Human Language Technologies – The Baltic Perspective
I. Skadiņa and R. Rozis (Eds.)

doi:10.3233/978-1-61499-701-6-66

Towards the First Dictation System for Latvian Language

Askars SALIMBAJEVS [a,1]
[a] *Tilde, Latvia*

Abstract. In this paper, we introduce the first dictation system for the Latvian language. We present its main features, the details of the automatic speech recognition (ASR) system used in this service, software architecture, and an evaluation of recognition quality. The service will provide Latvian-speaking people with the opportunity to dictate text in Latvian into their computers. The presented system achieved a word error rate (WER) of 23.86% in evaluation of dictation scenario, which is a good result for a system in the beta stage.

Keywords. Speech recognition corpus, dictation systems, Latvian, language model adaptation

1. Introduction

Automatic speech recognition (ASR) technologies for Latvian have a relatively short history. Even as recently as three years ago (i.e., in 2013 and before), there was no orthographically annotated speech corpus that would be suitable for ASR purposes. This greatly limited any possible research in this field. However, there have been attempts to develop ASR systems for Latvian without an annotated speech corpus by using acoustic model bootstrapping methods [1] in the Quaero project [2].

Since the creation of the first orthographically and phonetically annotated speech corpus for Latvian [3] – the *Latvian Speech Recognition Corpus* (LSRC), ASR technologies for Latvian have been actively researched (e.g., [4] and [5]). This research has also lead to the development of practical applications with ASR capabilities [6][7][8][9]. However, the technology had yet to reach a level where it is applicable in dictation scenarios in text editors.

For the development of dictation systems, it was necessary to: 1) create a specific corpus that would allow to evaluate and improve acoustic models of such systems, 2) train acoustic models for online speech recognition, 3) adapt the language model, and 4) develop a dictation software that can be used on typical consumer hardware. Therefore, in this paper, we present our first results on tackling these problems. The paper is further structured as follows: 1) section 2 describes the acoustic and language models for the Latvian dictation system, 2) section 3 describes the software implementation, 3) section 4 presents the evaluation of speech recognition quality and software optimization, and 4) the paper is concluded in section 5.

[1] Corresponding Author: Askars Salimbajevs, SIA Tilde, Vienibas gatve 75A, Riga, Latvia, LV-1004; E-mail: askars.salimbajevs@tilde.lv.

2. Speech Recognition System

Any speech recognition system contains two models: 1) acoustic model and 2) language model. The first model describes acoustic properties of phonemes and words. The second is used to evaluate word sequences coming from the acoustic model and to select the best hypothesis from the semantical, grammatical, and syntactical points of view.

Both of the acoustic and language models that we use for dictation are based on the corresponding models from Latvian Speech-To-Text Transcription Service [7].

2.1. Acoustic Model

We use the Hidden Markov Model – Deep Neural Network (HMM-DNN) acoustic model (AM) with iVector adaptation [10] and PLP features [11]. The model is implemented in the open-source Kaldi toolkit [12]. The phoneme repository consists of 37 grapheme-based phonemes, 1 unified filler\silence model, and 1 model for fragmented words and other garbage.

The model is expected to perform in real-time, generating a hypothesis as the audio is streaming. This means that the model should not be too complex. We use a Switchboard online recipe from Kaldi, which creates a p-norm deep neural network with 5 hidden layers (each having about 3000 neurons) and trains it with sliding window cepstral normalization and iVector adaptation.

The acoustic model (AM) is trained on a 100 hour-long LSRC [3] and a 9 hour-long *Latvian Dictated Speech Corpus* (LDSC) [13].

The LDSC is another speech corpus created specifically for speech recognition in Latvian. It consists of recordings of people dictating various texts (documents, social network posts) using dictation commands (punctuation, newline, etc.) and was created to facilitate the development of dictation systems for Latvian. The total length of the LDSC is 9 hours 19 minutes and 46 seconds. Roughly 1 hour of the LDSC is used for evaluation of the system, and all other data is used in training.

At the beginning, the model is trained using only data from LSRC. First, monophones are trained using about 1% of the LSRC, then the first triphone model is trained using about 10% of the LSRC, and so on. The data from LDSC is added after Speaker Adaptive Training (SAT) of a triphone model that is using 100% of the LSRC data. The SAT is then repeated. After that, alignments from the SAT model are used to train the deep neural network.

2.2. Language Model

For language model (LM) training, we use a 44 million sentence text corpus, which was automatically collected from Latvian web news portals. The corpus contains about 905M million tokens, both of words and punctuation symbols. In order to filter the corpus from noise/garbage and adapt the corpus for the dictation task, the following processing procedure was used:

- The raw text is processed with natural language processing tools that perform tokenizing, garbage filtering (for example, mixed case tokens, non-alphanumeric tokens), number conversion from digits to words with correct inflection (for this, we use a module from the Latvian text-to-speech system [14]), some abbreviation expansion, and true casing.

- Next, punctuation and special symbols are replaced with their respective pronunciations.
- Then formatting and action commands were artificially added as separate sentences.
- Finally, "New line" commands were appended after every second sentence in the text corpus.
- Each sentence is then verified by a spellchecker, and sentences that contain 2 or more errors are filtered out.
- Vocabulary is extracted from sentences containing no spelling errors.

As the result, the processed corpus contains 35M sentences, 674M tokens, and a vocabulary of 872K words. On this text corpus, two n-gram language models are trained with Kneser-Ney [15] smoothing:

- A heavily pruned 2-gram model for first-pass decoding.
- A big 3-gram unpruned model for lattice rescoring.

3. Dictation Software

Our dictation software uses client-server architecture: 1) dictation web-service is hosted in the cloud and 2) client software is installed on the personal computers of end users. Full-duplex communication is implemented using WebSocket technology: 1) the client connects to the server and begins to stream audio data and 2) the server receives the stream, processes it, and sends recognized text back to client.

3.1. Dictation Web-Service

The dictation service is integrated in the Latvian Speech-To-Text Transcription Service [7], which is based on the Full-duplex Speech-to-text System for Estonian [16]. It consists of a single master server and multiple workers, which can be used independently. The overall architecture of the web-service is presented in *Figure 1*.

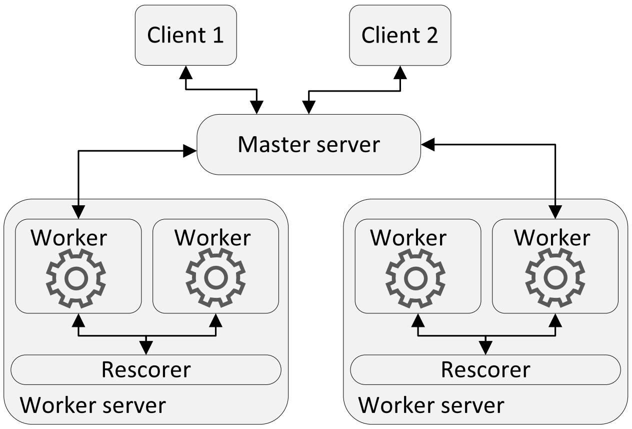

Figure 1. Web-service architecture.

This architecture makes it possible to easily scale up the system by just adding more servers and hardware. For example, workers and master servers can be hosted on different machines.

This setup, however, consumes a lot of random access memory (RAM), as each worker stores its own copy of LM in the memory. A 3-gram LM with an 800K word vocabulary can take 3-3.5 GB of RAM. In order to optimize memory usage, we use "shared rescoring": only a 2-gram pruned model is left inside the worker, but the big 3-gram LM for rescoring is moved to a separate process called "rescorer", which is shared among multiple workers. Rescorer is a separate multithreaded program that receives lattices from workers, performs rescoring, and sends rescored lattices back to workers. That way, only 1 copy of the big LM is stored in the memory. Rescorer can be run on a separate machine; however, in order to keep latency small, it's better to run rescorer on the same machine as workers.

3.2. Dictation Client

The client is a desktop application implemented in HTML\JS and uses the Chromium web-browser. It streams audio recorded from a user's microphone to a web-service and receives recognized text. The communication is full-duplex; audio and text transmission is performed simultaneously. Audio streaming is performed by sending separate frames with a duration of about 250ms each; however, a much smaller frame size is used internally (for recording and speech analysis). This size is selected automatically by Chromium and can be as small as 10ms.

A regular expression based parser is used to find and execute dictation commands in a received text.

In order to improve recognition quality and to minimize the server load, the client implements simple voice activity detection (VAD), which is based on the method described in [17].

This VAD classifies audio signal in frames by using 3 features:

- Signal energy in a frame.
- Spectral flatness measure in a frame.
- Dominant frequency in a frame.

Each feature has a threshold that is adjusted automatically in real-time to adapt the speech detector to the specific microphone and environment. When at least 2 features have values beyond the respective threshold, the frame is marked as speech, and the client begins to send data to the web-service. For robustness, the client also sends the preceding 0.6 seconds (typically about 60 frames), which were not marked as speech. If no speech is found in the interval of 0.3 seconds (typically about 30 frames), data sending is paused until a speech frame is detected again.

The features and their thresholds are calculated as in the original paper; however, one simple modification is made by using information from ASR. Each time that a client sends audio to the server but no speech is found by ASR, a special message is sent to the client, which instructs VAD to adjust thresholds using the same procedure as described in the original paper by Moattar and Homayounpour[17].

4. Evaluation

In this section, the evaluation of the dictation system is presented. Two types of the evaluation were performed: (1) the evaluation of speech recognition quality and (2) the evaluation of speech recognition web-service optimizations.

4.1. Speech Recognition Quality

First, the evaluation of speech recognition quality is performed on a held-out set from LDSC. The results are presented in *Table 1*. Initially, the existing general transcription system (without LM adaptation and trained on the 100 hour long LSRC data set) was evaluated. The resulting high word error rate (WER) of 40.7% indicates that there is a significant difference between the dictation and transcription tasks.

After special processing (described in section 2.2) of the text corpus used in LM training, the result is significantly improved, and WER is reduced to 27.3%. Further improvement can be achieved by augmenting the training data with 8 hours from LDSC, achieving a WER of 23.9%, which is a 41.3% relative improvement.

Table 1. Speech recognition evaluation

ASR system	WER
Baseline with non-adapted LM (100 hours)	40.7
Baseline with adapted LM (100 hours)	27.3
Augmented (108 hours) with adapted LM	**23.9**

4.2. Performance and Memory Usage

Next, the effect of the rescoring optimizations is evaluated. The evaluation was performed on a server machine with 6 physical cores (12 virtual cores) and 128 GB of RAM. We run 1 to 16 workers. The shared rescoring program is configured to use 16 threads.

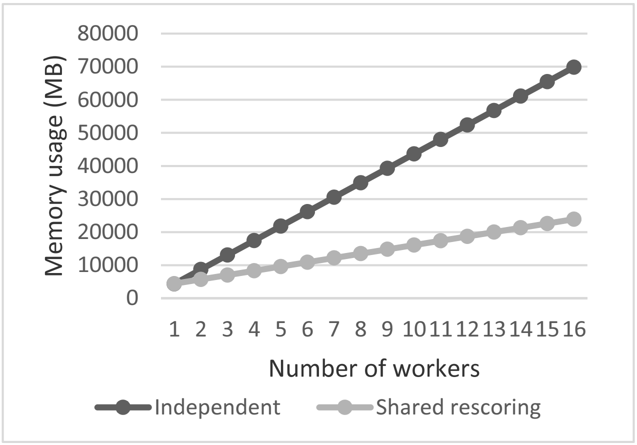

Figure 2. Idle memory usage.

Figure 2 shows the idle memory usage, i.e., how much memory is consumed by decoding the graph and language models alone. The advantage of "shared rescoring" is clearly seen. About 70 GB of memory is needed for storing models for 16 independent workers, while only about 24 GB are needed when using shared rescorer. Thus, a much simpler hardware can be used.

We also evaluate memory usage during decoding. *Figure 3* shows peak memory usage when simultaneously performing 1-16 decoding jobs of the same 1 minute long speech recording. The numbers reported represent the amount of memory used for lattices and decoding states only, excluding memory used for graph and models. The exact values are hard to analyze, because they are too dependent on operating system memory management specifics and the specific audio file. However, it shows that the shared rescoring method does not require additional memory during decoding.

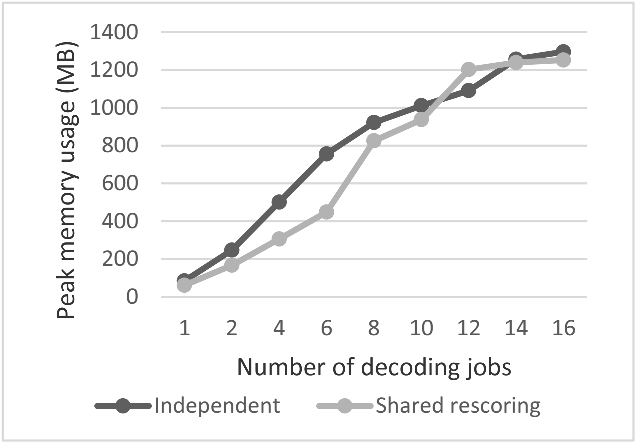

Figure 3. Peak memory usage during decoding.

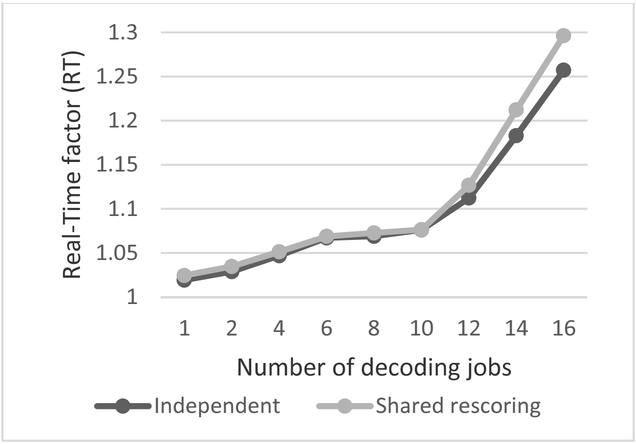

Figure 4. Real-time performance evaluation.

Finally, the effect on real-time performance is evaluated (see *Figure 4)*. The real-time factor (RT) is calculated by streaming 1-minute long speech recordings and measuring the time when last rescored hypothesis was received. For example, if we stream a 60 second speech recording and the last rescored part of transcription is received 1 second after the end of streaming, then we calculate RT as $61/60 \approx 1.02$. *Figure 4* shows that the difference between independent workers and workers with shared rescoring is very small and only starts to appear after 12 or more simultaneous decoding jobs are performed. This is most likely caused by the fact that the shared rescoring scheme uses a lot of multi-threading (for 12 decoding jobs it is: 12 worker threads + 1 master thread + 12 rescoring threads), and we have only 12 virtual cores.

It must be noted that we stream the same file for all workers. That means all lattice rescoring requests happen at the same time, which can be interpreted as worst-case scenario.

5. Conclusion

In the paper, we have presented the first dictation system for Latvian language. The system is in a beta stage and is yet to reach a level of accuracy where it is applicable for usage in the real world. The system achieved WER of 23.9% on a 1-hour held-out set from LDSC. In order to achieve this result, both acoustic and language models have been adapted. We also performed some optimizations in the back-end and front-end software. The front-end software implements voice activity detection in order to send data only when speech is detected. Optimizations in the backend allow to save 40 GB of RAM without significant performance degradation. The more efficient use of resources allows to use either a cheaper hardware or to run more workers on the same machine.

Acknowledgements

The research has been supported by the ICT Competence Centre project (ID 1.2.1.1/16/A/007), within the project “2.3. Speech recognition and synthesis for professional application”, of EU Structural funds.

References

[1] Oparin, I., Lamel, L., & Gauvain, J. (2013). Rapid Development of a Latvian Speech-to-text System. In ICASSP’13 (pp. 2–6). Vancouver, Canada.

[2] Lamel, L. (2012). Multilingual Speech Processing Activities in Quaero: Application to Multimedia Search in Unstructured Data. In *The Fifth International Conference: Human Language Technologies-The Baltic Perspective*.

[3] Pinnis, M., Auziņa, I., & Goba, K. (2014). Designing the Latvian Speech Recognition Corpus. In Proceedings of LREC 2014. Reykjavik, Iceland: European Language Resources Association (ELRA).

[4] Salimbajevs, A., & Pinnis, M. (2014). Towards Large Vocabulary Automatic Speech Recognition for Latvian. In *Human Language Technologies – The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT 2014* (Vol. 268, pp. 236–243).

[5] Salimbajevs, A., & Strigins, J. (2015). Using sub-word n-gram models for dealing with OOV in large vocabulary speech recognition for Latvian. In *Proceedings of NODALIDA 2015* (pp. 281–285).

[6] Vīra, I., & Vasiļjevs, A. (2014). The Development of Conversational Agent Based Interface. In Human Language Technologies - The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT 2014 (Vol. 268, p. 46).

[7] Salimbajevs, A., & Strigins, J. (2015). Latvian Speech-to-Text Transcription Service. In *Proceedings of Interspeech 2015* (pp. 722–723).

[8] Znotiņš, A., & Darģis, R. (2014) Baseline for Keyword Spotting in Latvian Broadcast Speech. In *Human Language Technologies - The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT 2014* (Vol. 268, pp. 75-82).

[9] Znotiņš, A., Polis, K., Darģis, R. (2015) Media Monitoring System for Latvian Radio and TV Broadcasts. In *Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association (Interspeech 2015)* (pp. 732-733).

[10] Miao, Y., Zhang, H., & Metze, F. (2014). Towards speaker adaptive training of deep neural network acoustic models. Proc. Interspeech 2014.

[11] Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. The Journal of the Acoustical Society of America, 87(4), 1738–52. doi:10.1121/1.399423

[12] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., … Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

[13] Pinnis, M., Salimbajevs, A., & Auzina, I. (2016). Designing a Speech Corpus for the Development and Evaluation of Dictation Systems in Latvian. In N. C. (Conference Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, … S. Piperidis (Eds.), Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). Paris, France: European Language Resources Association (ELRA).

[14] Goba, K., & Vasiljevs, A. (2007). Development of Text-To-Speech System for Latvian. In Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007 (pp. 67–72).

[15] Kneser, R., & Ney, H. (1995). Improved backing-off for M-gram language modeling. 1995 International Conference on Acoustics, Speech, and Signal Processing, 1, 181–184. doi:10.1109/ICASSP.1995.479394

[16] Alumäe, T. (2014). Full-duplex Speech-to-text System for Estonian. In Human Language Technologies - The Baltic Perspective - Proceedings of the Sixth International Conference Baltic 2014, Kaunas, Lithuania, September 26-27, 2014 (pp. 3–10). doi:10.3233/978-1-61499-442-8-3

[17] Moattar, M., & Homayounpour, M. (2009). A simple but efficient real-time voice activity detection algorithm. In European Signal Processing Conference (EUSIPCO) (pp. 2549–2553). doi:10.1007/978-1-4419-1754-6