removed. Intra-word space is mapped to a special character. The XLS-R-300M model is then fine-tuned using the CTC objective for 320 000 updates. We train using an effective total dynamic batch size of 25 600 000 samples (equaling 1600 seconds). A triangular learning rate schedule is used, with linear warm-up phase lasting for 10% of the training time, constant learning rate of 0.0003 that is used during the next 40% of the training time, and a linear decay to a learning rate that is equal to 0.05 of the maximum during the rest of training time. The convolutional feature extraction layers of the wav2vec2.0 model are frozen during fine-tuning. Feature-space SpecAugment (Park et al., 2019) and LayerDrop (Fan et al., 2020) are applied. Fine-tuning is implemented using the *fairseq* toolkit (Ott et al., 2019). Fine-tuning takes around five days on four NVidia A100 GPUs. The model is available at the HuggingFace model repository[7]. The larger XLS-R-1B model is finetuned using a similar setup, except here we use a smaller learning rate of 0.00003, a smaller effective batch size of 640 seconds, and also disable LayerDrop and reduce the SpecAugment time-dimension masking probability from 0.75 to 0.5. Those modifications were needed to get the larger model to converge in a more stable manner.

The CTC-trained model can be used for speech recognition as is, but using shallow fusion with a LM often helps to improve the accuracy further. For the end-to-end model LM, we use a subword vocabulary of 40 000 tokens, generated with the SentencePiece library (Kudo and Richardson, 2018) using the unigram LM method. A 4-gram LM is

[7] `https://huggingface.co/TalTechNLP/xls-r-300m-et`

Table 2: Word error rates on different evaluation sets of the current system and its predecessors, and relative improvement in WER with regard to the 2018 system.

	Broadcast conversations		Conference talks		User recordings		Avg.	Rel. impr. w.r.t 2018
	Dev	Test	Dev	Test	Dev	Test		
Our 2014 system (Alumäe, 2014)	18.0	17.9	23.7	26.3	N/A	N/A	N/A	
Our 2018 system (Alumäe et al., 2018)	11.0	8.1	14.5	12.9	29.4	22.7	16.4	
Current, Kaldi-based	10.1	7.9	13.8	10.4	23.0	18.3	13.9	-15%
Current, XLS-R-300M	9.9	7.3	10.9	9.6	22.7	16.6	12.8	-22%
Current, XLS-R-1B	9.1	6.9	10.4	9.0	20.5	15.7	11.9	-27%

trained for each text corpus using Kneser-Ney smoothing and the final LM is interpolated from the individual models, using interpolation weights optimized on development data.

Although it is possible to obtain timestamps of the decoded words from the CTC-based end-to-end model, such timestamps are generally not as reliable as those of a hidden Markov model based system. Therefore, we realign the recognition hypotheses of the end-to-end model using the Kaldi acoustic model described below. Pronunciations of the words in an utterance are automatically generated using a G2P tool and the pronunciation dictionary and alignment graph are generated on the fly.

2.2.3 Kaldi-based system. We compare the end-to-end model to a Kaldi-based (Povey et al., 2011) model. The Kaldi acoustic model is a factored time-delay neural network (TDNN-F) acoustic model (Povey et al., 2018) with a "multistream" architecture (Han et al., 2021). The multistream architecture uses five basic TDNN-F layers and 17 multistream layers in a $(3-6-9)$ dilation configuration. The acoustic model has around 21 million parameters. Speaker adaptation is done using i-vectors. We use standard Kaldi multi-condition data augmentation (Ko et al., 2017) for acoustic training data: training data is 3-fold speed perturbed, and the speed perturbed data is in turn augmented with reverberation, various environment sounds, music or babble noise from the MUSAN corpus (Synder et al., 2015). This increases the amount of training data by 15-fold in total. The acoustic model is trained for six epochs on the augmented data. It takes around seven days using six NVidia P100 GPUs.

The LM of the Kaldi-based system uses 200 000 compound-split units. A pruned 4-gram LM is used during the decoding and a recurrent LM is used for rescoring the lattices. After rescoring, we apply out-of-vocabulary (OOV) word recovery to reconstruct the orthographic transcripts of the decoded unknown words. Language modeling and OOV recovery is described in more detail in (Alumäe et al., 2018).

2.2.4 Results. Case-insensitive word error rates (WER) of our system on several datasets are shown in Table 2. Three diverse development and test sets are used: (1) broadcast conversations (mostly talk shows from different Estonian radio stations), (2)

Table 3: Word error rates on Estonian CommonVoice testsets.

	CommonVoice	CommonVoice 8	CommonVoice 9
Kaldi TDNN-F	13.72	14.61	15.33
+ RNNLM	11.87	12.67	13.28
wav2vec2.0 XLS-R-300M	12.55	13.41	14.06
+ 4-gram LM	10.59	11.40	11.99
wav2vec2.0 XLS-R-1B	10.13	10.82	11.51
+ 4-gram LM	8.81	9.39	10.02

talks from a local computational linguistics conference, and (3) a random sample of recordings uploaded to our system by end-users, including mostly interviews and meeting recordings. Results of both the Kaldi-based system as well as the end-to-end ASR model are shown and WERs of the 2018 and 2014 systems are given for comparison. All the results are based on fully automatic processing of long speech recordings, i.e., speech activity detection, speaker diarization and speech segmentation were used to obtain speech utterances, and WER was calculated based on the alignment of decoded and time-stamped words and the reference transcripts in segment time-marked (STM) files. As can be seen, the end-to-end models result in lower WER numbers than the Kaldi-based system, especially in more challenging datasets. The average relative improvement of the best current system with regard to our 2018 system is 27%.

Recently, the accuracy of end-to-end speech recognition models is often demonstrated on the Mozilla Common Voice datasets (Ardila et al., 2020). Therefore, we also include such results for future reference. Table 3 shows the WERs of different decoding setups using the Estonian test split of the CommonVoice[8], CommonVoice 8.0[9] and CommonVoice 9.0[10] datasets. The case-insensitive WER results are calculated after stripping punctuation marks from the reference transcripts. The CommonVoice dataset contains individual utterances, not long speech recordings. Speech activity detection, diarization and language identification are therefore not used in this experiment.

Results on CommonVoice show that the smaller end-to-end model fused with an external LM achieves around 10% relative decrease in WER, compared to the system that uses Kaldi-based decoding and rescoring with the recurrent LM. The larger end-to-end model gives around 17% further relative improvement over the smaller model, and around 25% relative improvement over the Kald-based setup.

We also attempted to integrate Transformer-based LM with the end-to-end model, but this resulted in minor or no improvements in WER, while the decoding time increased several times. Therefore, for practical reasons we don't use a Transformer-based LM in the current system.

2.3 Inverse text normalization

[8] https://huggingface.co/datasets/common_voice

[9] https://huggingface.co/datasets/mozilla-foundation/common_voice_8_0

[10] https://huggingface.co/datasets/mozilla-foundation/common_voice_9_0

Table 4: Some examples of spoken numerical expressions and the corresponding written forms.

Spoken form	Written form
kaksteist	12
kaheteistkümne	12
kaheteistkümnes	12.
kaheteistkümneks	12-ks
kaheteistkümnendate	12.-te
kahesaja üheksakümne kuuele tuhandele	296000-le

```
{ "word": "2001.",
  "start": 56.12,
  "end": 57.14,
  "unnormalized_words": [
    { "end": 56.33,
      "start": 56.12,
      "word": "kahe"},
    { "end": 56.54,
      "start": 56.33,
      "word": "tuhande"},
    { "end": 57.14,
      "start": 56.84,
      "word": "esimese"}]}
```

Fig. 1: Example of structure representing a recognized numerical expression.

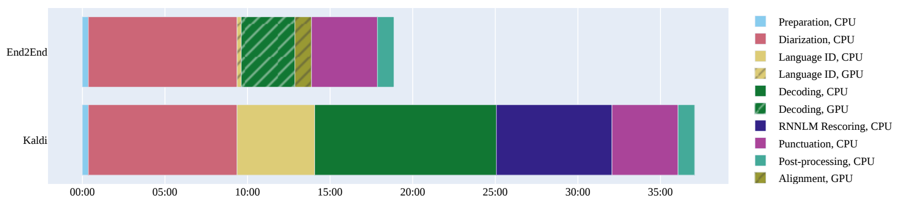


Fig. 2: Timeline of different processing steps of the two systems when processing a 53-minute radio show.

Human readers usually prefer that entities like dates, times, addresses and currency amounts in text are displayed in a nicely formatted way. The conversion of purely textual tokens into a formatted form is called inverse text normalization (ITN). Currently, our ITN component only deals with transforming numbers. However, since Estonian is an inflective language and inflections are also applied to numbers, the implementation of this process cannot be trivial (see Table 4). Our ITN component is implemented using hand-designed finite state transducer rules that follow the recommendations of the Estonian orthography for writing numbers. The Pynini library (Gorman, 2016) is used for the implementation.

When converting words into written form, we also preserve the original word sequence that corresponds to the number. Both the orthographic words and the original spoken words are given in the final structured transcript (see Figure 1 for a stripped sample).

2.4 Runtime performance

The Kaldi-based ASR pipeline is able to process most speech files faster than realtime, when only using CPUs (we usually configure it to use two CPUs during decoding). However, decoding with large wav2vec2 models is prohibitively slow on a CPU. Therefore, the new pipeline that relies on end-to-end ASR models requires a GPU for a few processing steps. This actually makes the new pipeline faster than the old one. Figure 2 shows the timeline of processing a 53-minute radio talkshow using the two systems.

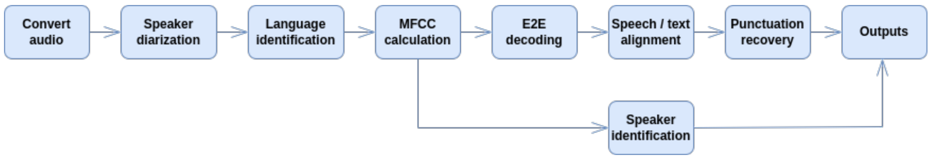


Fig. 3: Computational graph of our Nextflow pipeline.

Steps that are offloaded to a GPU are depicted with a striped fill. As can be seen, the new pipeline can process the 53-minute file in about 19 minutes, while using a GPU (NVidia P100) only for 5 minutes to do language identification, the actual decoding and speech-text alignment. This allows us to process four files in parallel on a server with a single GPU, with a realtime factor of about ⅓.

3 Deployment

3.1 Backend

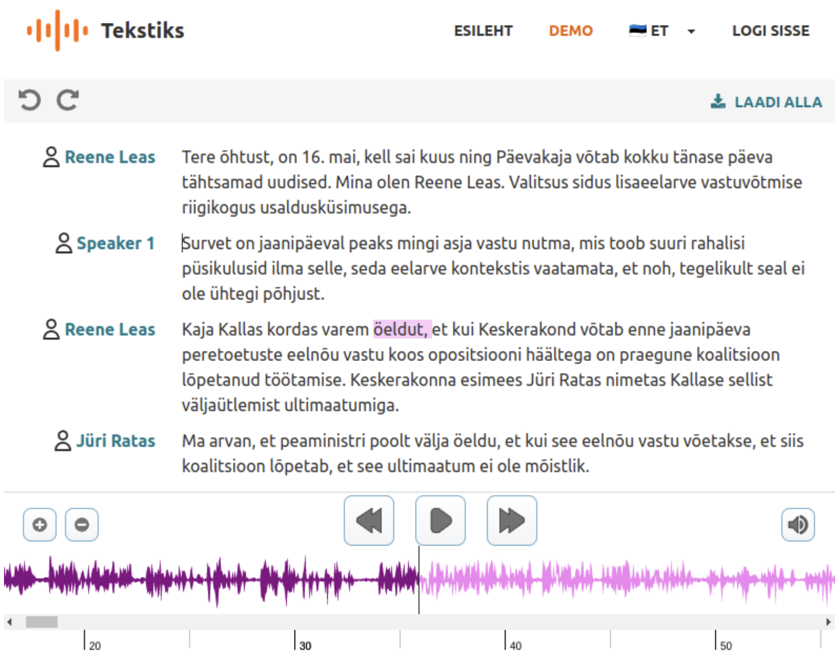
The implementation of the pipeline is based on a workflow management system (WMS) Nextflow. Nextflow is a general purpose WMS that was created to address challenges in the bioinformatics field where data processing pipelines utilize many external tools, many data streams with large quantities of data, the processing of which can span days or weeks. Nextflow addresses numerical instability and reproducibility of results, efficient parallel execution, error tolerance, execution provenance and traceability (Di Tommaso et al., 2017). Usage of Nextflow in NLP is still uncommon but it is being used in the field of NLP by at least one NLP research group to build various NLP pipelines (Reynaert et al., 2015)[11,12].

Nextflow is a general purpose tool because it allows to break down a workflow into distinct processing steps which can run code written in any language or tool that can be run on Linux. Figure 3 depicts the directed acyclic graph (DAG) generated by Nextflow after the execution of our processing pipeline. Nextflow handles the dependencies between those steps automatically and takes care of buffering, error handling and resuming steps so that no step has to be repeated if the inputs have not changed.

The main benefits for our system were firstly that Nextflow reports the progress of each processsing step. Compared to our previous Makefile-based system, the source script became shorter, more modular and easier to comprehend, modify and reuse by others. Thirdly, there were practical deployment and runtime benefits such as support for most of the popular cluster environments and support for container technologies (e.g. Docker).

Nextflow has a commercial user interface offering but for us its logging capabilities and sending of progress information to a custom URL were enough. We built a web API server to make it easier to upload speech recordings and receive real-time progress

[11] `https://github.com/proycon/nederlab-pipeline`
[12] `https://github.com/proycon/aNtiLoPe`



Reene Leas: Tere õhtust, on 16. mai, kell sai kuus ning Päevakaja võtab kokku tänase päeva tähtsamad uudised. Mina olen Reene Leas. Valitsus sidus lisaeelarve vastuvõtmise riigikogus usaldusküsimusega.

Speaker 1: Survet on jaanipäeval peaks mingi asja vastu nutma, mis toob suuri rahalisi püsikulusid ilma selle, seda eelarve kontekstis vaatamata, et noh, tegelikult seal ei ole ühtegi põhjust.

Reene Leas: Kaja Kallas kordas varem öeldut, et kui Keskerakond võtab enne jaanipäeva peretoetuste eelnõu vastu koos opositsiooni häältega on praegune koalitsioon lõpetanud töötamise. Keskerakonna esimees Jüri Ratas nimetas Kallase sellist väljaütlemist ultimaatumiga.

Jüri Ratas: Ma arvan, et peaministri poolt välja öeldu, et kui see eelnõu vastu võetakse, et siis koalitsioon lõpetab, et see ultimaatum ei ole mõistlik.

Fig. 4: Transcription editing user interface.

information and results. The server stores processing pipeline progress and queue information and it can make predictions and provide users with information on how long the queued task is going to take. This is important for a public service which is mostly used during business hours and is hosted on a computing cluster with limited processing capacity. For the publicly available ASR service we also have a separate web server that stores user account information and provides an API for the frontend user interface. It enables the frontend application to frequently save user text editing state and supports *undo* and *redo* actions for every change to the transcription performed within a single editing session.

3.2 Frontend

The frontend of the ASR system is a modern web application built using the SvelteKit application framework [13]. The primary functionality provided by the user interface (UI) is the ability to edit automatically generated speech transcriptions. Although the ASR results are useful for users, many users need to make corrections to the transcriptions. For this purpose, the UI provides an integrated audio player with a navigable sound waveform, as shown in Figure 4. The currently playing word is highlighted and users can also click on a word to seek the audio to the corresponding location in the speech recording. In order to achieve this, every word has hidden timecodes and the text editor

[13] `https://kit.svelte.dev/`

enables editing of HTML rather than plain text. Besides just transcribed text, the UI also displays the results of speaker diarization and word recognition confidence information. The transcribed text is initially split according to speaker turns. Each speaker is either given a name - in case the speaker was recognized during speaker identification - or given a unique name, e.g. *Speaker 1*. User is free to rename one or all occurrences of the speaker, add more speakers or remove them altogether.

To support HTML editing on the web, a text editing library has to be used[14] because the HTML editing support across web browsers is inconsistent, its standardization is incomplete and also because there are multiple ways in HTML to represent the same visual result which can lead to problems when for example accepting pasted-in input. The frontend library for HTML text editing is Tiptap[15] which is based on ProseMirror[16]. ProseMirror documents adhere to a strict schema which ensures that there is only one way to represent the same visual result. It also utilizes immutable state and transactions. Every change to the document state is achieved through transactions. That allows changes to be rolled back or applied again (undo, redo) and ensures consistency between transactions defined through API calls, which get triggered for example when user clicks a button on the editor toolbar, and modifications to the HTML DOM tree made by a user directly in the editor. This approach addresses out-of-order and other synchronization problems which would occur when frequently automatically saving the editor state to the backend server via (inherently asynchronous) API calls. As every user action, including *undo* and *redo*, is represented as a transaction, the message with less operations can always be dismissed for the one with more operations.

Every user of the UI has to register an account and be logged in before using the system in order to protect their privacy. We plan to allow users to explicitly permit the usage of their speech data for scientific usage. Previously some of the uploaded speech data has been anonymously used as part of an unlabelled speech corpora (Asadullah and Alumäe, 2018).

3.3 Usage

The system described in this paper is deployed at the website `https://tekstiks.ee`. The system is free to use for the general public, currently without any usage limitations. The aforementioned study of most frequent users (Olev, 2019) found that the most users were using the platform for interview and meeting transcriptions as well a lecture transcriptions. The system has also been used by the office of the Estonian government for transcribing government press conferences.

Figure 5 shows the number of weekly uploads to the transcription service during the period from the end of February, 2021 to May, 2022. The usage statistics shows significant seasonal differences. The average uploads per week over the studied time period is 891 (127 per day).

The open-sourced ASR pipeline is currently used on-premise by several Estonian companies and public institutions, operating in media monitoring, call center communication analysis and media production.

[14] `https://dvcs.w3.org/hg/editing/raw-file/tip/editing.html`
[15] `https://tiptap.dev/`
[16] `https://prosemirror.net/`

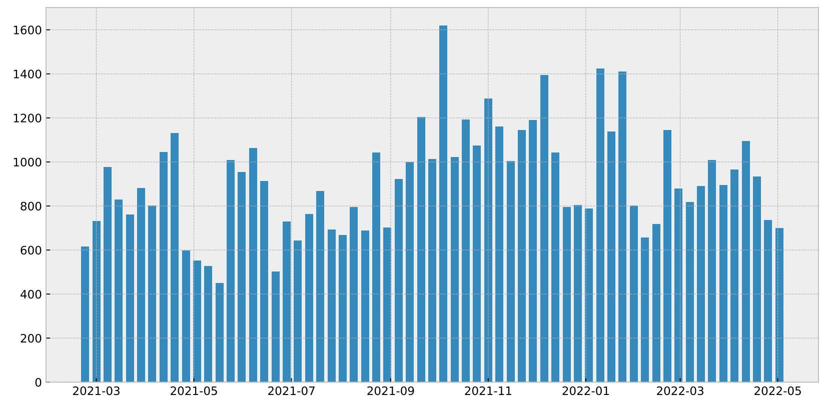


Fig. 5: Weekly upload counts of the public Estonian speech transcription service `https://tekstiks.ee`

4 Conclusion

This paper presented the latest version of our Estonian speech transcription system system. The system uses a modern end-to-end wav2vec2.0 architecture and is able to benefit from a pre-trained model trained on 436K hours of unannotated speech data in 128 languages. It achieves a 7.3% WER on test set of broadcast conversations, 9.6% on conference talks and 16.6% on various user-submitted recordings. The average relative improvement compared to our 2018 system is 22%. This confirms that it is possible for smaller languages to benefit from such a pre-trained model.

The system has been designed to handle semi-spontaneous speech from various domains, such as broadcast conversations, lecture recordings and interviews, and be useful for a large audience of users, additionally performing punctuation restoration, speaker diarization, speaker identification and language identification. The processing pipeline is based on a workflow management system (WMS). The WMS has enabled monitoring of the progress of the pipeline execution. This has allowed us to provide real-time progress information to end-users. The system has been deployed and hosted as a publicly available service. It is based on distinct modular components which are available in open-source form for installing on-premise. The service provides a graphical user interface transcription editing, interactive recording listening and speaker annotation functionalities.

Acknowledgements

This work has been partially conducted in the project “ICT programme” which was supported by the European Union through the European Social Fund. The study has been supported by the European Regional Development Foundation (the project “Centre of Excellence in Estonian Studies”). The authors acknowledge the TalTech supercomputing resources made available for conducting the research reported in this paper.

References

Alumäe, T. (2014). Recent improvements in Estonian LVCSR, *SLTU*.

Alumäe, T., Tilk, O., Asadullah (2018). Advanced rich transcription system for Estonian speech, *Baltic HLT*, pp. 1–8.

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., Weber, G. (2020). Common Voice: A massively-multilingual speech corpus, *LREC*.

Asadullah, Alumäe, T. (2018). Data augmentation and teacher-student training for LF-MMI, *21st International Conference on Text, Speech and Dialogue*.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., Auli, M. (2021). XLS-R: Self-supervised cross-lingual speech representation learning at scale, *arXiv preprint arXiv:2111.09296* .

Baevski, A., Zhou, Y., Mohamed, A., Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations, *Advances in Neural Information Processing Systems* **33**, 12449–12460.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P., Palumbo, E., Notredame, C. (2017). Nextflow enables reproducible computational workflows, *Nature Biotechnology* **35**, 316–319.

Fan, A., Grave, E., Joulin, A. (2020). Reducing transformer depth on demand with structured dropout, *ICLR*.

Gorman, K. (2016). Pynini: A Python library for weighted finite-state grammar compilation, *SIGFSM Workshop on Statistical NLP and Weighted Automata*, pp. 75–80.

Han, K. J., Pan, J., Tadala, V. K. N., Ma, T., Povey, D. (2021). Multistream CNN for robust acoustic modeling, *ICASSP*, pp. 6873–6877.

Kallas, J., Koppel, K. (2019). Estonian National Corpus 2019. `https://doi.org/10.15155/3-00-0000-0000-0000-08565L`

Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition, *ICASSP*.

Kudo, T., Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *arXiv preprint arXiv:1808.06226* .

Laur, S., Orasmaa, S., Särg, D., Tammo, P. (2020). EstNLTK 1.6: Remastered Estonian NLP pipeline, *LREC*, pp. 7152–7160.

Lippus, P. (2011). *The acoustic features and perception of the Estonian quantity system*, Ph.d. thesis, University of Tartu.

Lison, P., Tiedemann, J. (2016). OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles, *LREC*.

Meister, E. (2021). A corpus of elderly Estonian speech (under development). `https://doi.org/10.15155/9-00-0000-0000-0000-00220L`

Meister, E., Meister, L. (2015). Development and use of the Estonian L2 corpus, *Workshop on Phonetic Learner Corpora*, pp. 45–47.

Meister, E., Meister, L., Metsvahi, R. (2012). New speech corpora at IoC, *XXVII Fonetiikan päivät*.

Olev, A. (2019). *Web application for authoring speech transcriptions*, Ms. thesis, Tallinn University of Technology.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling, *NAACL-HLT: Demonstrations*.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition, *Interspeech*.

Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohamadi, M., Khudanpur, S. (2018). Semi-orthogonal low-rank matrix factorization for deep neural networks, *Interspeech*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., Vesel, K. (2011). The kaldi speech recognition toolkit, *ASRU*.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., Mori, R. D., Bengio, Y. (2021). SpeechBrain: A general-purpose speech toolkit. arXiv:2106.04624.

Reynaert, M., Van Gompel, M., Sloot, K., Van den Bosch, A. (2015). Piccl: Philosophical integrator of computational and corpus libraries, *Proceedings of CLARIN Annual Conference 2015*.

Silero Team (2021). Silero VAD: pre-trained enterprise-grade voice activity detector (VAD), number detector and language classifier, `https://github.com/snakers4/silero-vad`.

Synder, D., Chen, G., Povey, D. (2015). MUSAN: A music, speech, and noise corpus. arXiv:1510.08484.

Valk, J., Alumäe, T. (2021). VoxLingua107: a dataset for spoken language recognition, *Spoken Language Technology Workshop (SLT)*, pp. 652–658.

Wang, C., Rivière, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., Dupoux, E. (2021). Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation.
`https://arxiv.org/abs/2101.00390`

Received August 19, 2022 , accepted August 25, 2022