UNIVERSITY OF LATVIA
FACULTY OF COMPUTING

Askars Salimbajevs

MODELLING LATVIAN LANGUAGE FOR AUTOMATIC SPEECH RECOGNITION

Doctoral Thesis
Submitted for the Ph.D. (Dr. Sc. Comp.) academic degree

Scientific supervisor:
Dr.sc.comp. **Inguna Skadiņa**

RIGA 2019

ABSTRACT

In recent years, the success of speech technologies like speech recognition and speech synthesis for languages like English has prompted a new excitement about spoken interfaces and an interest in further research of these technologies. However, most of the research and development are concentrated around "big" languages and smaller languages like Latvian are not covered. For example, no speech recognition technology for Latvian was publicly released until 2015.

The aim of this doctoral thesis is to research methods and models for automatic speech recognition for Latvian language and develop practical tools and systems. Both theoretical and practical aspects are covered, including a research on acoustic and language models, system adaptation for specific tasks, automatic data collection and augmentation, inverse text normalization (punctuation restoration) and practical system development.

A practical application of the research resulted in a public online Speech-To-Text service for Latvian and an integration of audio transcription and dictation functionality into a desktop and mobile applications (e.g. Tildes Birojs software package and Tildes Balss). In 2015 these were the first services of this kind for the Latvian language.

The developed ASR system is evaluated on different evaluation corpora, which emulate different usage domains. On a general domain evaluation set the developed ASR achieves a word error rate of 10.1% and significantly outperforms Google (error rate of 36.2-50.6%) and Speechmatics (error rate of 25.2%) ASR solutions for Latvian that recently became available.

Keywords: automatic speech recognition, acoustic modelling, language modelling

ACKNOWLEDGEMENTS

Unreserved thanks are due to Dr.sc.comp. Inguna Skadiņa, for guiding me through the research process and providing insight, encouragement, feedback for the thesis and the involved publications.

Special thanks have to be expressed to the Tilde company for giving me the possibility to participate in the various research projects and for being a great place for me to work and develop myself as a researcher. My gratitude to several current and past comrades-in-work for giving advice from their professional experience, participating in thought provoking discussions and development process of the methods described in the thesis. In particular, Jevgenijs Strigins, Mārcis Pinnis, Indra Ikauniece and Andris Varavs, who have been great colleagues in my research and are co-authors of my scientific papers.

The thesis is based on the research and development done in several research projects at Tilde that has been supported by EU Structural funds. Without this support, the goals of the thesis would be very difficult (if not impossible) to achieve.

Also, thanks to all the anonymous reviewers of the publications for their constructive criticism, which has been very useful for further studies and contributed to the progress of the work of the thesis.

On a personal note, I would like to warmly thank parents, brother and unnamed friends, thank you all for your patience and support.

This dissertation is dedicated to the memory of my grandfather Alexander Sobolevsky, who has been my childhood inspiration. To a large extent, thanks to him, I am the person I am today. Together with my parents, he laid the fundamentals of my personality and lust for knowledge, that is the basis of all my further life's journey and this work in particular.

CONTENTS

1. INTRODUCTION

1.1. Research Area

The research area in the focus of this thesis is the Automatic Speech Recognition (ASR). ASR is the inter-disciplinary sub-field of computational linguistics and natural language processing (NLP) that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is also known as "speech recognition" (SR), "computer speech recognition", or just "speech to text" (STT). The speech recognition field deals with the following problems:

- how to convert analog audio signal to digital form and extract speech relevant information;
- how to model acoustic properties of the phonemes and how to distinguish among phonemes;
- how to integrate the linguistic and world knowledge into a recognition process;
- how to obtain the data for statistical model training, how to process this data and how to train the models;
- how to deal with variability in environments, voices, accents, recording equipment, syntax and semantics;
- how to make recognition process efficient, as the search space of all possible sentences is enormous;
- how to adapt ASR for specific tasks and domains;
- etc.

It is a diverse field that relies on knowledge of language at the levels of signal processing, acoustics, phonology, phonetics, syntax, semantics, pragmatics, and discourse. The foundations of spoken language processing lie in computer science, electrical engineering, linguistics, and psychology.

From the technology perspective, speech recognition has a long history with several waves of major innovations. First attempts on speech recognition can be traced back to 1950s. In 1952 three Bell Labs researchers built a system for single-speaker digit recognition (Davis et al, 1952). The 1950s era technology was limited to single-speaker systems with vocabularies of around ten words.

Speech recognition research in the 1980's was characterized by a shift in methodology from the more intuitive template-based approach (a straightforward pattern recognition paradigm) towards a more rigorous statistical modelling framework. Although the basic idea of the hidden Markov model (HMM) was known and understood early on in a few laboratories, e.g., IBM (Jelinek et al, 1975) and

the Institute for Defense Analyses (IDA) (Ferguson, 1980), it wasn't until mid- 1980's that the hidden Markov model became the preferred method for speech recognition. The popularity and use of the HMM as the main foundation for automatic speech recognition and understanding systems has remained constant over the past two decades, especially because of the steady stream of improvements and refinements of the technology.

In the beginning of 2010s a new technology wave emerged. Using hybrid HMM and deep neural networks (DNN) allowed to greatly improve speech recognition accuracy by 30%-50% (Dahl et al., 2012; Hinton et al., 2012).

Today, however, many aspects of speech recognition have been taken over by a trend of moving away from HMM's and adopting pure neural network approach, using a deep learning methods like Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), Connectionist Temporal Classification (CTC) (Graves et al., 2006), and etc. As of 2017, neural network based CTC-trained (or using similar approaches) ASR systems have outperformed the hybrid HMM-DNN in almost all areas and dominate the field in terms of accuracy of recognition.

Where does the state-of-the-art speech recognition system stand today? The answer very much depends on the specific task and language. For a "big" and widely used languages like English, automatic speech recognition is almost approaching human parity in the cases where audio recordings are clean and vocabulary is limited. There have also been claims that ASR achieve human parity on a test set of conversional speech with telephone quality (Xiong et al., 2016). However, when conditions are not ideal, we can see that humans are far more robust, and there is still a large gap. Noise, persons speaking simultaneously in the background, reverberations, far-field recognition and other adverse conditions still pose a great challenge to an ASR system.

For smaller languages, the situation can be completely different. Modelling languages for ASR requires sufficiently large annotated speech corpora, monolingual text corpora, natural language processing tools, computing power to train statistical models etc. For "small" under-resourced languages ASR solutions, as well as language technologies in general, are not as well developed due to the lack of linguistic resources and technological approaches that enable ASR solutions to be developed cost effectively. This has resulted in a technological gap between these two groups of languages, because of the lack of research and resources even the simplest scenarios of ASR can be impossible or pose a lot of trouble for smaller languages.

1.2. Relevance of the Research Problem

Most of the research in speech recognition (as well as ready-made tools and language resources) is usually focussed on "big" most spoken languages, like English, Mandarin, French, German,

Spanish etc. (Rehm & Uszkoreit, 2012). For languages with small number of speakers, the situation can be completely different. For such languages, ASR solutions, as well as language technologies in general, are not as well developed due to the lack of linguistic resources and technological approaches that enable ASR solutions to be developed cost effectively. This has resulted in a technological gap between these two groups of languages. For example, the amount of research and effort on speech recognition for languages of Baltic states (i.e. Estonian, Latvian and Lithuanian) is incomparable to work performed for widely-spoken languages. However, there is a strong interest in research that allows to make this technology available for such languages.

In recent years, the success of spoken interfaces in smartphones and tablets has prompted new excitement about the speech technologies. This success has stimulated many developers to embrace speech technologies for their native languages. Automatic speech recognition can have many useful applications, e.g.:

- Speech is a natural communication interface for humans, the ability to control PC, smartphone, car etc. by voice can be convenient and efficient in many cases. For example, you can select destination and start navigation in a car without taking your hands off the steering wheel and going through many tangled menus.
- For many people dictating text is much faster than typing, older or not experienced users can really benefit from ability to enter text just by speaking.
- Speech recognition is very useful for people with disabilities, they can use speech recognition as an input device for their personal computer. One could also use speech recognition as a type of hearing aid.
- Speech recognition can be used to transcribe audio recordings, for example, meetings, court hearings, dictaphone recordings, lectures, interviews etc. The transcription then can be used for keyword search or, after some processing, as a usual text.

Such applications were not possible for Latvian as there were no automatic speech recognition solutions. This necessity for speech recognition technology for Latvian language has been the driving force of this work. Moreover, before starting the research, the author conducted a small-scale literature review about “speech recognition research for languages of Baltic States, Estonian, Lithuanian and Latvian”. This small-scale literature review showed that there is a clear knowledge gap and Latvian language is the least researched among languages of Baltic States. Therefore, this work addresses an unsolved problem of modelling Latvian language for ASR.

1.3. Aim and Objectives

The aim of the research is to find efficient algorithms and create optimal models and systems for Latvian language speech recognition. Therefore, the research seeks to answer the following questions:

1. What data is needed to train statistical models for speech recognition for Latvian?
2. How to get this data and how to process it before training?
3. What modelling methods and algorithms are the most appropriate for Latvian?
4. What are the best model parameters?
5. How to design speech recognition systems and adapt them to specific applications?
6. How to process the raw output of speech recognition systems in order to adapt it for some specific application?

It is also important to consider the complexity of Latvian language:

- Latvian language is inflective; each word can have many different surface forms with different endings. Word endings must agree with other words in sentence.
- Latvian language has a rich word-formation options. New words can be formed by adding suffixes, prefixes or by making compounds.
- Word order in sentence is relatively free.

These features of Latvian language create additional difficulties for speech recognition and finding solutions for these problems is important part of this research.

At the end of this study the author expects that all research questions will be answered, and research hypotheses will be proven. The ability to answer selected research questions will:

- Facilitate data collection and processing for Latvian speech recognition.
- Provide guidelines for creating general and application-specific systems for Latvian speech recognition.
- Provide a baseline for future Latvian speech recognition research.
- Possibly help speech recognition research for languages similar to Latvian (e.g. Lithuanian).

It was also anticipated that after the end of the research a speech recognition system for Latvian will be developed and integrated into real business applications.

1.4. Research Hypotheses

The following research hypotheses have been proposed:

1. It is possible to develop statistical models for large vocabulary general-purpose automatic speech recognition for Latvian.
2. It is possible to achieve high accuracy of recognition with standard or modified versions of methods, developed for "big" languages.
3. Developed general-purpose Latvian speech recognition models have reasonable computational requirements, therefore can be used in practical applications and deployed as publicly available services.
4. Developed general-purpose Latvian speech recognition models can be adapted for specific domains and tasks.

Research hypotheses in the thesis are proved using experimental methods.

1.5. Research Methods

The following main contemporary research methods are used by the author:

- **Scientific literature review** – author analysed different literature (books, journals, arxiv preprints and publications in the natural language processing) in order to identify baseline and state-of-art methods speech recognition and also to prove the existence of knowledge gap – a lack of research on speech recognition for Latvian language.
- **Iterative development** – the tools and models developed in this work have been designed, implemented and deployed in an iterative manner, improvements were proposed and implemented after reviewing results of previous iteration.
- **Controlled experiments** – suitability and effectiveness of algorithms and trained statistical models were evaluated in controlled experiments (Wohlin et al., 2003).
- **Automatic quantitative evaluation** – the evaluation methods used in the scope of thesis are fully automatic and provide quantitative results that enables objective comparison of various approaches and models.
- **Error analysis** – where necessary, the author has performed manual error classification in order to identify possible error causes and directions for future improvements.

1.6. Main Results

The main contributions of this thesis are as follows:

- Analysis of different acoustic and pronunciation modelling methods for Latvian language, that shows that the best results can be achieved using pure grapheme-based approach, training data augmentation and time-delay neural network models.
- Implementation of method for extracting acoustic model training data from inaccurately annotated speech recordings from Web. This method allows to increase the amount of the training data fully automatically without any additional manual transcription. Increasing the amount of training data enables of more advanced and more accurate acoustic models.
- A corpus of 186 hours of annotated speech data from Saeima session recordings from 2011-2014, which was created using above mentioned method.
- Solutions for monolingual text data filtering methods, which allow to filter noise and garbage from a fully automatically crawled text corpus and train accurate Latvian language models for ASR.
- Implementation and approbation of large vocabulary general-purpose Latvian automatic speech recognition system, which achieves word error rate of 10.1% on general domain test set and outperforms the Latvian ASR by Google.
- Adaptation of general-purpose speech recognition models to various specific tasks like dictation, address input and Saeima session transcription and approbation of adapted systems. The adapted systems achieve word error rate of 12.6% on dictation, 7.9% on street address input and 5.9% on Saeima transcription.
- Implementation of first punctuation restoration statistical model for Latvian language. This model is used to restore punctuation in ASR transcripts and significantly improves the readability of transcripts.

1.7. Approbation and Publication of the Author's Work

The Latvian automatic speech recognition system developed in this research, is published as a Latvian Speech-To-Text transcription service (Salimbajevs & Strigins, 2015a). The web-service provides ASR for several applications:

- A free online file transcription service http://www.tilde.lv/balss, which provides audio and video file transcription.
- A software package Tildes Birojs, which provides audio and video file transcription, as well

as dictation functionality.

- A mobile app Reizrēķins (available in Google Play store), which teaches kids multiplication.
- A mobile app Tildes Balss (available in Google Play store), which enables any Android OS user to speak to their smartphone in Latvian language.
- A number of commercial applications developed for Tilde clients (names cannot be given according to non-disclosure agreements).

The results of this research were presented at the following international conferences:

- The 8th Conference on Human Language Technologies - The Baltic Perspective (Baltic HLT 2018), Tartu, Estonia, 2018;
- 11th International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 7-12, 2018;
- 18th Annual Conference of the International Speech Communication Association (INTERSPEECH 2017), Stockholm, Sweden, 2017;
- The 7th Conference on Human Language Technologies - The Baltic Perspective (Baltic HLT 2016), Riga, Latvia, 2016;
- The 10th Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia, 2016;
- 16th Annual Conference of the International Speech Communication Association (INTERSPEECH 2015), Dresden, Germany, 2015;
- The 20th Nordic Conference of Computational Linguistics (NODALIDA 2015), Vilnius, Lithuania, 2015;
- Recent Advances in Natural Language Processing (RANLP 2015), Hissar, Bulgaria, 2015;
- The 6th Conference on Human Language Technologies - The Baltic Perspective (Baltic HLT 2014), Kaunas, Lithuania, 2014.

Research results are reported in the 10 papers published in the proceedings of the international conferences:

- Salimbajevs, A., & Kapočiūtė-Dzikienė, J. (2018). General-purpose Lithuanian Automatic Speech Recognition System. In Human Language Technologies - The Baltic Perspective - Proceedings of the Seventh International Conference Baltic HLT 2018, Tartu, Estonia, September 27-29, 2018.

- Salimbajevs, A. (2018). Creating Lithuanian and Latvian Speech Corpora from Inaccurately Annotated Web Data. In LREC 2018, 11th International Conference on Language Resources and Evaluation, Miyazaki, Japan, May 7-12, 2018.
- Salimbajevs, A., & Ikauniece, I. (2017). System for speech transcription and post-editing in Microsoft Word. In INTERSPEECH 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017 (pp. 825–826).
- 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).
- Salimbajevs, A. (2016a). Bidirectional LSTM for Automatic Punctuation Restoration. In I. Skadina & R. Rozis (Eds.), Human Language Technologies - The Baltic Perspective - Proceedings of the Seventh International Conference Baltic HLT 2016, Riga, Latvia, October 6-7, 2016 (Vol. 289, pp. 59–65). IOS Press. http://doi.org/10.3233/978-1-61499-701-6-59
- Salimbajevs, A. (2016b). Towards the First Dictation System for Latvian Language. In I. Skadina & R. Rozis (Eds.), Human Language Technologies - The Baltic Perspective - Proceedings of the Seventh International Conference Baltic HLT 2016, Riga, Latvia, October 6-7, 2016 (Vol. 289, pp. 66–73). IOS Press. http://doi.org/10.3233/978-1-61499-701-6-66
- Salimbajevs, A., & Strigins, J. (2015a). Latvian Speech-to-Text Transcription Service. In INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015 (pp. 723-725)
- Salimbajevs, A., & Strigins, J. (2015b). Error Analysis and Improving Speech Recognition for Latvian Language. In Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria (pp. 563–569).
- Salimbajevs, A., & Strigins, J. (2015c). Using sub-word n-gram models for dealing with OOV in large vocabulary speech recognition for Latvian. In B. Megyesi (Ed.), Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Institute of the Lithuanian Language, Vilnius, Lithuania (pp. 281–285). Linköping University Electronic Press / ACL.
- Salimbajevs, A., & Pinnis, M. (2014). Towards Large Vocabulary Automatic Speech Recognition for Latvian. In Human Language Technologies – The Baltic Perspective (pp.

236–243). IOS Press.

1.8. Outline

The remainder of this document is structured as follows:

- Section 2 is devoted to theoretical background of speech recognition, description of basic ASR components, advantages and disadvantages of different approaches. It also contains review of related research on speech recognition for Latvian and other Baltic languages.
- Section 3 describes the acoustic modelling for automatic speech recognition for Latvian language, presents experiments on different acoustic models, grapheme-to-phoneme modelling, training data augmentation and collecting new training data from the Web. The section is based on the publications Salimbajevs & Pinnis (2014), Salimbajevs & Strigins (2015b) and Salimbajevs (2018).
- Section 4 presents language models for Latvian ASR, experiments on training data filtering and sub-word recognition. The section is based on the publications Salimbajevs & Pinnis (2014), Salimbajevs & Strigins (2015b) and Salimbajevs & Strigins (2015c).
- Section 5 focusses on adaptation of general domain ASR system to specific tasks: dictation, Saeima session transcription, address recognition and punctuation restoration. In the of this section the Latvian Speech-To-Text transcription web-service is presented, which is the first publicly available speech recognition service for Latvian. The section is based on the publications Salimbajevs (2016a), Salimbajevs (2016b), Pinnis et al. (2016) and Salimbajevs & Ikauniece (2017).
- Section 6 summarizes results of the research and gives conclusions about this work.

2. CONCEPTS OF SPEECH RECOGNITION

Spoken language is used to communicate information from a speaker to a listener. Speech production and perception are both important components of the speech chain. Spoken interaction can be divided into several distinct elements (see **Figure 1**). Speech begins with a thought, some semantic message, and intent to communicate in the brain. The computer counterpart to the process of message formulation is the application semantics that creates the concept to be expressed. After the message is created, the next step is to encode the message into a sequence of words. This is done by some language system. Words are combined into utterances, a continuous piece of speech generally beginning and ending with a voiced or un-voiced pause. Each word consists of a sequence of phonemes that corresponds to the pronunciation of the words. Each utterance also contains a prosodic pattern that denotes the duration of each phoneme, intonation of the utterance. Once the language system finishes the mapping from semantic concepts to utterances, a series of neuromuscular signals activate muscular movements to produce speech sounds. The speaker continuously monitors and controls the vocal organs using his or her own speech as feedback.

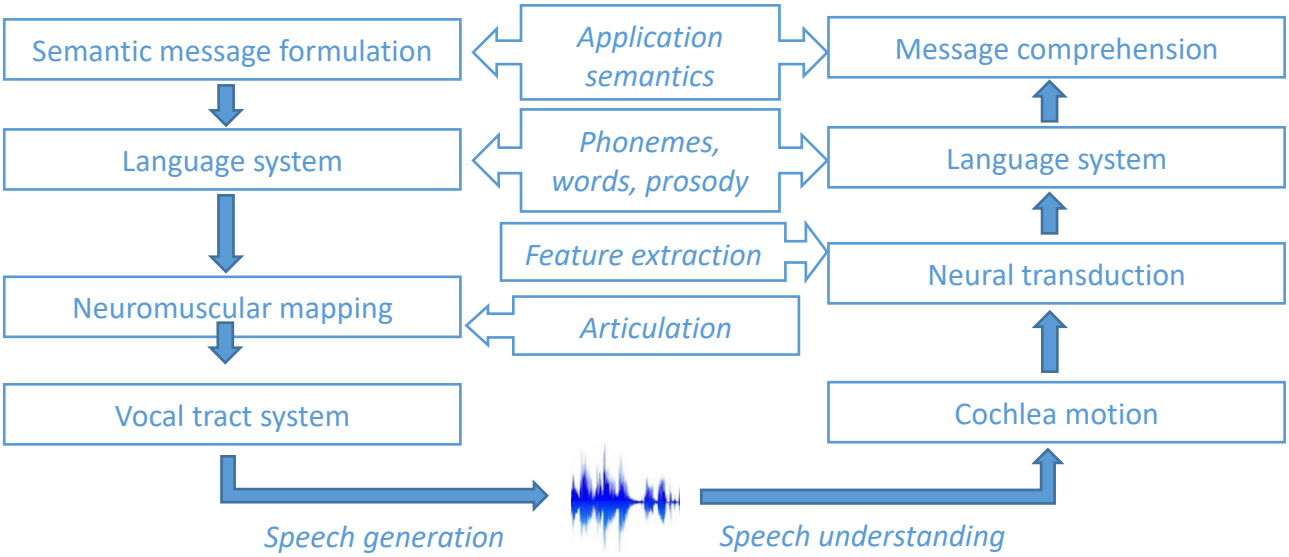

Figure 1. *Speech generation and understanding.*

The speech understanding process works in reverse order. First the signal is passed to the cochlea in the inner ear, which performs frequency analysis. A neural transduction process follows and converts the spectral signal into activity signals on the auditory nerve, which are then processed by the brain and the message is extracted. Currently, it is unclear how neural activity is mapped into the language system and how message comprehension is achieved in the brain.

From computer programming point of view the task of speech recognition is to convert speech into a sequence of words by a computer program. The long-term goal of speech recognition is to enable people to communicate more naturally and effectively, as speech is the most natural communication modality for humans. Achieving this ultimate objective requires deep integration with

many NLP components, which would enable computer program to understand the semantic messages encoded in the speech and execute the requested task or simply engage in a dialog. However, in this work we will only focus on a core speech-recognition task – conversion of speech into a sequence of words.

2.1. Nature of Sound

Sound is a longitudinal pressure wave formed of compressions and rarefactions of transmission medium (typically air) molecules, in a direction parallel to that of the application of energy. Compressions are zones where air molecules have been forced by the application of energy into a tighter configuration, and rarefactions are zones where air molecules are less tightly packed. Although there are many complexities relating to the transmission of sounds, at the point of reception (i.e. the ears or microphone), sound is readily dividable into two simple elements: pressure and time. These fundamental elements form the basis of all sound waves. They can be used to describe, in absolute terms, every sound we hear.

The alternating nature of compression and rarefaction of air molecules along the path of an energy source can be described by the graph of a sine wave as shown in **Figure 2**. In this representation, crests of the sine curve correspond to moments of maximal compression and troughs to moments of maximal rarefaction. However, real life sound waves are a lot more complex and are a combination of various sound wave frequencies (and noise).

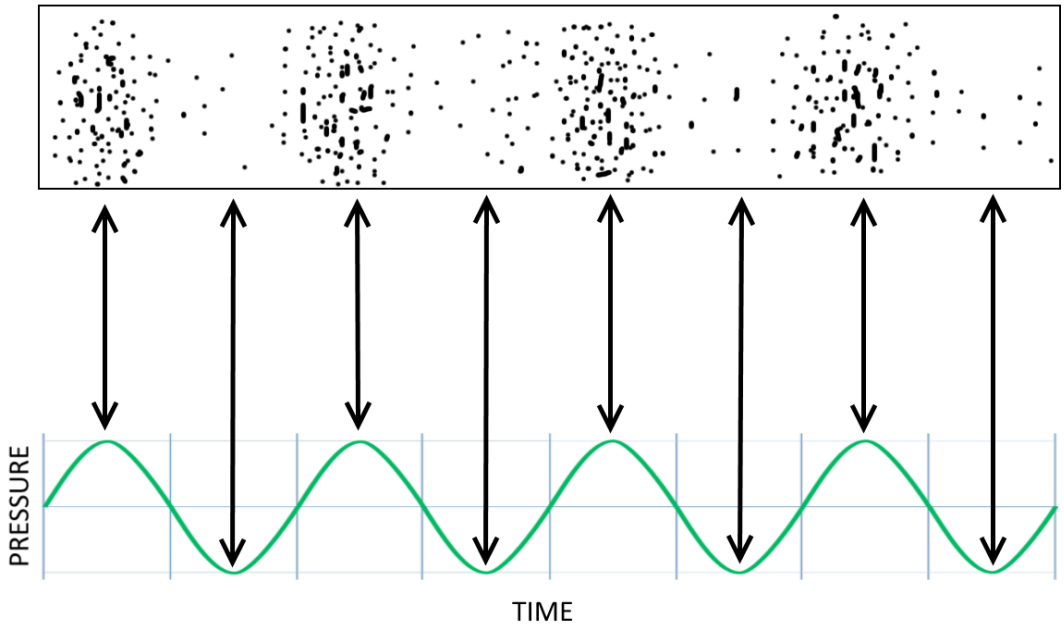

Figure 2. *Sound as series of compressions of air molecules and as a pressure wave.*

This complexity can be reduced by decomposing a sound wave into sinusoidal plane waves, which are characterized by these generic properties:

- Frequency, or its inverse, the Wavelength

- Amplitude
- Sound pressure / Intensity
- Speed of sound
- Direction

At the point of reception, the air pressure variations of a sound wave cause vibration of diaphragm in human ear or sound sensing material in a microphone. The most common type of microphone is the dynamic microphone, which uses a coil of wire suspended in a magnetic field; the condenser microphone, which uses the vibrating diaphragm as a capacitor plate, and the piezoelectric microphone, which uses a crystal of piezoelectric material.

2.2. Speech

Speech is the vocalized form of communication produced by air-pressure waves emanating from the mouth and the nostrils of a speaker. Speech is based upon the syntactic combination of lexicals and names that are drawn from a very large vocabulary. Each spoken word is created out of the phonetic combination of a limited set of speech sound units, called *phonemes*. These vocabularies, the syntax which structures them, and their sets of speech sound units differ, creating many thousands of different, and mutually unintelligible, human languages.

Because every speaker has a unique vocal anatomy, vocalizations of speech sounds are also unique, so perception is based on commonality of form. For example, the **Figure 3** shows spectrograms of the same phoneme produced by two different speakers.

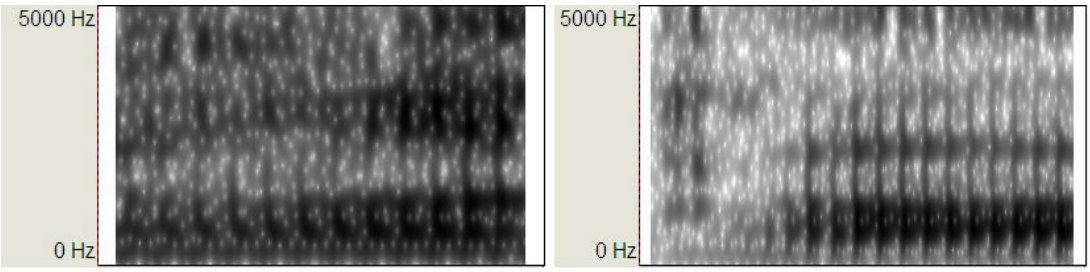

Figure 3. *Spectrograms of the same phoneme pronounced by two different speakers.*

The dark horizontal bands are called *formants***.** Formants are distinctive frequency components of the acoustic signal. Each phoneme has a characteristic combination of formants and their trajectories. In theory, that means that the information required to distinguish between phonemes, can be represented purely quantitatively by specifying peaks in the amplitude/frequency spectrum. However, in real life, as it can be seen on **Figure 3**, the acoustic representation of the same phoneme can be very different, there is a great variability because of different environment, unique speaker voice characteristics etc. This makes speech recognition a very non-trivial task, which requires

complex solutions to address this variability.

2.3. *Speech Recognition Problem*

A source-channel mathematical model is often used to formulate speech recognition problems. As shown on **Figure 4**, the communication starts when speaker decides the source word sequence T and produces it through his/her text generator. The source sequence is passed through a *noisy communication channel,* the speech generator (speaker's vocal apparatus) produces the speech waveform, the acoustic signal X, that is then processed by a signal processing component of the speech recognizer. This channel is called noisy, because some information can be lost during conversion of source into waveform, during transmission of the audio signal and in the signal processing component. At the other end of communication channel is the speech decoder, that decodes the acoustic signal X into a word sequence T', which we hope is close to the original sequence T.

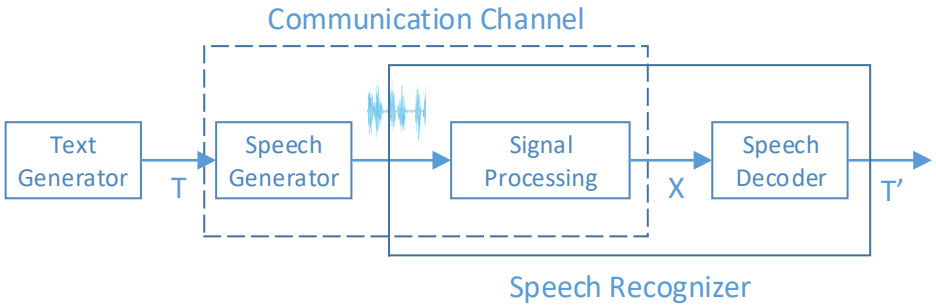

Figure 4. *Speech recognition as noisy communication channel problem.*

Typical speech recognition system can be divided into basic components shown in the **Figure 5**. Voice data from a microphone or other source is first received by a signal processing component, that typically perform pre-emphasis, automatic gain control and feature extraction. The extracted feature vectors are passed to the decoder, which produces the recognition results. Applications interface with the decoder to get recognition results. The decoding of the speech signal requires at least three models to generate the word sequence that has the maximum posterior probability for the extracted input feature vectors.

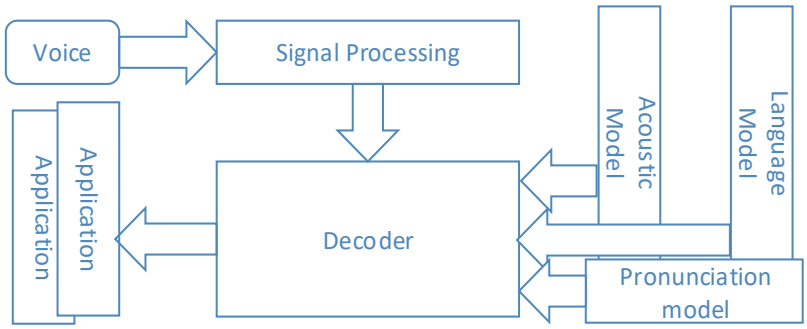

Figure 5. *Decomposition of speech recognition into major components.*

Acoustic model contains the knowledge about acoustics, phonetics, microphone and

environment variability, gender and dialect differences among speakers, etc. Pronunciation or grapheme-to-phoneme model refer to knowledge of which phonemes sequences constitute possible words. Pronunciation model often is viewed as a part of acoustic or language model. Language model represents a system's knowledge of word semantics, which words are likely to co-occur, and in what sequence. Knowledge of operations a user may wish to perform may also be included in the language model.

Another way to look at the speech recognition problem, is to formulate it mathematically as follows:

$$T' = arg \max_{\widehat{W}} P(W|X)$$

where X is the acoustic signal and T' is decoded word sequence.

In other words, we are trying to find word sequence that has the maximal conditional probability given acoustic signal X. For this we need to train a statistic model $P(W|X)$. Unfortunately, it's practically impossible to estimate this probability directly, for example, one would need to collect training set of acoustic signal examples with multiple different transcripts for each audio recording, but there are no naturally occurring data of this kind.

This problem can be solved by evaluating this probability indirectly by using Bayes rule, thus the problem is rewritten as follows:

$$T' = arg \max_{\widehat{W}} \frac{P(X|W)P(W)}{P(X)}$$

where $P(X|W)$ is the conditional probability of acoustic signal X given word sequence W, or acoustic model, $P(W)$ is unconditional probability of word sequence W, or language model, and $P(X)$ is unconditional probability of acoustic signal X.

Because optimal value of $\widehat{W}$ is independent of $P(X)$, this probability can be ignored in the optimization process and the decoder works only with non-normalized probability. However, some estimate of this probability might be necessary if one would want to calculate normalized probability of T' given acoustic signal X. For example, for calculating the confidence of the decoder for given recognized word sequence.

The language model can be estimated from large text corpora. But acoustic model can be naively estimated from the large collection of audio recordings of the same word sequences. In practice, however, decoder recognizes not the word sequences, but phoneme sequences, because they have much less variability (word count is not limited, but there are only dozens of phonemes). This makes it much easier to prepare the training data and to perform the recognition. The mapping

between words and phonemes are performed by so called pronunciation or grapheme-to-phoneme model, which can be separate or a part of acoustic or language model.

Word error rate (WER) is a common metric of the performance of a speech recognition system. The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level. The WER is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors and further work is therefore required to identify the main source(s) of error and to focus any research effort.

The WER is calculated by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment and then computing:

$$WER = \frac{S + D + I}{N}$$

where

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- N is the number of words in the reference (N=S+D+C).

Many questions arise when training acoustic and language models and building practical speech recognizers, associated with speaker vocal tract differences, speech style and rate, likely words and phrases, unknown words, grammatical variation, noise interference, nonnative accents, getting and processing training data, determining best modelling methods and parameters, understanding how to design speech recognition systems and adapt them to specific applications, and confidence scoring of results. A successful speech recognition system must address most if not all of these uncertainties.

In many cases the solutions to these questions are applicable across different speech recognition systems and different languages. So, for example, methods successfully applied for close-talk prepared speech, can be to some extent reused for close-talk spontaneous speech. Or methods successfully adopted for English speech recognition, can be transferred to Latvian. However, when developing speech recognition for a new language (i.e. a language for which there is no (or very little) previous research), there will be always new questions, that needs to be addressed. For Latvian language, these questions include:

1. What data is needed to train statistical models for speech recognition for Latvian?
2. How to get this data and how to process it before training?

3. What modelling methods and algorithms are the most appropriate for Latvian?
4. What are the best model parameters?
5. How to design speech recognition systems and adapt them to specific applications?

2.4. Feature Extraction

Acoustic signal, including human voice, is difficult to process and recognize directly, as it contains a lot of noise and much unnecessary information which may trouble the speech recognition. Also, a speech signal is a time-series signal, but the information that is needed to recognize speech is contained in the frequency-domain (as formants). That means, that we need to use some feature extraction method (that performs a Fourier or similar transform, among other things) or use a model that can learn such transformations.

The second approach requires a very complex model engineering and huge amount of training data. So, historically, the first approach is much widely adopted.

During feature extraction, a discretized acoustic wave is converted into a series of feature vectors, which capture the spectral information necessary to make recognition, remove unnecessary information and reduce noise. These feature vectors can then be passed to a statistical model.

Two of the most popular feature vector types are MFCC (Mel-frequency Cepstral Coefficients) (Mermelstein, 1976) and PLP (Perceptual Linear Prediction) (Hermansky, 1990). There are also many other features types, however they all have much in common. In this subsection, a more detailed description MFCC feature extraction will be given as an example.

The MFCC feature vectors are typically extracted by performing the following steps:

1. Audio signal is divided into a small segments (frames), a special windows function is applied for each frame.
2. Each frame is projected into frequency domain by Fourier transform.
3. Power spectrum is calculated from Fourier transform for each frame.
4. Mel filters are applied to obtained spectrum.
5. Absolute filter values are replaced with logarithms and discrete cosine transform is applied, as if it were a signal. There result is called *cepstrum*.
6. MFCC vectors are formed from the amplitudes of the resulting spectrum.

In the first step, audio signal is typically divided into 20-30ms frames which overlap by 10ms. Each frame is transformed into a N-dimensional feature vector. This setup is considered to contain all the necessary information for making decisions on which phoneme is contained in the frame (Atal,

1976).

Next, in order to calculate correct Fourier transform of each frame, we need to apply a window function. In MFCC and other features typically Hamming window function is used.

A Fourier transform then projects the data from time domain to the frequency domain. From this transform a power spectrum is computed, i.e. we calculate the energy of the signal per unit time in the each of Fourier transform frequency bins in the current frame.

At this step, one could reduce number of frequency bins to some N and use them as N-dimensional feature vector. This type of feature vectors is called – "filter-bank features" (because, so called filter-bank is applied to the frequency bins of the frame).

MFCC idea is to position the filters so that most important information is captured with most precision. For this so-called Mel-scale (Stevens & Volkmann, 1937) is used, which is a perceptual scale of pitches judged by listeners to be equal in distance from one another. Or in other words, it describes how well human ear can distinguish between frequencies. Using this scale, we align filters in such way that frequency area that is well recognized by human ear has higher density of filters.

Figure 6 shows few examples of Mel-filters. Triangular filters are typical used, which have a value of 1 for the central frequency and linearly decline to zeros in the boundaries. Mel-filters are applied to the power spectrum.

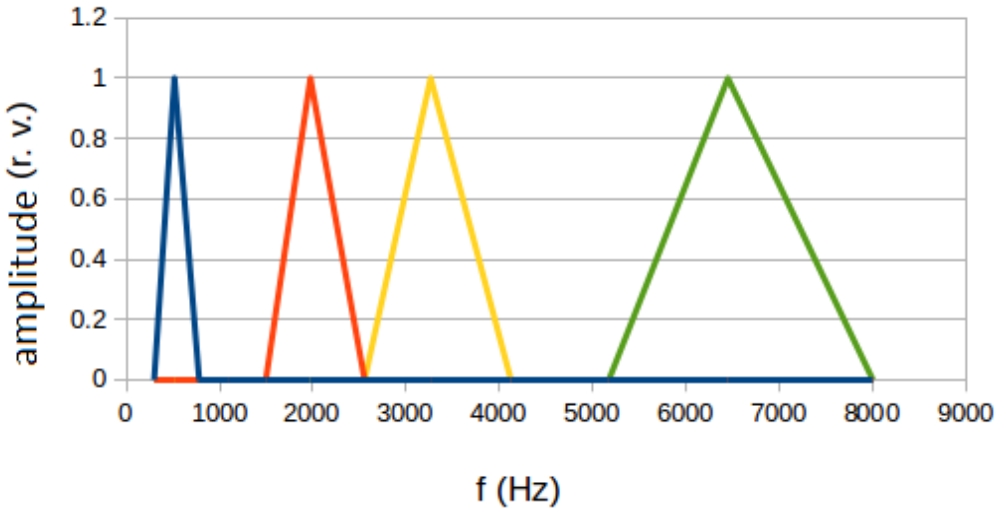

Figure 6. *Mel filters 1,5,7 and 10 in the frequency interval 300-8000Hz.*

In the final step, a discrete cosine transform (DCT) is performed on logs of filter values and first N values a taken as MFCC feature vector (N is usually 13). Sometimes, DCT can be skipped (Fook et al., 2013).

The MFCC feature vector describes only the power spectral envelope of a single frame, but as it was mentioned in Section 2.2 speech also have information in the dynamics i.e. what are the

trajectories of the MFCC coefficients over time. It turns out that calculating the MFCC trajectories and appending them to the original feature vector increases ASR performance (if we have 12 MFCC coefficients, we would also get 12 delta coefficients, which would combine to give a feature vector of length 24).

To calculate the delta coefficients (differential coefficients), the following formula is used:

$$d_t = \frac{2\sum_{n=1}^{N} n(c_{t+N} - c_{t-N})}{2\sum_{n=1}^{N} n^2}$$

where d_t is a delta coefficient from frame t computed in terms of the static coefficients c_{t+N} to c_{t-N}. A typical value for N is 2. Delta-Delta (Acceleration) coefficients are calculated in the same way, but they are calculated from the deltas, not the static coefficients.

To further improve the classification power of features, a frame splicing is typically performed. For example, we concatenate nine consecutive frames together, 4 previous frames as left-context, central frame and 4 next frames as right-context. This helps the recognition, but creates another problem – dimensionality of the input is increased. Many of the new dimensions are redundant and correlated. So, in order to reduce the dimensionality and decorrelate input data usually a Linear Discriminant Analysis (LDA) transform is applied (Yu et al., 1990; Haeb-Umbach &Ney, 1992). LDA features are then fed into a statistical model for speech recognition.

2.5. Acoustic Model

From section 2.3 we know that the acoustic model is a conditional probability of some acoustic signal X given the word sequence W and is typically written as $P(X|W)$. In simple words, acoustic model tells us what is the probability that word sequence W sounds like X. When performing speech recognition, acoustic signal X is given as an input and the goal of the decoder is to find the word sequence W that maximizes this probability.

To learn how to estimate these probabilities we train a statistical model on a large set of examples, called *speech corpus*. Each example is a word sequence and a corresponding audio recording. The more diverse and bigger set of examples we collect, there more reliable model can be obtained.

Let's suppose we are developing a speech recognizer, that can only recognize word sequences W_1 and W_2. For this we ask 10 speakers to say both W_1 and W_2 each 20 times. From these recordings, we can get the necessary statistics to estimate $P(X|W_1)$ and $P(X|W_2)$. However, it is easy to see that this approach has several problems:

- If we would like to add word sequences W_3 to our speech recognizer, we would need to

record new audio examples. Moreover, if change just one word in W_1 or W_2, we would have to make new recordings again.

- W_1, W_2 and W_3 can have a lot of common words and sounds, but we cannot use these similarities.
- Our speech examples are designed for specific purpose; we cannot use naturally occurring audio for training our model.

The solution to this problem is to use smaller modelling units, i.e. words or better phonemes. Word models can be used when only a small vocabulary of words should be recognized. Phoneme models are necessary to build large vocabulary speech recognition systems. Nowadays, almost all practical speech recognition systems use phonemes as acoustic modelling units, because phoneme models like can use recordings of any words for training and recognize any word sequence without recording new audio samples.

That way word sequence W is replaced by phoneme sequence P: P_1, P_2 … P_N, where each P_i is trained and estimated independently. A grapheme-to-phoneme (G2P) model describes the mapping between a sequence of phonemes and a word. In its simplest form, a G2P model is a dictionary – a list of words and their corresponding canonical phonetic pronunciations. It can also be a finite state transducer (FST) or a statistical model.

The adoption of Hidden Markov Models (HMM) for acoustic modelling marks an important milestone in a speech recognition research. The statistical approach of HMMs put less emphasis on emulating the way the human brain processes and understands speech and was controversial with linguists since HMMs are too simplistic to account for many common features of human languages (Huang et al., 2014). However, the HMM proved to be a highly useful way for modelling speech and became the dominant speech recognition algorithm in the 1980 (Juang & Rabiner, 2006).

2.5.1. Hidden Markov Models

Hidden Markov Model based acoustic models act on the assumption that a series of observed feature vectors, which represent word or phoneme, are generated by the Markova chain. In the large vocabulary ASR, each phoneme has its own HMM and words are obtained by contacting them into one single HMM. During decoding word HMMs are concatenated into utterance HMM, which are then matched to audio signal.

As it can be seen on **Figure 7**, HMM is finite state machine, an abstract machine that can be in exactly one of a finite number of states at any given time point. Hence, acoustic HMM can change its state only once per input frame.

At each time point t HMM can change its state from i to j with probability a_{ij}. The current state at each time point is not known (hence, the name – Hidden Markov Model). However, at each time point HMM can emit an observation vector x_t from probability distribution $b_j(x_t)$. In the acoustic modelling context, an observation vector x_t is the feature vector which is derived from the input audio signal. Observations vectors are emitted when HMMs is in one of the so-called emitting states.

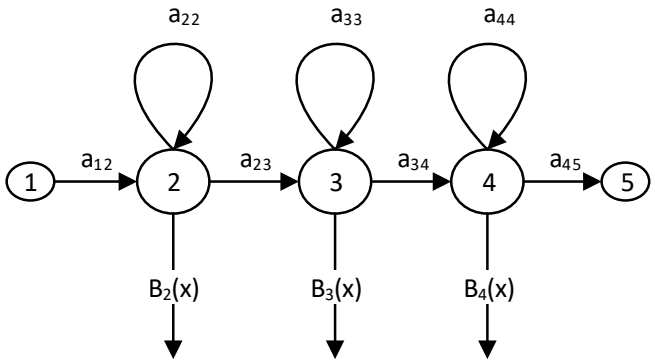

Figure 7. *HMM with 3 emitting states and Bakis topology.*

Most speech modelling and recognition engines use HMMs with so-called Bakis topology. In other words, at each time point HMM can either stay in the current state or go to the next state. Also. there are only two non-emitting states: (1) the start state and (2) the end state. Both states are visited only once and are necessary for making implementation easier.

Probability density $b_j(x_t)$ describes the distribution of observation vectors in position j. For continuous density HMM, $b_j(x_t)$ is typically modelled by Gaussian Mixture Model, hence such acoustic model approach is typically referred as HMM-GMM.

Model for distribution $b_j(x_t)$ for some state j describes some sound event, and this model should be sufficiently complex to distinguish between different sounds and should be sufficiently generalized and robust to deal with the variability of natural speech.

There are three important algorithms for HMMs:

- Baum-Welch algorithm, which is used during acoustic model training to estimate of the parameters of a hidden Markov model given a set of observed feature vectors.
- Forward-backward algorithm, which is used during decoding to find the probability of a given HMM state sequence given the observations vectors.
- Viterbi decoding, which is used to effectively find most probable sequence of hidden state given the observation vectors.

To sum up, the Baum-Welch algorithm is used during training to find a_{ij} and b_j given hidden state sequences (generated from reference text) and feature vectors (extracted from audio). Then, if a_{ij}

un b_j are known, the probability of the $P_A(X|W)$ can be calculated, i.e. the probability of a feature vector X for given word or phoneme W:

$$P_A(X|W) = \sum_s P(X, s|W)$$

where $s=s_1,s_2,s_3...s_T$ is the sequence of HMM states, which generates the sequence of feature vectors $X=x_1,x_2,...,x_T$. P(X,s|W) is calculated as follows:

$$P(X, s|W) = \prod_{t=1}^{T} b_{s_t}(x_t)\, a_{s_t s_{t+1}}$$

where s_{T+1} is the end state of HMM.

Direct estimation of these probabilities is too computationally intensive, also the number of possible HMM state sequences that generate the given observations might be very large or unlimited, especially when T is large (and it's typically is large, because input frames should be very short, like 10-20 msec). In order to solve this problem above mentioned Viterbi and forward-backward algorithms are used. Forward-backward algorithm efficiently calculates P(X,s|W), and Viterbi algorithm is used to find the most probable HMM state sequences.

2.5.2. Context-dependent Phonemes and Tied States

The acoustic properties of the phonemes frequently vary depending on the previous and following phonemes. The position of the phoneme in the word or syllable also frequently has an impact on the articulation. Therefore, in modern ASR systems, the modelling unit is not individual phonemes, but context-dependent phonemes. Models for individual or context-independent phonemes are typically called monophone models. Two of the most popular context-dependent phonemes types are triphones and quinphones.

The triphone is dependent on the one phoneme before and one phoneme after. The context of the quinphone is two phonemes before and after. For example, tuple (a,l,i) is a triphone for phoneme "l", which is preceded by phoneme "a" and succeeded by phoneme "i".

When using triphones, the number of model parameters is dramatically increased. If previously it was necessary to find HMM parameters for each of 33 phonemes, now we need to find HMM parameters for 33*33*33=35937 triphones. Collecting training data which would sufficiently cover all these triphones is too expensive or even impossible. Therefore, the numbers of parameters should be optimized.

The solution to this problem is to share the parameters across similar context-dependent phonemes. In modern ASR systems, the parameters sharing is even more deeper as parameters are

shared across individual HMM states.

The parameter sharing is usually implemented as follows:

- All HMM states are clustered. Number of clusters is much smaller than number of possible triphones and is determined by triphone occurrence statistics.
- A classifier is built to assign clusters for unseen triphones and their HMM states. Usually a decision tree is used (Bahl et al., 1991).
- HMM states that belong to the same cluster share parameters, i.e. a_{ij} and b_j are the same for all HMM states in one cluster.

Parameter sharing is sometimes called "state tying", hence HMM states with shared parameters are called tied states. State clusters are sometimes called senones. A senone's dependence on context could be more complex than just left and right context. It can be a rather complex function defined by a decision tree, or in some other way.

Sharing is not limited to triphones of the same phoneme and can occur across different phonemes, which is sometimes called "phone tree crossing". This option is particularly useful for grapheme-based pronunciation models which use graphemes (letters) instead of real phonemes. State tying between phonemes allows to connect different graphemes that correspond to the same sound.

2.5.3. Neural Network Models

In the end of 1980s neural networks became an attractive method for acoustic modelling. Since then, the neural networks were used in many aspects of speech recognition: phoneme classification (Waibel et al., 1989), isolated word recognition (Wu & Chan, 1993) and speaker adaptation.

However, despite being effective in classifying short time units, such as individual phonemes or isolated words, the neural networks we rarely successful in continuous speech recognition, mainly because of their lack of ability to model temporal dependencies.

In recent years, significant progress in speech recognition studies has been linked with adoption of deep learning techniques together with HMM. For example, TANDEM method uses a deep neural network for feature extraction and then these features are passed to classic HMM-GMM model (Hermansky et al., 2000).

Another approach is to use deep neural networks for modelling HMM state emission probability distributions, i.e. instead of GMM models. This approach was shown to be very effective. State-of-the-art ASR systems employing this approach achieve 33% to 50% relative improvement over best HMM-GMM models (Seide et al., 2011). Also, there have been claims that on some evaluation sets models using this approach can even outperform humans (Stolcke & Droppo, 2017; Xiong el al.,

2016).

The input of hybrid HMM-GMM model are feature vectors and the output is a probability distribution between senones, which is typically implemented using a soft-max layer. The input layer may consume multiple consecutive input frames to increase the context of decision making. There is at least one hidden layer between input and soft-max layer. Hybrid HMM-GMM models are usually trained as follows:

- First, a strong HMM-GMM model is trained;
- HMM-GMM model is used to produce alignments between feature vectors and individual HMM states at each frame;
- Alignments are used to train the DNN.

Then during decoding, the input vectors are fed into the DNN and HMM state probabilities are calculated by performing a forward-pass of DNN.

2.5.4. Speaker Adaptation

As was shown in Section 2.2 the pronunciation of the same phoneme can be very different between various speakers. That means the acoustic model should be able deal with speaker variability.

One of possible approaches is to make acoustic model adaptable. In most modern ASR systems, this is typically achieved using a method called Speaker Adaptive Training (SAT) (Anastakos et al., 1997).

During SAT, acoustic models are trained using adapted features. For HMM-GMM systems this is typically achieved using feature-space Maximum Likelihood Linear Regression (fMLLR). Each input feature vector is multiplied with special adaptation matrix, which is calculated for each speaker. The values in the matrix are trained to maximize the likelihood of the reference text given the input vectors.

Then, in the first recognition pass, the fMLLR matrix is estimated in the unsupervised way, and then used in the second recognition pass to produce the final result.

For HMM-DNN models one of the most popular and successful methods is i-vectors speaker adaptation (Miao, 2014). I-vectors is a special technique introduced for speaker recognition (Dehak et al., 2011). In simplified terms, i-vector is 100 to 1000-dimensional vector which convey the speaker characteristic among other information such as transmission channel, acoustic environment or phonetic content of the speech segment. A detailed description of i-vector paradigm is beyond the scope of this work.

I-vectors can be integrated in various ways:

- I-vector extractor can be trained externally or on the same training data as acoustic model.
- I-vectors can be fed into a separate neural network, which will calculate a linear transformation (e.g. shift) of input features.
- Alternatively, i-vectors can be concatenated with acoustic feature vectors and fed directly into an acoustic DNN.

The steps to train SAT-DNN acoustic models with iVectors can be summarized as follows:

1. Extract i-vectors for training speakers;
2. Update features (by simple concatenation or by some transform);
3. Train DNN model in the new speaker-adapted feature space.

During decoding, we simply need to extract the i-vector for current speaker. Feeding the i-vector to the SAT-DNN architecture will automatically adapt it to this speaker. No initial decoding pass and no DNN fine-tuning are needed on the adaptation data.

2.6. Language Model

2.6.1. Regular Grammars

The purpose of language modelling is to build a mathematical model which describes the language and assign the probability that given sentence belongs to the language. The model could be statistic and estimated from some large text corpus, or it can be deterministic and built manually, to assign binary values (or some weights) for sentences.

The deterministic models typically limit the language to some controlled subset and use a fixed small vocabulary. They are particularly useful in scenarios like recognition of voice commands.

The most common type of limited deterministic language models are grammars, particularly regular grammars. Regular grammar can be defined as follows:

$$G = (N, X, P, S)$$

where

- N – finite set of non-terminal symbols;
- X – finite set of terminal symbols;
- P – finite set of production rules;

- $S \in N$ – start symbol.

Production rules is a pair [non-terminal symbol, sequence symbols from alphabet $N \cup X$], which usually is written in one of the following forms:

$$A \rightarrow aB$$

or

$$A \rightarrow a$$

or

$$A \rightarrow \varepsilon$$

where $A, B \in N$ and $a \in X$.

In simple terms, production rules describe how to replace non-terminal symbols with terminal symbols. If grammar is well defined, this replacement process will end, as only terminal symbols will remain.

There are alternative definitions which simplify design of grammars (for example, by allowing right side to contain multiple non-terminal symbols), however they describe the same language class and can be transformed into form described here.

In speech recognition systems, terminal symbols are words, and production rules generate sentences that this system should be able to recognize. If we assign weight (or probability) to each production rule, we get a stochastic grammar. This allows to calculate probabilities for whole sentences. Using these probabilities, speech recognition system can score multiple hypotheses and select the one with higher score.

The grammars are usually written manually which is a long and hard work. While designing a grammar one must be able to predict a typical end-user phrases, variations in them, be able to find a compromise in the complexity, i.e. a simple grammar will be increase the accuracy of ASR, but users will find it hard to use, on the other hand, a complex grammar will be much more flexible and easier to use, but will adversely affect the quality of recognition. So there have been suggested various ways to train grammars from data (Baker, 1979), both supervised (Clark, 2007) or unsupervised (Clark & Lappin, 2010; Tu & Honavar, 2008).

2.6.2. Count-based Models

Word n-gram language models (LM) are probabilistic models that attempt to predict the next word based on the previous *n-1* words. To approximate the underlying language in this way, the assumption that each word depends only on the previous *n-1* words must be made. This assumption

is very important, because it massively simplifies the estimation of such a model from the given data.

To estimate an n-gram language model, a large text corpus is used. For an estimated model, probabilities are calculated in the following way:

$$P(w_i|w_{i-1},\dots,w_{(i-n)+1}) = \frac{cnt(w_{(i-n)+1},..,w_{i-1},w_i)}{cnt(w_{(i-n)+1},..,w_{i-1})}$$

where *cnt* is the count of given word sequences in a text corpus.

Class-based n-gram models are similar to word n-gram models, but they model relationships between classes, which contain multiple words.

The number of classes is much smaller than the number of words, which make the model smaller, but more robust, the model's generalisation capacity is increased while slightly losing the precision.

The mapping between words and classes, can be defined as following deterministic function:

$$C: w \rightarrow C(w).$$

Then probability of some word w_i can be computer using following equitation (Whittaker, 2000):

$$P(w_i|w_1,\dots,w_{i-1}) = P(w_i|C(w_i))P(C(w_i)|C(w_{i-N+1},\dots,C(w_{i-1}))$$

A probabilistic function can also be used for mapping words to classes. Such function will assign word to some class with some probability. This allows word to belong to multiple classes simultaneously, with some probability assigned for each class. This can be particularly useful for words that can change their classes depending on the context. Then, in order to calculate probability of the word w_i it is necessary to perform summation over all possible word class histories (Ney et al., 1994):

$$P(w_i|w_1,\dots,w_{i-1}) = \sum_c P(w_i|c)\left[\sum_s P(c|s)P(s|w_{i-N+1},\dots,w_{i-1})\right]$$

2.6.3. Neural Network Models

Researchers suggested many other methods for language modelling, which do not perform any n-gram counting. One of the most successful and important alternative methods are neural networks, which was first introduced by (Bengio et al., 2003).

The advantage of neural network approach is better generalization on unseen word sequences. The reason for this is that in neural network language models sparse word histories are projected into a low-dimensional space. In this space, similar word histories are tied together even if they were never observed in the training data.

The theoretical advantage has been experimentally proven many times, however count-based models still remain popular in many cases. This is usually because neural network language models are much harder to train and use, due to:

- much higher computational requirements, especially during training;
- large number of hyperparameters;
- the complexity of implementation.

As of current day, most widely used and successful neural network architecture for language modelling are recurrent neural networks (RNN). Such languages models were first adopted by Mikolov et al. (2010).

In simple case, the RNN model consists of three layers: input layer, output layer and hidden layer (see **Figure 8**).

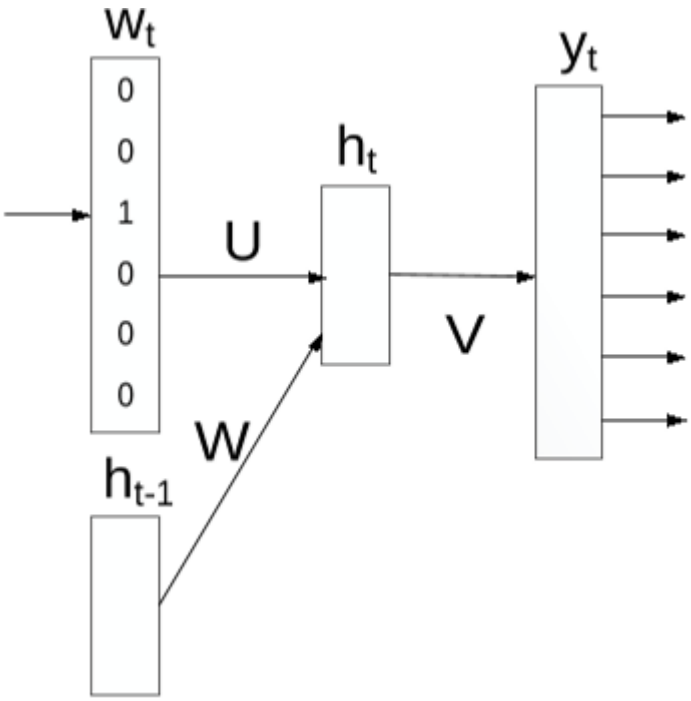

Figure 8. *Recurrent neural network language model*

The input is a vector w_t which represents the current word using a one-hot encoding. This vector is transformed using the input layer (transform U) and then fed into a hidden layer.

Another input is a vector h_{t-1}, which is calculated by the hidden layer for the previous word. This vector is also transformed (transform W) before passing to the hidden layer. At the start of the process, h_{t-1} is initialized with zeros.

Next, the hidden layer calculates the values h_t, which are passed to the output layer, which outputs probability distribution for the next word and usually is implemented as soft-max over language model vocabulary. Hidden layer activation h_t is also passed to the computation for the next

word.

RNNs can theoretically use an unlimited previous context for their decisions. In practice, however, the ability of pure RNNs to remember long-term dependencies is limited, so there exists number of extensions to make remembering long contexts possible (Hochreiter & Schmidhuber, 1997; Cho et al., 2014).

For inflective languages like Latvian there is also another problem – large number of surface forms for each word. As the result the word dimension of the input (word embedding) and outputs layers can be more than 600 000. That makes convergence of the models almost impossible and also requires a lot of computational resources for training and inference, therefore making models not practical. Convergence can be achieved by reducing vocabulary, however this increases out-of-vocabulary rate and makes models less accurate than traditional n-grams. Also, even if such large vocabulary RNN model will provide better WER, the performance drop and hardware requirements can be too big, therefore making it not suitable for commercial deployment.

2.6.4. Entropy, Perplexity and OOV

The probability that the language model assigns to any text that was not in the training data gives a sense of how well the model works on the previously unseen data. The higher is the average probability that is assigned to fluent and grammatically correct sentences, the better is the model.

From information theory point of view language model can be viewed as an information source, which generates a sequence of tokens $w_1,w_2,w_3....w_n$ with probability $P(w_1,w_2,w_3....w_n)$. Tokens can be words, words pairs or sequences, word classes, sub-word units (e.g., syllables). When one token is outputted, information source reduces the uncertainty of the next token. The average information (entropy) for each token can be calculated as follows:

$$H = -\lim_{n\to\infty} \frac{1}{n} \sum_{w_1,\dots,w_n} P(w_1,\dots,w_n) \log_2 P(w_1,\dots,w_n)$$

The source can be assumed to be ergodic, as people can successfully use the language without having to know and remember all the previously spoken words, we can distinguish the words in the conversation based on some of the previous words. This assumption allows to replace the summation of all possible token sequences with one infinite token sequence, which then can be approximate with sufficiently long sequence:

$$\breve{H} = -\frac{1}{n} \log_2 P(w_1,\dots,w_n)$$

Then language model perplexity can be defined as:

$$\mathrm{PP} = 2^{\tilde{H}} = P(w_1, \dots, w_n)^{-\frac{1}{n}}$$

Perplexity can be perceived as the branching factor, i.e. as the average number of possible tokens that can follow any given token. That means the complexity of the speech recognition task can be interpreted as the uncertainty that occurs when choosing one correct token from PP tokens with the same probability. The higher the PP, the more difficult is the recognition task.

To sum up, the lower is the perplexity, the better is the model. So, in order to improve the speech recognition accuracy, perplexity should be minimized. Though sometimes there are exception to this rule. Clarkson (1999) gives an in-depth analysis of the relationships between recognition accuracy and perplexity.

Language models are limited by their vocabulary, which is typically formed by the words that were found in the training corpus. Words outside the model vocabulary can always be found. That means two languages model can be compared in terms of out-of-vocabulary (OOV) rate. OOV is calculated by taking some text T and counting words that are not in vocabulary V:

$$OOV_T(V) = \frac{count\ of\ words\ from\ T\ that\ are\ not\ in\ V}{word\ count\ in\ T}$$

Usually the lower is the OOV, the better model is suited for recognizing texts like T.

2.7. Discriminative Training

The count-based language models, HMM acoustic models and Gaussian mixture models are all generative models. In probability theory and statistics, a generative model is a model for randomly generating observable data values, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences.

Generative models contrast with discriminative models, in that a generative model is a full probabilistic model of all variables, whereas a discriminative model provides a model only for the target variable(s) conditional on the observed variables. In simple terms, a generative method models how the data was generated in order to categorize a signal. It asks the question: based on my generation assumptions, which category is most likely to generate this signal? A discriminative method does not care about how the data was generated, it simply categorizes a given signal.

The positive properties of generative approach for ASR are:

- the problem is described completely;
- a natural training criterion, maximum likelihood estimation (MLE);
- almost closed form solutions by EM (expectation/maximization) algorithm;
- nice from the mathematical point view.

However, there is also a negative side:

- During training, we learn model $P(W,X) = P(W) \cdot P(W|X)$, while what we actually need is $P(W|X)$.
- Density estimation $P(X|W)$ is a harder problem than classification $P(W|X)$, there is no guarantee that the observed data X from sentence W actually have a higher likelihood probability $P(X|W)$ than the likelihood $P(X|W')$ of some other competing sentence W'.
- Integrating hybrid components like neural networks is either hard or "break" generative properties.

The idea of discriminative approach for speech recognition is to estimate $P(W|X)$ directly:

$$P(W|X) = \frac{P(W,X)}{\sum_{W'} P(W',X)} = \frac{P(W) \cdot P(W|X)}{\sum_{W'} P(W') \cdot P(W'|X)}$$

However, there is one very important problem - the sum over all possible sentences in denominator, which results in a very complex optimization problem. In the simpler versions of discriminative training, this is solved by performing optimization on phoneme or word level, however, as the end goal of an ASR system is to recognize utterances, not single phonemes, there have been proposed methods for sequence discriminative training: maximum mutual information (Bahl et al., 1986), minimum classification error (MCE)(Juang et al, 1997), minimum word/phone error (MWE/MPE) (Povey & Woodlang, 2002), minimum Bayes risk (MBR) (Vesely et al., 2013) etc. All these methods use various types of approximations and short-cuts to compute the distribution in the denominator.

Also, usually discriminative training is performed after initialization of $P(W,X)$ using maximum likelihood training. So, there are no gains in simplifying the training. Nevertheless, most experiments show that there are relative improvements by 5-10% in WER over maximum likelihood approach.

One of most popular discriminative training objective functions is maximum mutual information (MMI):

$$\mathcal{F}_{MMI}(M) = \sum_{r=1}^{R} log \frac{P_M(W_r)^k \cdot P(W_r|X_r)}{\sum_{W'} P(W')^k \cdot P_M(W'|X_r)}$$

where M represents the acoustic model parameters, W_r are the training utterances, X_r is the feature vector sequence corresponding to the training utterance r, k is the acoustic scale as used in decoding and P(W) is a language model (usually weakened such as a unigram).

Although likelihood and posterior probability are transformable based on Bayes' rule,

generative and discriminative criteria often generate different results, training of ASR acoustic models using discriminative approach has been shown to provide significant reduction in word error rates. Arthur Nadas (1983) showed that if the prior distribution (language model) and the assumed likelihood distribution family are correct, both MLE and MMI are consistent estimators, but MMI has a greater variance. However, when some of those premises are not valid, it is desirable to use MMI to find the estimate that maximizes the mutual information (instead of likelihood) between sample data and its class information. The difference between these two estimation techniques is that MMI and other discriminative criteria not only aims to increase the likelihood for the correct class, but also tries to decrease the likelihood for the incorrect classes. Thus, discriminatively trained acoustic model in general possesses more discriminating power among different competing hypotheses.

2.8. End-to-end Speech Recognition

Consider there is some input sequence x (of length T) and the output is some output sequence y (also of length T). As long as we have an objective function on the output sequence y, we can train a classifier to produce the desired output. However, in speech recognition output sequence (text) is much shorter than the input sequence (audio wave), which poses a problem for many machine learning methods, especially for neural networks.

One of the great advantages of hidden Markov models is their ability to be trained on an unaligned data. This inherent property of HMMs and corresponding training and inference algorithms makes them a very good choice for speech recognition task.

State-of-the-art ASR acoustic models combine HMMs and neural networks by using neural networks for modelling HMM state emission probability distributions. In simple words, HMMs are used to produce alignments, but acoustic event recognition is performed by neural network. HMMs are also used to produce aligned data (e.g. by inserting “blank” labels) for neural network training.

While pure HMM and hybrid HMM-DNN approaches have proved successful, they have several drawbacks:

1. there are explicit assumptions while designing the state models for HMMs;
2. there are explicit (and often questionable) dependency assumptions to make inference tractable, e.g. the assumption that observations are independent for HMMs;
3. for standard HMMs, training is generative, even though sequence labelling is discriminative;
4. HMM, DNN and language models are all optimized separately, while joint optimization might give better results.

5. during training it's necessary to train a number of intermediate HMM models, this makes training process long and complex, also errors or non-optimal choices for intermediate models will typically affect end model performance.

This gives a motivation to remove HMM's from speech recognition systems and perform speech recognition using neural networks only. However, it was largely impossible before wide adoption of RNN with LSTM or GRU units, and, most importantly, before the introduction of Connectionist Temporal Classification (CTC) (Graves et al., 2006).

The key idea behind CTC is that instead of somehow generating the label as output from the neural network, we instead generate a probability distribution (via soft-max layer) at every timestep (from $t=1$ to $t=T$). Suppose that for each input sequence x (acoustic feature vectors) we have a label ℓ. The label is a sequence of phonemes from some phoneme set L, which is shorter than the input sequence x; let U be the length of the label. We can then decode this probability distribution into a maximum likelihood label. Finally, we train our network by creating an objective function that coerces the maximum likelihood decoding for a given sequence x to correspond to our desired label ℓ.

This requires some clever tricks, objective functions, and output decoding algorithms; which are covered in (Graves et al., 2006) and (Graves, 2008), as well as in many other following research papers (Miao et al., 2015; Chorowski & Jaitly, 2016).

The introduction of CTC inspired many similar methods (Povey et al., 2016) and enabled end-to-end speech recognition. End-to-end models jointly learn all the components of the speech recognizer. Thus, it simplifies the training process and deployment process. There have also been work that shows that feature extraction from raw audio can be performed by neural networks (Sainath et al., 2015). Joint optimization and automatic pronunciation learning may also result in reduction WER.

While CTC has shown tremendous promise in end-to-end speech recognition, it is limited by the assumptions of independence between frames - the output at one frame has no influence at the outputs at the other frames. The only way to ameliorate this problem is through the use of a strong language model (Zhang et al., 2017), which is trained independently on large text corpus. Thus, such CTC-based system is end-to-end-trained, but it is not an end-to-end model when decoding.

An alternative approach to CTC-based models are attention-based models. Attention-based ASR models were first introduced simultaneously by Chan et al. (2016) and Bahdanaua et al. (2016). Unlike CTC-based models, attention-based models do not have conditional-independence assumptions and can learn all the components of a speech recognizer including the pronunciation,

acoustic and language model directly. This makes it not only an end-to-end trained system but an end-to-end model. By the end of 2016, the attention-based models have seen considerable success including outperforming the CTC models (with or without an external language model) (Yu et al., 2016; Chorowski & Jaitly, 2016).

The drawbacks of the end-to-end speech recognition approach are:

1. "end-to-end" approaches attempt to take structure out of the system, like language model, the knowledge of pronunciations of words, or the concept of speech feature extraction, this makes such system less controllable and more difficult to adapt to new domains;
2. by taking out the structure such methods require much larger amount of training data;
3. currently end-to-end ASR systems are unable to outperform state-of-the-art results of hybrid systems, especially if used without the classic language model (thus breaking the pure end-to-end process).

Because of these reasons, this thesis will focus on a more traditional hybrid HMM-DNN methodology and end-to-end methods will be left for future research.

2.9. Related Work for Languages of Baltic States

Before starting the research, the author conducted a small-scale literature review about "speech recognition research for languages of Baltic States, Estonian, Lithuanian and Latvian". This was necessary to verify that there is indeed a knowledge gap and a motivation for a research.

Because review was performed in the beginning of 2014, only papers published in period of 2000-2013 were reviewed. Papers before 2000 were ignored to get selection of recent articles and modern methods.

The research question that the literature review should answer is as follows:

- Is there any research on speech recognition for Estonian, Latvian, and Lithuanian?
- What kind of speech recognition systems are being developed for these languages? Isolated words, continuous speech, large vocabulary, small vocabulary etc.
- What methods are being used?
- What are the state-of-the-art results?

The literature review is expected to answer to these questions and hopefully identify a gap where research has not yet been done or where results are unsatisfactory and could be improved by further research.

2.9.1. Protocol of the Review

At the start the review, a search query was created in order to reflect possible paper titles in the general speech recognition research area. The search query was as follows:

```
“(latvian|lithuanian|estonian) ("speech recognition"|"speech to text")”
```

The search in the Google Scholar index returned 1027 documents published between 2000 and 2013 including. Normally, this would mean that we should make our search query more specific and narrow the results. However, because Google Scholar sorts results by relevance, it was decided to perform first filtering step in batches of 10 and stop when all papers in the batch are rejected. As the result, first 100 papers from 1027 were analysed.

In the first step documents were filtered based on the titles. Papers filtered out were on different research topics or focused on a narrow specific subtopics of speech recognition and did not describe any speech recognition system. I.e. authors of this papers concentrated on some specific questions in speech research, but did not perform any actual speech recognition experiments. 100 papers were analysed and 19 papers remained after this filtering step.

In the seconds filtering step, articles were removed based on the publication type, only peer-reviewed articles from journals and conference proceedings were left. 14 papers remained after this filtering step.

For the third filtering step, the abstracts of the remaining papers were analysed. 3 papers were filtered out because they did not contain any real speech recognition experiments.

The remaining 11 papers were skimmed. Unfortunately, author could not find full text of 1 paper, so it was filtered out. However no more papers were filtered out and all of the remaining 10 were considered relevant to the topic.

List of papers selected for detailed review and analysis can be found in Appendix 1. The findings are summarized in the next section.

2.9.2. Results of the Review

The analysis of the 10 papers on the topic of speech recognition for languages of Baltic states revealed the following:

- Only one paper is devoted to developing speech recognition for Latvian. 4 papers are focused on Estonian and 5 on Lithuanian. This maybe a good indication that speech recognition problem for Latvian is underresearched.
- Subjective comparison of rankings of journals and conference proceedings in which these 10

papers where published, showed that there might a significant difference in quality of research for different languages. Most of the papers on Estonian speech recognition (as well single paper on Latvian) were presented in international well-known conferences on speech and natural language processing. On the other hand, papers on Lithuanian language have been published in local or more general topic journals.

- Also, it should be noted, that 2 papers on Lithuanian speech recognition present a very disputable method – using English or other foreign language recognizer language for recognizing Lithuanian words. This is achieved by emulating Lithuanian word pronunciation with foreign language phonemes. While this method may be an interesting theoretical exercise, it's applicability in practice is disputable.
- The most used acoustic modelling method in 10 analyzed papers is Hidden Markov Model models. This is a classic method, however it's considered outdated, as most of the state-of-the-art systems use Deep Neural Networks. 1 paper on Latvian and 1 on Lithuanian also experimented with Artificial Neural Networks. 1 paper on Lithuanian speech recognition also used classic, but outdated Dynamic Time Warping method. The conclusion what we can draw from this is that there is a lack of research using modern state-of-the-art methods for speech recognition for Estonian, Latvian and Lithuanian.
- All 4 papers for Estonian language, 2 papers for Lithuanian and 1 paper for Latvian present continuous speech recognition systems. 3 other papers for Lithuanian focus on a simpler isolated word scenario which is not considered relevant nowadays.
- There were 3 papers on large vocabulary speech recognition for Estonian and 1 for Latvian. 3 papers focused on medium vocabulary speech recognition for Lithuanian. 1 paper described limited (or small) vocabulary recognition system for Estonian and 2 for Lithuanian.
- All papers reported evaluation scores however it is not possible to compare one to another (because of different testing sets, different vocabularies etc.). There is for a standardized testing sets for each of the reviewed languages.

To sum up, this small-scale literature review helped to identify a clear knowledge gap – there is very little research on speech recognition for Latvian language. As the result author decided that this will be the main topic of his research.

2.10. Related Work for Latvian Language

Automatic speech recognition technologies for Latvian have a relatively short history. Before 2014 there was no speech corpus, which could be used for ASR purposes, available. Therefore, only

one paper was found during literature review above.

Oparin et al., (2013) developed a broadcast speech recognition system for the Quaero project (Lamel, 2012) using acoustic model bootstrapping. This system was developed in about a month period without use of a transcribed audio training data. The only supervision was mapping of Latvian phones to the ones existing in other languages to select cross-lingual seed models, text normalization and grapheme-to-phoneme conversion. Iterative application of various methods and tools led to a reduction from the initial word error rate of 74% down to 19%. A WER of 20.2% was achieved in the Quaero 2012 STT evaluation. Unfortunately, the results of this work are not reproducible as neither the ASR, nor the training and evaluation data were made publicly available.

Later, as the result of work by Pinnis et al. (2014) a Latvian Speech Recognition Corpus (LSRC) was created to facilitate the speech recognition research for Latvian language. The corpus consists of 100 hours manually annotated speech audio data. More detailed description of LSRC is presented in section 0.

Since the creation of the LSRC, ASR technologies for Latvian have been actively researched simultaneously by many researchers. The author of this work performed a more general and comprehensive research, which aims to provide baseline solutions for all components of speech recognition for Latvian, while other authors typically focussed on single components of ASR and/or smaller tasks.

Dargis & Znotiņš (2014) created a baseline system for keyword spotting Latvian Broadcast Speech, which was using a large vocabulary ASR trained on a LSRC corpus. This baseline system was then used in the development of Media Monitoring System for Latvian TV and Radio (Znotiņš et al, 2015). The word error rate of this system is rather high - 51%, however in keyword spotting task the system achieves 81% F1 score (90% precision and 73% recall).

Auzina et al. (2014) used phonetically annotated part of LSRC to compared various rule-based and statistical methods for grapheme-to-phoneme modelling (which is an important part of ASR). This work is continued in the bachelor thesis of Darģis (2014), where grapheme-to-phoneme and language modelling for Latvian are investigated. In this bachelor thesis language models are trained on small corpora and only old HMM-GMM acoustic models are used. Also, this bachelor thesis does not report any WER, but uses other performance measures for some reason.

Master thesis of Strigins (2015) also focusses on language modelling, but it uses modern HMM-DNN acoustic models and a large text corpus. The text processing methods proposed in this Master thesis allowed to significantly improve the quality of a large vocabulary speech recognition system for Latvian. Experiments with sub-word recognition are performed to better model the infective

nature of Latvian language, however results show that using larger training data and vocabulary works better and is much simpler. The work was advised by author of this thesis.

The success of Latvian Speech Recognition corpus lead to the development of Latvian Dictated Speech Corpus (Pinnis et al, 2016), which contains about 9h of dictated speech with special commands (like text formatting and naming punctuation). This corpus allowed to develop and evaluate the first dictation system for Latvian language.

Finally, in August 2017, Google added Latvian speech recognition to its services[1]. For obvious reasons, the methodology and data that was used in the development of this ASR is not publicly available, however it is an important milestone for automatic speech recognition for Latvian, as it enables wide audience of Latvian-speaking Google product users to use their native language. Google also provides an API, so developers all around the world can implement Latvian speech recognition in their applications. Around that time UK company Speechmatics also started to provide ASR service for Latvian. Both companies have not published any information on data and methods that they use for Latvian.

The comparison between Google Latvian ASR, Speechmatics Latvian ASR and ASR developed by author of this work can be found in the final section of the thesis.

[1] https://www.blog.google/products/search/type-less-talk-more/

3. ACOUSTIC MODELLING FOR LATVIAN

This section is devoted to acoustic modelling for Latvian language. It describes training and evaluation data, methods used for acoustic modelling, grapheme-to-phoneme modelling, acronym recognition, other parts of acoustic modelling and obtained experimental results.

The section 3.6 on the automatic data collection methods is based on the publication Salimbajevs (2018). Other results of this section are published in Salimbajevs & Pinnis (2014), Salimbajevs & Strigins (2015b) and Salimbajevs (2018).

3.1. Acoustic Model Training Data

As described in previous section, speech recognition relies on statistical models that are built using large data sets. That means the very first question that needs to be solved to build speech recognition models for Latvian is finding or collecting the data. A large collections of transcribed audio recordings are called speech corpora. Collection and annotation of speech corpora is expensive process. Fortunately, such data for Latvian language became available when author of this thesis started his research on speech recognition for Latvian. **Table 1** summarizes speech corpora used for training acoustic models in this work.

Table 1. *Acoustic model training corpora*

Corpus	**Size, h**	**% of all**
LSRC	100	34.0%
LDSC	8	2.7%
Saeima11	186	63.3%
Total	**294**	**100%**

The first speech corpus created for speech recognition research for Latvian language is the Latvian Speech Recognition Corpus (LSRC) by Pinnis et al. (2014). The corpus consists of 100 hours of orthographically annotated speech audio data and 4 hours of phonetically annotated speech audio data. The corpus is both phonetically rich (has a good triphone coverage) and balanced in many dimensions. It captures both spontaneous (61%) and prepared speech (39%), both genders (54% men and 46% women) and different ages groups (16-25: 12%, 26-50: 62.4%, 51-75: 25.6%). The audio recordings were done in different environments (office, street, car …) and with various recording devices. The audio is stored WAV format, with 44.1kHz sampling rate and 16 bits allocated per sample. Corpus contains approximately 837K words by 1851 different speakers.

The second corpus used to train acoustic models in this work is the Latvian Dictated Speech Corpus (LDSC). The LDSC is the speech corpus created specifically for speech recognition in Latvian. The total length of the DSC is 9 hours 19 minutes and 46 seconds. It consists of 287 speeches (22,763 running words) that are spoken by 30 speakers (15 men and 15 women; however, the data distribution is not equal, as length of male recordings is only 3 hours and 52 minutes) of two age groups (16-25

and 26-50). From 9 hours of LDSC, 8 hours are appended to the training data and the remaining data is used for evaluation tasks.

The speeches were recorded using 23 different recording devices (15 smartphones, 4 tablet computers, 2 desktop computers and 2 laptops). Each speech was recorded simultaneously with one to four different devices (therefore, the unique total length is smaller – 4h and 15 minutes).

The corpus features 36 different speech commands, which are used in dictation scenarios, and 93 different pronunciation variants of the commands. The most frequent commands are punctuation commands (over 68%), followed by special symbol commands (over 22%). The corpus contains both spontaneous and prepared (read) speech.

While 100 hours is enough to train baseline acoustic models, more complex models that use methods like deep neural networks can make use of much larger amounts of data (like thousands and tens of thousands of hours). Because manual collection and annotation of new speech recordings is an expensive process, a fully automatic method was developed which allows to create speech corpora from inaccurately transcribed speech recordings that can be found on the web. This method was developed as part of this research and allowed to obtain additional 186 hours of acoustic model training data from the Latvian Saeima webpage. More details on the method and the results of using this corpus in the training are presented in section 3.6

3.2. Evaluation Data

The speech recognition systems are evaluated by decoding some evaluation recordings and then comparing the recognized text with reference and calculating WER. Evaluation data should be accurately annotated and represent the intended usage scenario, if one wants to choose the best model for telephone speech recognition, then the evaluation should be performed on real telephone speech data. The summary of evaluation corpora used in this work is presented in **Table 2**.

***Table 2**. Evaluation corpora*

Corpus	Domain	Size, h
LSRC held-out	General	0.83
EvalWebNews	General/news	0.38
EvalGeneral	General	2.50
LDSC-test	Dictation	1.00
Saeima-test	Saeima	1.00
Address-test	Street addresses	0.13

In first experiments a held-out set of 1500 utterances (about 50 minutes) from Latvian Speech Recognition Corpus was used. This held-out set has several issues:

- It consists of very short utterances (2 seconds in average).
- Test utterances are not semantically segmented (utterances can begin in the middle of

sentence and end abruptly) and therefore can have low language model scores.

- Because LSRC was created from recordings from many various sources, the held out set also contains very different utterances from various domains and environments.

Due to these reasons, it was decided to create other evaluation sets, which would contain longer semantic utterances from a single domain.

The first test set to be created, was a small 23-minute long annotated speech corpus was recorded using a smartphone. The corpus contains records of 10 unprofessional speakers who read random news from web news portals. For conciseness, we will call this corpus - "EvalWebNews".

System that is more accurate on average, can be outperformed by a worse system on some small subset of data (small sample size problem), so it's important that evaluation set is large enough to show real differences between ASR systems. As Latvian ASR became more and more accurate, the probability of such evaluation errors increased, so it was decided to create a larger test set for general domain. The test set is called "EvalGeneral" (900 utterances by 480 speakers) and consists of EvalWebNews, Saeima-test and more than one hour of Riga city council session recordings.

Next, roughly 1 hour of the LDSC recordings is used for ASR evaluation on dictation scenario (this test corpus will be called LDSC-test). It contains about 1800 utterances with various dictation commands from 9 speakers.

Finally, for the Saeima adaptation task the evaluation corpus was collected and annotated as follows:

- First about 10h of recordings of sessions from 2014 to 2016 were downloaded from Saeima webpage.
- Recordings were chosen semi-randomly, prioritizing recordings of more recent sessions with large speakers count, but without actually looking inside recordings.
- Next these recordings were segmented into single speaker short utterances. By randomly picking from all utterances a 1.5 hour subset was selected.
- This 1.5 hour subset was processed by general domain Latvian ASR and then corrected by human annotators.
- Finally, recordings that are incorrectly annotated, too noisy or simply not in Latvian were filtered out.

As the result, a 1 hour long speech corpus of the debates in the Parliament of Latvia was collected (or "Saeima-test" for short). The corpus contains 439 segments, which were recorded by

about 300 different speakers (estimated).

For evaluation of street address recognition system, a small test set, called "Address-test" was recorded. This test set consists of 136 audio recordings (16 MB, 8 minutes 44 seconds, 506 words). The audio data were recorded by 35 different speakers (12 women and 23 men). 81 recordings were done in office rooms, and 55 entries in cars (with and without background music).

3.3. HMM-GMM Speech Recognition Models

Developing a speech recognition for a new language is non-trivial task which involves solving many sub-problems. Before starting to train complex neural network models, one must find answers to some smaller questions like: what will be phoneme set, how words will be transformed into phoneme sequences, what data will be used for acoustic model training, how to prepare data for language model training and so on. This subsection is based on results of publication Salimbajevs & Pinnis (2014).

Training most of the modern neural networks requires to build simpler HMM-GMM models first. End-to-end or "flat-start" trainable neural network models appeared only recently and, in most cases, are behind of traditional hybrid systems in terms of recognition quality.

3.3.1. Initial HMM-GMM Acoustic Models

The initial acoustic model developed at the beginning of this research was a 40 phoneme, 3-state Hidden Markov Model with 3000 tied states, each described by 8 Gaussian mixture components. There was no speaker adaptation, feature transformation, or multi-pass decoding methods. The goal was to identify problems in the training dataset and get understanding of very basic core methods before investigating more advanced acoustic modelling and acoustic model adaptation methods. For conciseness and easy disambiguation let's call this initial model - "INT-GMM-8".

The model was trained using the CMU Sphinx toolkit (Lee et al., 1990) with 13-dimensional Mel Frequency Cepstral Coefficient (MFCC) features on a 100-hour Latvian Speech Recognition Corpus (LSRC) (see section 0 for more details). A held-out set of 1500 utterances (about 50 minutes) from training data (called "LSRC held-out") was used for testing and optimization.

Because any speech recognition experiment requires both acoustic and language models, a "stub" 3-gram language model was trained on the raw transcripts of the LSRC held-out set. The idea was that such language model would "simulate" perfect language knowledge for testing sentences and it will be possible to concentrate on acoustic model only.

After performing multiple experiments with different feature extraction parameters, we identified parameters which work best with LSRC. The most notable changes were: (1) using DCT-

II transform instead of the default “legacy” transform and (2) adding liftering (cepstrum filtering).

When the initial model was trained, it was used in experiments with different phoneme sets, voiced fillers, and grapheme-to-phoneme models. The PocketSphinx (Huggins-Daines, 2006) decoder from the CMU Sphinx toolkit was used for decoding and evaluating speech recognition quality.

3.3.2. Grapheme-to-phoneme Model

As large vocabulary ASR models use phonemes to recognise words, it is important to select an optimal phoneme set which would allow us to (1) acquire sufficient statistics for phonemes and build more accurate models and (2) unambiguously and effectively recognize words from recognised phoneme sequences.

Latvian language has a highly phonemic orthography, i.e., pronunciation and spelling mostly correspond in a predictable way. Therefore, it is natural to use a grapheme-based model, the initial phoneme set was formed by 33 phonemes which have a one-to-one correspondence to letters of the Latvian alphabet and 7 most frequently used diphthongs (Latvian has 10 diphthongs in total, although some diphthongs are mostly limited to proper names and interjections): [ai], [au], [ɛi], [iɛ], [oi], [ui] and [uo].

A rule-based grapheme-to-phoneme (G2P) algorithm was developed, which basically maps letters directly to phonemes using one-to-one correspondences. The algorithm also replaces two letter combinations “ai”, “au”, “ei” etc. with corresponding diphthongs, which is its only difference from a pure grapheme-based approach.

Several experiments were performed in order to find optimal phoneme set and G2P rules. The experiments consisted of the following actions:

- Making changes to the phoneme set and G2P algorithm
- Retraining the acoustic model.
- Calculating the WER on the test set.

Experiments were performed using acoustic model “INT-GMM-8” and stub language model. At this stage, we achieved a WER of 14% on the held-out set. Such a good result should not be surprising, considering the “ideal” language model that was used in this experiment.

Not every possible variation of phoneme set was evaluated, because in many cases implementation of reliable G2P algorithm, would be too complex. Instead experiments focused on variations that are phonetically motivated and easy to implement reliably. For example, there are 3 types of ‘o’ in Latvian and rules that describe the usage of each ’o’ are relatively simple and robust.

The following changes to initial phoneme set and G2P were tested:

- Removing diphthongs one by one from initial diphthong list;
- Distinguishing between different 'o' phonemes;
- Replacing [c] grapheme-phoneme with [t]+[s] combination (because such separate phoneme does not exist);
- Replacing long vowels with two short ones (e.g. [ā] -> [a] + [a]);

After these experiments the best performing phoneme and rule set was identified. This phoneme set, which will be called *baseline phoneme set*, contains (1) 33 phonemes which have a one-to-one correspondence to letters of the Latvian alphabet and (2) 4 diphthongs ([ai], [au], [ɛi] and [iɛ]). Any small deviation from this baseline set resulted in a small increase of WER (see examples in **Table 3**).

Table 3. *Experiments on changing the baseline phoneme set*

Phoneme set description	**Word error rate**
Baseline phoneme set	13.7%
Baseline phoneme set (no phone tree cross)	18.2%
Separating 'o' into [o] and [uɔ]	14.2%
Phoneme [ss] is added	14.1%
Phoneme [c] is replaced by [t]+[s]	14.2%
Long vowels replaced with two short vowels	14.9%
[ɛi] diphthong is removed	13.9%
[ui] diphthong is added	14.2%

The experiments also show that turning on the phone tree crossing option in CMU Sphinx gives a strong improvement. This option enables sharing HMM states of different phonemes, which is very useful for grapheme-based models. When using this option in the described setup the WER is improved by 3-5% absolute over a setting where the option is turned off.

The LSRC includes 4 hours of data that are annotated both phonetically and orthographically. This allows to evaluate the quality of the automatic G2P conversion algorithm.

The G2P and LSRC phoneme sets are not identical, the LSRC phoneme set is much more detailed and includes all G2P phonemes as a subset. That means phonetical transcriptions in LSRC can be simplified and transformed to G2P set. Depending on how to define this transformation, one can obtain a phoneme error rate of 9-13% for given G2P algorithm. Almost half of these errors are substitutions, deletions constitute less than 1% of the errors, and the remaining errors are insertions (i.e., a phoneme is inserted by the rule-based G2P where the human transcription does not have one).

Several experiments have been also performed in order to understand how different G2P errors can influence the WER. In these experiments, synthetic errors were injected into the G2P algorithm,

then the acoustic model was retrained, and WER evaluation was repeated. **Table 4** shows some examples of these experiments. It was concluded that there is no simple correspondence between the G2P error rate and the WER, because the resulting WER depends on the character of specific errors, e.g., two G2P algorithms can have a similar phoneme error rate, but a very different WER. Some errors such as substitutions between similar phonemes have little effect on WER, while errors such as the insertion of an extra phoneme can lead to noticeable WER degradation.

Table 4. *Effect of different G2P errors*

Error description	G2P phoneme error rate	Word error rate
Insertion of extra phoneme "g" after "k" in some cases	+ 24% absolute	+ 0.5% absolute
Insertion of extra phoneme "k" after "e" in some cases	+ 15% absolute	+ 3% absolute
Using separate phonemes instead of diphthongs	+ 4.8% absolute	+ 0.5% absolute
Deletion of phoneme "o" after consonants in some cases	+ 0.5% absolute	+ 2% absolute
Substitution between similar sounds "p" and "b"	+ 4.8% absolute	+ 0.2% absolute

3.3.3. Statistical Grapheme-to-phoneme Model

To address and lower the number of insertion errors, experiments with two more complex G2P models were performed. The first one was trained with Phonetisaurus (Novak et al., 2012), which utilises weighted finite-state transducers (WFST) for decoding a representation of a grapheme-based n-gram model trained on data aligned by an advanced many-to-many alignment algorithm (which is a variant of the EM algorithm) (Jiampojamarn et al., 2007). The second one is a statistical machine translation (SMT) model which translates from "grapheme" language to "phoneme" language (Auzina et al., 2014).

Both models were evaluated on a small held out data set from the phonetically annotated corpus. The phoneme error rate for both models is given in **Table 5**.

Table 5. *Advanced G2P models*

G2P model	Phoneme error rate
Phonetisaurus WFST model	5.24%
Statistical machine translation	3.26%
Baseline G2P	9-13%

Both models achieved significantly better results in terms of phoneme error rate than previously described rule-based G2P algorithm. The superiority of the SMT model can be explained by the fact that the Phonetisaurus model is trained on a pronunciation dictionary which was extracted from the phonetically annotated 4-hour corpus and includes all pronunciations from this corpus. At the same time, the SMT model was trained on the full phoneme transcriptions of the training set, not just

isolated words. This allows the SMT model to take into account word boundaries and phonemes from adjacent words. After training is done, the SMT model is used to translate all transcriptions. This translation is then processed, and a static pronunciation dictionary with multiple pronunciation variants is created.

Despite the better phoneme error rate, no improvements in WER were observed. Moreover, the result degraded significantly in the case of the SMT-based G2P model, because it introduced a lot of ambiguous pronunciation variants. This ambiguity comes directly from the nature of human speech which is captured by precise training labels. As a result, words in our static dictionary have a large number of pronunciations, many of which overlap with the pronunciations of other words, making the "recovery" of the right word strings from such a dictionary difficult. This result corresponds with the findings by Saraçlar et al. (2000).

As we were unable to improve upon the baseline rule-based G2P, it was decided to use this model for future experiments.

3.3.4. Filler Word and Noise Models

While speaking, humans can make grammar mistakes, repeat words, make corrections and restart whole sentences, laugh, breath into microphone, insert so called "filler words" (voiced pauses like "ah", "uh", "er", "um"). Ideal speech recognition system must be able to deal with all these defects of spontaneous speech (Butzberger et al., 1992).

Table 6. *Noise and filler models*

Noise/filler	**Occurrences in training corpus**
[e], [ē], and their variations	13,192
[m] and its variations	1,060
[a], [ā], and their variations	1,263
[h], [hmm], [kh], etc.	126
Mix of [n], [en], [s], [u]	162
Mix of rare voiced fillers	114
Non-speech noise	4,481
Breathing	45,041
Laughing	431
Silence	18,290

In this work, we train noise and filler models for filtering out non-speech noises and voiced pauses. The LSRC has 107 unique labels for voiced pauses and non-speech events. It quickly became evident that training such a large amount of noise/filler models is not effective. Some of these labels are almost identical, while some are too rare for acquiring sufficient statistics for training and generalisation of reliable acoustic models.

Therefore, when training HMM-GMM acoustic models using CMU Sphinx these labels were

grouped into 10 more generic noise and filler models (listed in **Table 6**). While it is possible to use the same list of fillers for Kaldi framework (Povey et al., 2011), it would require changes inside training procedures or adding fillers as words to the language model. For Kaldi models a more natural choice is to combine 10 filler models into 2 models:

- 1 filler\silence model for non-speech noises. E.g. sounds that are not spoken words, such as breathing, laughing etc.
- 1 garbage model for spoken noise. For example, fillers, foreign words and fragmented words that were not fully pronounced.

3.3.5. Advanced HMM-GMM Models

After initial experiments have been successfully performed and initial INT-GMM-8 system has been built, it was decided to proceed with more complex models, that perform multiple input feature transformations and are able to adapt to vocal characteristic of current speaker.

There are number of ways how to improve previous HMM-GMM model for Latvian, which can be categorized into following main groups:

- Collecting more training data.
- Using more advanced features, e.g. increasing context information, advanced preprocessing and normalization.
- Using more complex and larger models, e.g. increasing number of Gaussians, increasing senone count etc.
- Speaker adaptation, e.g. training gender-specific models, training speaker adaptive (SAT) models etc.
- Discriminative training.

Initial HMM-GMM models were trained using Sphinx toolkit. Many of the above-mentioned options are not available in Sphinx. Moreover, more advanced neural network models are also not available. Implementing these methods from scratch may take considerable time. Also, it would not be very reasonable, as many of these options are available in Kaldi framework. Therefore, the rest of research and experiments were performed with Kaldi.

The stub language model which was used previous sections, can be useful when making some initial decisions, but it's not adequate for more serious speech recognition experiments. Therefore, when evaluating updated HMM-GMM acoustic models, 3-gram WebNews-LM-1-100 language model (described in section 4.1) is used. This LM is trained on WebNews text corpus (version 2014)

and vocabulary is limited to 100 thousand most frequent words. Because of RAM constraints the model is pruned to about 50MB (creating Kaldi decoding graph from 100MB language model requires 8+ GB of RAM which was not available for first experiments).

Table 7. *Comparison of Sphinx, Kaldi and different advanced methods*

Model	Toolkit	Features	WER, %
3000 senones, 24000 Gaussians	Sphinx	MFCC	52.8
3000 senones, 96000 Gaussians	Sphinx	MFCC	50.9
5000 senones, 40000 Gaussians	Sphinx	MFCC	53.1
5000 senones, 160000 Gaussians	Sphinx	MFCC	55.6
4000 senones, ~90000 Gaussians	Kaldi	MFCC	47.5
3000 senones, ~70000 Gaussians	Kaldi	PLP	49.7
4000 senones, ~90000 Gaussians	Kaldi	PLP	47.5
5000 senones, ~97000 Gaussians	Kaldi	PLP	47.9
4000 senones, ~90000 Gaussians, LDA	Kaldi	MFCC	44.9
4000 senones, ~90000 Gaussians, LDA	Kaldi	PLP	44.3
4000 senones, ~90000 Gaussians, LDA, SAT	Kaldi	PLP	42.5
4000 senones, ~90000 Gaussians, MMI	Kaldi	PLP	42.5
4000 senones, ~90000 Gaussians, bMMI	Kaldi	PLP	42.2
4000 senones, ~90000 Gaussians, LDA, SAT, MMI	Kaldi	PLP	40.9
4000 senones, ~90000 Gaussians, LDA, SAT, bMMI	Kaldi	PLP	40.5

Table 7 contains results of experiments performed to evaluate effect of different speech recognition toolkits, features and advanced techniques. Evaluation was performed on the EvalWebNews test set.

Several conclusions can be drawn from these results:

- Kaldi outperforms CMU Sphinx using models of comparable complexity and the difference is even bigger when using more advanced methods not present in Sphinx.
- 4000 seems to be the optimal senone count for HMM-GMM models and LSRC corpous.
- PLP features perform better than MFCC when combined with LDA and other advanced methods.
- The best result is obtained when all the advanced methods are combined together.

Kaldi framework has its own system for acoustic model names, so for conciseness and according to Kaldi practices, we will call this acoustic model - “LSRC-tri3b”.

3.4. Feed-forward DNN Acoustic Model

The word error rate achieved by previous LSRC-tri3b system is still very high for a general-purpose speech recognition. Also, it uses traditional HMM-GMM based methods that are surpassed in all areas by deep learning methods. Starting around 2009-2010 (Hinton et al., 2012) modern

successful speech recognition systems widely use deep neural networks. Moreover, the use of deep neural networks in speech recognition remains a very active research field. Every year several new methods are introduced that further improve the quality of speech recognition.

So, it was decided that further improvement of Latvian ASR is not possible without adopting state-of-the-art neural network approach. The following acoustic model was trained:

- 13-dimensional PLP features.
- 37 base phonemes.
- 2 filler/garbage models from section 3.3.2.
- 100h LSRC training corpus with alignments obtained from LSRC-tri3b model.
- i-vectors for speaker adaptation (Miao et al., 2014).
- HMM-DNN (Kaldi nnet2) approach with the following neural network architecture:
    - the input of neural network is one audio signal frame, that is described with 113 dimensions (13 PLP dimensions + 100 i-vector dimensions);
    - special input layer, that combines input vector with input vectors from 7 previous and 7 next frames, this create input of 113*15=1695 dimensions;
    - special LDA layer, which performs LDA transform and reduces dimension count to 295;
    - 4 hidden layers with Euclid-norm as non-linearity (activation) function;
    - SoftMax output layer with 3161 dimensions (corresponding to senones).

Because this model uses Kaldi nnet2 framework and LSRC for training, it will be called LSRC-nnet2. The model can be further improved by performing discriminative SMBR training (Vesely et al., 2013), the improved model will be called LSRC-nnet2-smbr accordingly. **Table 8** shows the results of evaluation of new LSRC-nnet2 and LSRC-nnet2-smbr acoustic models on the EvalWebNews test corpus. In this evaluation the WebNews-LM-1 language model from section 4.1 is used, vocabulary is 200K word units and model is pruned to 50 and 100 MB.

Table 8. *Evaluation of LSRC-nnet2 model*

Acoustic model	Language model size, MB	WER, %
LSRC-tri3b	50	38.26
LSRC-nnet2	50	37.39
LSRC-nnet2	100	35.50
LSRC-nnet2-smbr	50	36.89
LSRC-nnet2-smbr	100	**35.05**

At least three conclusions can be from these results. First, as expected deep neural network outperform classic HMM-GMM models. Second, using less aggressive pruning and increasing size of language model allows to achieve significant improvements. Finally, word error rate is still high.

3.5. Acronym Recognition

The ASR prepared in previous section cannot recognize any acronyms and brand names. This is due to several reasons: (1) during language model training corpus pre-processing, tokens that are written in uppercase can be lowercased, but tokens containing digits are filtered out, (2) baseline G2P algorithm provides incorrect pronunciations for acronyms.

Our test corpus contains only few words like this (and they were out-of-vocabulary), so this error was not identified during previous error analysis.

Before doing any modifications to the ASR, it was important to evaluate the current situation. For this, 253 utterances with 267 acronyms were selected from LSRC audio corpus.

First, the current system was evaluated on this subset. As expected no acronyms were recognized correctly. Next, acronyms were added to the WebNews-LM-2 language model (by adding list of exceptions to the filtering), but G2P model was left unchanged. The result was the same. After that, G2P was extended by adding special rules for acronyms. This time 30% of acronyms were recognized correctly (see details in **Table 9**).

***Table 9**. Acronym recognition using special rules in G2P*

Acronym	Occurrences	Correctly recognized, %
TV	85	20
LV	72	44
LNT	31	29
ASV	24	21
PSRS	17	29
LNNK	11	36
PCTVL	6	17
LTV	6	0
LMT	6	67
ABLV	5	0
SS	1	0
RNA	1	100
MFFF	1	100
LP	1	0

Next, it was decided to collect a bigger list of acronyms and provide manually checked pronunciations. For this, words in uppercase or mixed with digits were extracted from WebNews text corpus and sorted by frequency. This resulted in a giant (tens of thousands of words) list of candidates, many of which were noise in the corpus.

Intuitively correct acronyms should be more frequent in the text corpus than some noisy tokens. So, it was decided to take 1000 most frequent candidates, process them with adapted G2P algorithm. After manual check, all of the selected candidates were identified as true acronyms. However, pronunciations created by automatic method were not always correct and were manually edited.

Then, acronyms were added to the exceptions list of text filtering procedures and acronym pronunciations were added as exceptions into G2P. The updated and retrained language model is called WebNews-LM-2-ABBR.

Because EvalWebNews corpus have only few acronyms, a new small test corpus was created - "AbbrTest". 110 most frequent acronyms were selected from the previous list of 1000 and for each acronym a sentence have been manually created. 5 people were asked to say these sentences (22 sentences each) and make audio recordings using microphone.

The updated ASR was evaluated on this AbbrTest corpus and WER of 16.42% was achieved for whole sentences. 24% of acronyms in test sentences were not recognized correctly. The WER on EvalWebNews decreased by 3.5% relative This result allows to conclude that ASR language model and G2P was successfully adapted to recognize most popular acronyms in Latvian.

3.6. Automatic Acquisition of Training Data

Training of a Deep Neural Network acoustic model for an automatic speech recognition (ASR) system requires large amounts of transcribed audio. For a general-purpose ASR, the training data should be as diverse as possible. Recordings should contain both prepared and spontaneous and isolated and continuous speech by various speakers in various conditions.

For Low Resource Languages (LRLs), where there may only be a few hours of transcribed audio, this is a very serious issue. Latvian language is a little bit luckier, as there are a 100h Latvian speech corpus. However, even this is not much when compared to the resources available for languages like English.

In recent years, improvements in data storage and networking technology have made it feasible to provide Internet users with access to large amounts of multimedia content. This content can be automatically collected and processed for the purpose of training statistical models. However, in many cases, this content is not structured or organised in an accurate and machine-readable form.

For example, the Latvian Parliament (Saeima) website contains a large archive of video recordings of parliamentary sessions and edited transcripts. One may want to collect this data and use it for the development of a general-purpose speech recognition system. This data can also be used for adapting existing ASR for transcription of Saeima sessions.

Unfortunately, edited transcripts do not have any timing information. Also, these transcripts are not normalised and are not 100% accurate. There are differences between what was said and what is written in the edited transcripts, and this is due to several reasons:

- While speaking humans can make grammar mistakes, repeat words, make corrections and restart whole sentences, edited transcript contains only final, grammatically correct and re-formulated sentences without these kinds of typical speech defects.
- Edited transcript is a written document, and it should obey specific formatting rules and contain additional information, like speaker names and summaries, that is embedded into the text.
- This also means that numbers, dates, percent signs, etc. are written with digits and symbols, not as words. Converting these tokens back to words is complicated and error-prone for inflected languages like Latvian and Lithuanian.

This section describes the method what was used to obtain additional training data by aligning audio and edited transcripts from Latvian parliament websites (Salimbajevs, 2018). This helped to get noticeable improvement for the Latvian ASR. The process is fully automatic and does not require any human labelling of audio data. The described method is not specific to Saeima session transcript and can be applied to other sources of audio with imprecise or partial transcription.

The alignment between long audios and their corresponding transcripts has been previously studied in the context of various applications. Panayotov et al. (2015) use existing ASR and audio-alignment techniques for creation of a large training corpus from public domain audio-books, and Anguera et al. (2014) use ASR for different languages and a clever phoneme-based alignment approach for training speech recognition with very limited language resources. Prahallad & Black (2011) describe the creation of aligned corpora for building text-to-speech systems, and Hazen (2006) focuses on the automatic alignment and correction of inaccurate text transcripts through an iterative process.

The method that are used in this work is similar to Panayotov et al. (2015) and Hazen (2006). The main differences are the use of the SpkDiarization toolkit (Rouvier et al., 2013) for segmenting large audio recordings into smaller manageable segments and for providing speaker diarisation, an optional intermediate step where the retrained model is used for better alignment, a different utterance extraction process and the fact that it is being done for the less-researched and less-resourced Latvian language.

3.6.1. Processing of Saeima Transcripts

Latvian Saeima website contains a huge archive of audio and video recordings of parliamentary

sessions. A web-crawler script was implemented and used for collecting video files and the corresponding human-edited reference transcripts from the website. As the result, 300 hours of video recordings were downloaded (downloaded files correspond to recordings during period of 2011-2014).

Next, audio is extracted from each video file and processed by the LIUM SpkDiarization toolkit, which segments audio into smaller parts and groups them into clusters (that should correspond to different speakers). This is important because the end goal is to add these segments to the training data, so each segment should be reasonably short and contain speech only from one speaker. Also, clustering by speaker is important for correct Speaker Adaptive Training.

Reference transcripts are normalised with the same tools that are used for language model training corpus preparation. Some obvious garbage and punctuation are removed, and all words are lower-cased. Numbers and dates are converted from digits to words.

Audio segments are then processed by ASR, that uses LSRC-nnet2-smbr acoustic model (section 3.4) and WebNews-LM-3 language model (section 0). Clustering information is used during recognition for speaker adaptation.

After all these processing steps, from each video file the following files are obtained:

- A corresponding inaccurate reference transcript in normalised form.
- A set of short audio files that roughly correspond to separate utterances. This set is sorted in chronological order and clustered into different speakers.
- A raw ASR transcript for each short audio file.
- Word alignment information (when each word is pronounced and length of pronunciation) from ASR for each audio file.

3.6.2. First Alignment

Next, each inaccurate reference transcript file is aligned with the per-utterance ASR transcripts. Similarly to Panayotov et al. (2015), the Smith-Waterman alignment algorithm (Smith & Waterman, 1981) is used for text alignment. An example of such alignment is shown in **Figure 9**. The grey boxes represent boundaries between different utterances obtained by the LIUM SpkDiarization toolkit.

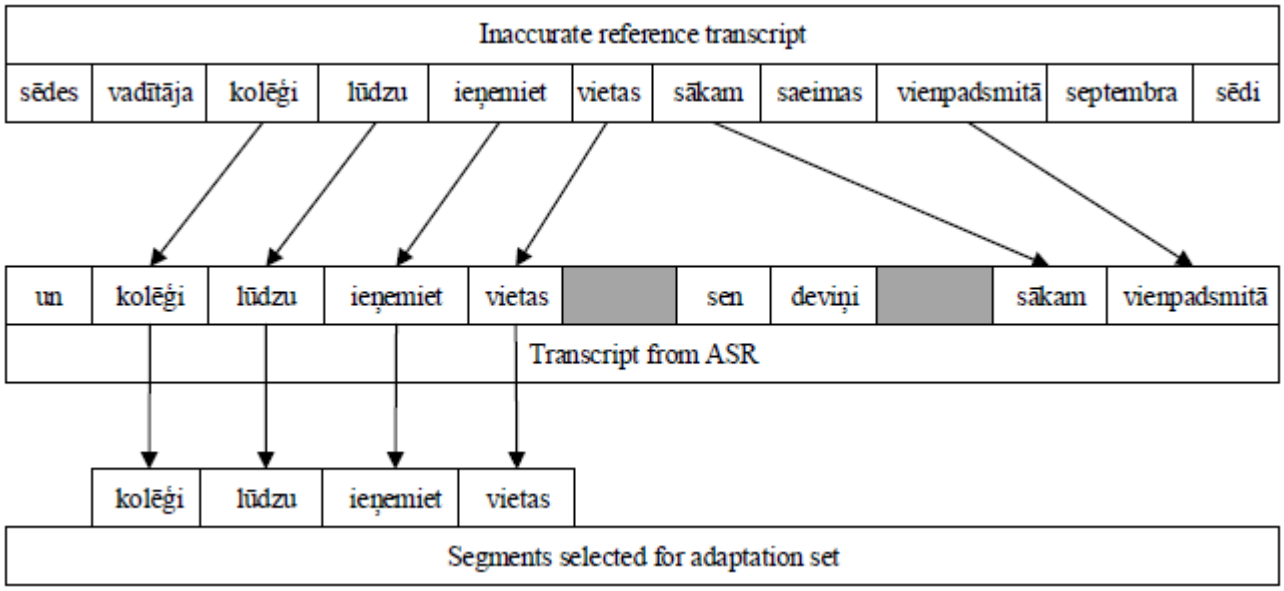

Figure 9. *Alignment between the reference transcript and ASR output.*

After the alignment, in each utterance, continuous sequences of matched words that are longer than some threshold (e.g. 3 in **Figure 9**) are selected. Using word alignment from ASR, these word sequences are extracted from utterance audio files and are added to the new training data set together with their transcripts from ASR. Speaker diarisation is also preserved, so that utterances from the same speaker are more or less grouped together.

A length threshold is needed to filter out possible alignment errors, for example, short word sequences like "un tas ir" ("and that is"), "un ir" ("and is"), etc. are rather frequent and can be either misrecognised by ASR or misaligned. Also, it's assumed that ASR word alignment for longer sequences is more accurate, so extracting longer sequences is less likely to cut off word beginnings and endings. A threshold of 5 or more consecutive words is used in this work in the first alignment step.

3.6.3. Second Alignment

Word sequences extracted in the first alignment step can already be used for training acoustic models. However, the improvement from adding these sequences to the training data will be limited because existing ASR already recognised them correctly; they already match the acoustic model quite well. The parts that were not recognised accurately (and not aligned) can be much more useful, as they are examples of when the existing acoustic model is not good enough.

Successfully extracted segments could be used as "anchor points" so that the audio between anchor points will be mapped to the text between anchor points. This mapping then can be used to help ASR to recognise this part correctly and produce a better alignment.

However, before that, there is an optional intermediate second alignment step that is needed to

improve the alignment and get more anchor points. The data extracted in the first alignment step is appended to the training data, retrain the acoustic model (only the DNN part of the acoustic model is retrained) and repeat decoding and alignment. Again, the data is extracted and appended to the baseline training set. The model is retrained again.

Next, the extraction threshold is relaxed from 5 consecutive words to 22 consecutive phones. The number 22 represents the average length of 3 average words in Latvian and is calculated on vocabularies of general domain ASR for Latvian. This less strict threshold creates more anchor points.

3.6.4. Pseudo-Force Alignment

After obtaining mappings between misrecognised audio segments and reference text, the natural choice would be to perform a classic force alignment. However, in our case, the reference text is not 100% accurate, so classic force alignment won't be able to produce a good alignment. Instead a pseudo-force alignment step similar to (Hazen, 2006) was performed.

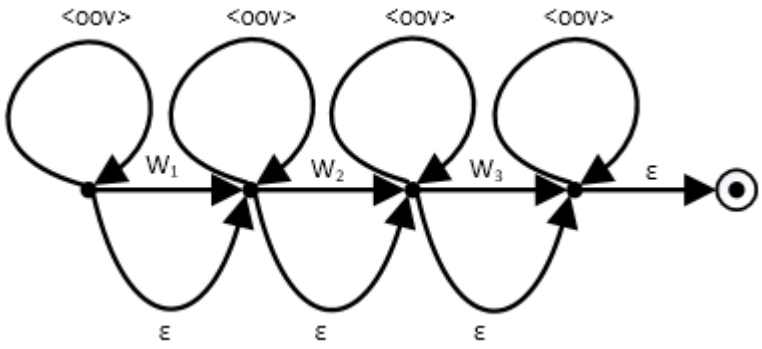

***Figure 10**. Example FSA for the pseudo-forced alignment of the word sequence w_1, w_2, w_3 with insertions, deletions and substitutions allowed.*

Previously matched and extracted segments are used as "anchor points" so that the audio between anchor points is mapped to the text between anchor points. In the case of edited transcripts, reference text can contain long regions of insertions and audio can contain long non-speech regions, so there are limits for text and audio length. Mappings that are too long are filtered out.

In pseudo-force alignment, it's assumed that errors in the transcript are possible, therefore insertions of new words and substitutions for existing words are allowed. Deletions are also allowed. This process is realised through the composition of a pseudo-forced alignment finite state acceptor (FSA) with a lexical transducer from the baseline ASR. An example alignment FSA that allows insertions, substitutions and deletions is shown in **Figure 10**.

Insertions are modelled through the use of an out-of-vocabulary (OOV) word filler model. In Latvian ASR with have 2 filler models, silence model and garbage model. Both are single phone models with 5 HMM states. Garbage model was used only during training to capture foreign words,

fragmented words and spoken noise. This model is used in pseudo-force alignment as OOV model.

After decoding the pseudo-force aligned audio segments, segments with a length of 22 or more phonemes are extracted and appended to the training data.

3.6.5. Evaluation

For evaluation two test corpora are used: (1) 1h EvalWebNews corpus, that was recorded using smartphones and contains recordings of non-professional speakers reading news from web pages and (2) a 1hr evaluation set from randomly chosen utterances from sessions of the current 12th Saeima (see section 3.25.3.1). Both corpora were manually annotated.

All downloaded 300 hours of the Saeima data were processed at once. As a result, 120 hours of aligned data were extracted after the first alignment.

An improvement in recognition quality was observed (see **Table 10**) after adding these 120 hours of Saeima recordings to the acoustic model training data. Experiments were performed using WebNews-LM-3 3-gram general domain language model (described in section 0). Word error rate was reduced from 19.6% to 17.0% (13% relative).

Table 10. *Evaluation of Latvian speech recognition on the general domain test set.*

	Training data, h	**WER, %**
Baseline (LSRC)	100	19.4
+ Saeima (1st step)	220	17.0
+ Saeima (2nd step)	249	17.0
+ Saeima (3nd step)	286	16.9

As described in the intermediate second alignment step, ASR improved by 120h extracted in the first step is used to repeat the decoding and alignment procedure. This allowed to improve the alignment, first to 136 hours and then, by changing the threshold to 22 characters, to 149 hours. However, no improvement in WER was detected in either case.

Because a large part of the data has been successfully decoded in the first two steps, only 57 hours of data were selected for pseudo-force alignment. 37 hours were successfully aligned and added to the training set, the WER of ASR trained on all of the data combined is 16.9%.

Table 11. *Evaluation of lattice oracle word error rates.*

	Training data, h	**WER, %**
Baseline (LSRC)	100	8.1
+ Saeima audio (1st step)	220	7.5
+ Saeima audio (2nd step)	249	7.4
+ Saeima audio (3rd step)	286	7.1

It was decided to minimize the effect of language model scores and calculate oracle word error rates on lattices obtained after decoding (see **Table 11**) as previous evaluation showed very small

improvement from pseudo-force alignment step,

Finally, because the data was collected from Saeima recordings, it was interesting to know if the recognition of Saeima sessions have improved. Only the best system trained on 296h was evaluated. Experiments were performed using Saeima-LM 3-gram domain adapted language model (described in section 5.3.1).

Table 12. *Evaluation on Saeima session transcription task.*

	Training data, h	**WER, %**
Baseline (LSRC)	100	8.6
+ Saeima audio (3rd step)	286	7.2

Results in **Table 12** clearly show that adding this automatically collected training data helps to significantly improve the recognition quality, even for high quality recordings of Saeima. WER is improved by 16.2% relative from 8.6% to 7.2%.

In the first and second alignment steps, segments of data that have a 100% match with the reference transcript are extracted. This means that existing ASR already recognises such segments correctly. Interestingly, after adding these segments to the training, an improvement in WER is seen.

This can be caused by (1) better senone coverage in the larger training set and (2) a large "language model bias" that allowed some segments to be recognised correctly even when the acoustic score was too low.

When doing pseudo-force alignment, this "language model bias" is even larger, so adding pseudo-force aligned data should improve word error rate even more. However, experiments showed only a small improvement. This means that only a small part of the pseudo-force aligned data is complementary to the data extracted in the first two steps.

For conciseness, the 186h of Saeima audio recordings obtained using method described in this section, will be called “Saeima11” in other sections of this thesis. The acoustic model trained on this corpus will be called “AUG1-nnet2-smbr” correspondingly.

To conclude, automatic data collection from the Web turned out to be very useful method for improving the acoustic models for Latvian speech recognition models and allowed to significantly improve the word error rate on different testing corpora. Importantly, there is still a room for future improvement as there exist many other sources of inaccurately annotated data on the Web.

3.7. TDNN Sequence Discriminative Acoustic Model

Increasing the amount of data enables training more complex models, that can better capture the long-term dependencies between acoustic events, which is important for accurate speech recognition.

LSRC-nnet2 acoustic models described in previous sections use a fixed small context window, that can be enlarged by scaling up the model input layer. However, this is undesirable as it makes model too large and too difficult to train. Also, despite being a feedforward architecture, computing the hidden activations at all time steps multiple times is computationally expensive. Thus, an acoustic model which can effectively deal with long temporal contexts without becoming too large is required.

Recurrent neural networks (RNNs) which use a dynamically changing contextual window over all of the sequence history rather than a fixed context window have been shown to achieve state-of-art performance on LVCSR tasks (Sak et al., 2014). However due to recurrent connections in the network, parallelization during training cannot be exploited to the same extent as in feed-forward neural networks.

Another neural network architecture which has been shown to be effective in modelling long range temporal dependencies is the time delay neural network (TDNN) proposed in (Waibel et al., 1989). TDNN networks are little bit behind RNNs in terms of recognition accuracy, but they require less data and are much faster to train and tune, so TDNN were chosen in this experiment.

Sequence discriminative training of neural networks has been shown to provide significant reduction in WER (Kingsbury, 2009; Su et al., 2013; Vesely et al, 2013). For this Kaldi uses the lattice-free version of the maximum mutual information (MMI) criterion LF-MMI, that can be used with GPU training.

To make such computation feasible they use a phone n-gram language model, in place of the word language model. To further reduce space and time complexity the computation is performed at one third the standard frame rate. Because of this conventional 3-state left-to-right HMM topology is replaced with 1-state HMMs that can be traversed in one frame.

Reduced framerate and the fact that LF-MMI maximizes the conditional log-likelihood of the correct transcript makes it in somewhat similar to CTC, but the difference is that in CTC the probabilities are locally normalized, but in MMI they are globally normalized.

TDNN acoustic model for Latvian are trained on augmented audio corpora using alignments and lattices obtained by *LSRC-Saeima11-nnet2-smbr*. Following Kaldi recommendations feature type is changed from 13 dimensional PLP to 40 dimensional MFCC. Similarly, to previous model, i-vectors are used for speaker adaptation.

3.7.1. Revised Grapheme-to-phoneme Modelling

Initial decisions on grapheme-to-phoneme modelling were made using stub LM and INT-GMM-8 models. It is possible that these decisions are not optimal for more complex acoustic models that work on much larger context and real large vocabulary language models.

So, it was decided to repeat some of the G2P experiments on LSRC-tri3b model (which is much faster to train) and check if previous decisions are still the most optimal. The experiments were performed on EvalWebNews data set using WebNews-LM-3 and Saeima-LM. The Saeima language model is used in this experiment because it's out-of-domain for EvalWebNews, so the importance of acoustic model for correct recognition is increased.

The following modifications to baseline G2P were evaluated:

- Using G2P algorithm from Latvian TTS by Goba & Vasiljevs (2007);
- Removing all "diphones" and modelling only letters;
- Replacing [c] grapheme-phoneme with [t]+[s] combination (because such separate phoneme does not exist);
- Replacing long vowels with two short ones (e.g. [ā] -> [a] + [a]).

The results in **Table 13** show that the optimal G2P is the simplest one which treats every letter as separate phoneme. The results are consistent for both language models used in this experiment.

Table 13. *Experiments on changing the baseline G2P*

G2P model	Lang_general, WER	Lang_saeima, WER
Baseline	35.9%	37.3%
TTS G2P	36.3%	38.2%
No diphones	**35.7%**	**36.0%**
No [c] phoneme	36.2%	36.8%
No diphones and no long vowels	36.2%	37.0%
No diphones and no [c] phoneme	35.9%	36.7%

3.7.2. Training Data Augmentation

In section 3.6 additional training data was collected by crawling Web and aligning audio recordings with imperfect transcriptions. Another way of getting more training data is by adding distorted or other way modified copies of the original data. Potential distortions can include:

- Adding various types of stationary noise;
- Adding artificial or real non-stationary noise;
- Adding volume and speed perturbation;
- Modelling acoustic of different environments through applying different room impulse responses;
- And many others.

Many of these distortions have been shown to effective. Some of them might not be easy to implement and databases of noises can be required, however it's still much cheaper than recording

and/or annotating new training data.

In recent years volume perturbation, speed perturbation and room impulse methods were shown to be particularly effective and easy to implement (Ko et al., 2015; Peddinti el al., 2016; Ko et al., 2017). Both can be implemented using standard signal processing algorithms and there are free and publicly available room impulse response libraries on the Internet.

The idea behind volume perturbation is the fact that usually speech training corpora contain recordings will normalized volume. Also, while it's easy to normalize volume when training and decoding the whole audio recording at once, in the online setup normalization will be always not accurate. Therefore, in order to make the acoustic models more robust to changes of volume and normalization inaccuracies, the training data is augmented with copies of the same recordings, but with randomly perturbated volume.

The idea behind speed perturbation is to better cover the variability of the pace of speaking by augmenting the training data with slowed down or speed up versions of the original utterances. This method should be particularly useful for smaller datasets, as bigger sets will be naturally more diverse with respect to speaking speed. However, it was shown this method can be effective even for bigger speech corpora (Ko et al., 2015).

Both perturbations are performed on the utterance level. Though in perturbation inside utterances might also be useful, as theoretically speakers can change the pace of speech and/or volume multiple time in the same utterance. It is an open research question if such perturbation will help to improve the accuracy.

Adding copies with different room impulse responses applied helps to simulate different rooms and environments. This method aims to make the acoustic model more robust to reverberation effects of different environments, this should also help in the scenarios where microphone is located at some distance to the speaker's mouth. One of features of this method is that, alignments from original data can be re-used, thus simplifying the training process.

Another way to distort audio samples in order to make ASR more robust is to train acoustic model on a mixed bandwidth data (e.g. 8kHz an 16kHz). This approach has been shown to improve both wideband and narrowband recognition (Seltzer&Acero, 2007; Li et al, 2012; Deng et al, 2013).

In this work, a publicly available database[2] of the simulated room impulse responses is used. This data includes simulated room impulse responses with various room configs, that were created by randomly sampling room parameters and speaker positions. This database was created for

[2] Availabe on OpenSLR site. http://www.openslr.org/26/

comparing the performance of acoustic models trained with data reverberated with real and simulated impulse responses (Ko et al., 2017).

3.7.3. Experimental Setup

First, volume perturbation was performed. The perturbation is performed on the original recordings without creating additional copies, so that amount of the training data is not increased. This is motivated by the fact that it has been shown that improvement from volume perturbation is almost non-existent (Ko et al., 2015), but by adding copies training time is significantly increased.

Next, speed perturbation is performed. Here 2 copies of each training utterance are created, one is slowed down to 90% speed and the other is speed up to 110%. As the result, the amount of training data is increased 3 times.

In order to simulate different environments and variation in distance between speaker and microphone a 1 copy of each volume and speed-perturbed utterance was created, and randomly chosen room impulse response was applied on each. As the result, the amount of training data is doubled. Moreover, a mixed bandwidth training is simulated by randomly applying lowpass filter to about the half of the data. The filter attenuates frequencies above 4kHz and emulates the recognition of upsampled narrowband audio.

To sum up, the following recipe for TDNN acoustic models was created:

- All available training data corpora (LSRC, LDSC, Saeima11) are combined into single data set (294 hours).
- 3-way speed perturbation of audio data (90%, 100% and 110% speed) to make model robust to different speech tempo.
- Applying simulated room impulse responses to make model robust to reverberation.
- Randomly applying lowpass filtering to about 50% of corpus to make model robust to narrowband audio.
- The total amount of training data is increased 6 times, from 294 hours to 1764 hours.
- Using pure grapheme-based pronunciation model (no diphones, one-to-one letter to phoneme mapping)
- Factorized TDNN with multisplicing and skip connections:
    - 40-dimensional MFCC vectors and 100 dimensional i-vectors as input.
    - 13 layers: input layer with LDA transform, 11 TDNN layers and 1 output SoftMax layer.

- ReLU activation function and BatchNorm.
- LF-MMI discriminative training.

The acoustic model trained using this recipe will be called "AUG2-TDNN" in this work. All experiments were performed using WebNews-LM-3 3-gram language model (described in section 0).

3.7.4. Evaluation

The effect of performing various training data augmentations was evaluated (shown in **Table 14**) on 2 test sets:

- EvalGeneral, to evaluate WER on good quality wideband (16kHz) audio.
- Narrowband version of EvalWebNews, to evaluate acoustic model robustness to mismatched audio conditions. For this a special version of EvalWebNews was created, where all frequency information above 4kHz was filtered.

Table 14. *Evaluation of models trained with data augmentation*

Data augmentation				**WER, %**	
Reverb	**Speed**	**Lowpass**	**Total size, h**	**EvalGeneral**	**EvalWebNews (narrowband)**
No	No	No	294	10.5	52.5
Yes	No	No	588	**10.0**	18.9
Yes	Yes	No	1764	10.1	17.0
Yes	Yes	Yes	1764	10.1	**13.9**

The results show that proposed data augmentation methods help in both matched and mismatched acoustic conditions. The best result on EvalGeneral is obtained by augmenting the training data by applying different reverberations (room impulse responses), adding other augmentation methods results in small increase of WER, but it's still lower than for non-augmented version. The improvement is especially significant in mismatched conditions, where each data augmentation method contributes to significant improvement of WER and the best result is achieved by performing all proposed data augmentations.

To sum up, results show that the data augmentation improves the accuracy and especially the robustness of the Latvian ASR. It seems that original training data is not enough capturing variability of different environments. This is particularly true for added 186 hours of Saeima data, because all of them were recorded in the same environment with the same microphones.

To conclude, the updated acoustic model (AUG2-TDNN) was compared to previous best model (AUG1-nnet2-smbr) on several testing corpora using WebNews-LM-3 3-gram general domain language model. The results in **Table 15** show that the new model significantly outperforms the previous model in all evaluations.

Table 15. *Comparison with previous best model*

Test corpus	**WER, %**	
	AUG1-nnet2-smbr	AUG2-TDNN
EvalWebNews	16.8	**10.3**
Saeima	10.4	**8.3**
LDSC-test	39.1	**17.1**
EvalWebNews(narrowband)	22.0	**13.9**

The difference is especially noticeable on LDSC-test and can be explained by the fact that AUG2-TDNN model is trained on LDSC. Section 5.2.1 describes an adapted acoustic model with the same feed-forward design as AUG1-nnet2-smb, which is trained for dictation task using LSRC and LDSC corpora. This model achieves WER of 35.7%, which is better than 39.1% by AUG1-nnet2-smbr and demonstrates the influence of LDSC data. Nevertheless, AUG2-TDNN outperforms this adapted system by more than 50% relative.

4. LANGUAGE MODELLING FOR LATVIAN

In first experiments with acoustic modelling for Latvian a "stub" language model was used, however for serious speech recognition experiments a real language model is required. A language model represents a system's knowledge of word semantics, which words are likely to co-occur, and in what sequence. In this work the language model should represent the general domain knowledge of Latvian language, that should enable continuous large vocabulary "free-form" speech recognition.

This section is based on author's work published in Salimbajevs & Pinnis (2014), Salimbajevs & Strigins (2015b) and Salimbajevs & Strigins (2015c).

4.1. Language Model Training Data

In order to train statistical language models, a monolingual text corpus is needed. Text corpora used in this work are listed in **Table 16**.

Table 16. *Sources of Latvian monolingual text corpora*

Corpus	**Size, sentences**	**Description**
WebNews corpus (version 2014)	44.5M	Automatically crawled collection of texts from Latvian news portals
WebNews corpus (version 2016)	46.6M	Automatically crawled collection of texts from Latvian news portals
Saeima corpus	1.3M	Official Saeima session transcripts

WebNews corpus is a collection of automatically crawled and automatically segmented text from biggest Latvian news portals. The crawled texts are not restricted to any particular domain, that means the corpus represents general domain. Because corpus is collected automatically, it can be periodically updated by rerunning crawling software. Two versions of WebNews corpus, 2014 and 2016, are used in this work.

For automatic speech recognition system adaptation task, transcripts of all previous Saeima sessions were automatically collected from Saeima webpage. Saeima transcript text corpus consists of 1.3M sentences and 21M words. The corpus is used to train domain specific language model for the adapted ASR system.

4.2. N-gram Language Model

After initial decisions on pronunciation model and initial acoustic model were made the next step was to replace the "stub" language model with an initial large vocabulary model.

In order to train statistical language models, a monolingual text corpus is needed. Initial statistical language models for Latvian were trained on WebNews corpus (version 2014).

Before training, the text corpus is pre-processed in the following way:

- First, the text was tokenised and split into sentences.
- Then, non-alphanumeric tokens, punctuation, URLs and other unrecognized tokens (tokens that contain both digits and characters, tokens using mixed casing) are filtered out.
- Next, number conversion from digits to words with correct inflection. For this, a module from the Latvian text-to-speech system was used (Goba&Vasiljevs, 2007). This module also expands some abbreviations like "km" to kilometres if they are next to some number.
- First word of each sentence was true-cased based on statistics from whole corpus. If word is more frequent in lowercased form, then it is lowercased, otherwise, it is left capitalized.

After pre-processing, the corpus consisted of 38.5M sentences (40% of them were 10-20 words long) and 592M running words. The vocabulary of the whole text corpus was 2.8M word surface forms.

From this large WebNews corpus, a smaller corpus of 3M (called WebNews3M) sentences was also created using the Moore & Lewis (2010) data selection method. Using this method, sentences from larger "out-of-domain" corpus can be selected based on their similarity to the sentences in the smaller corpus "in-domain" corpus. Here, above mentioned 38.5 corpus is used as out-of-domain corpus and training transcripts as in-domain corpus. As the result, 3M sentence corpus is created.

From both corpora, several 3-gram language models were trained. Because WebNews corpus is collected automatically it can contain spelling errors and various noise tokens. In order to filter these tokens and also make LM loadable (working with 2.8M vocabulary can require a lot of RAM and CPU) the vocabularies of different sizes were selected from the most frequent words in the corpus. To further reduce the size of language models, they are pruned to the same size of about 50 MB.

A recurrent neural network language model was trained on the same filtered small WebNews3M corpus using limited 50K word vocabulary. Unfortunately, experiments with larger vocabulary and larger training corpus failed – model was unable to converge and showed very poor results.

The models were combined with "INT-GMM-8" acoustic models and evaluated (see **Table 17**) in terms of perplexity and speech recognition WER on LSRC held-out set (described in section 3.2).

Table 17. *Experiments with 3-gram models*

Language model	Corpus	Vocabulary	Perplexity	OOV	WER, %
WebNews-LM-1	WebNews	50K	423.964	8.1%	37.5%
WebNews-LM-1-100	WebNews	100K	528.162	4.3%	48.5%
WebNews3M-LM-1	WebNewsSmall	50K	496.401	7.3%	41.7%
WebNews3M-LM-1-100	WebNewsSmall	100K	610.328	4.3%	45.5%
WebNews3M-RNNLM	WebNewsSmall	50K	n/d	7.3%	36.6%

The best result was achieved by RNN LM model with 50,000 words, however, the improvement is only 1% WER absolute and the model performs several times slower. Because of this and inability to train larger models with large vocabulary (also see Section 2.6.3) it was decided to leave neural network language models out of scope of this work and focus on N-gram language models.

The second-best result was achieved by a 3-gram model with the vocabulary size of 50,000 words. Interestingly, while the out of vocabulary (OOV) rate is smaller with larger vocabulary, the WER degraded. The idea of selecting sentences from the large corpus that are similar to the sentences in the acoustic training set transcripts turned out to be unsuccessful. Again, OOV rate is lower, but perplexity and WER degraded.

At the time when these experiments with initial speech recognition model were performed, the cause of these results was unknown. That lead to erroneous conclusion, that usage of smaller vocabularies is better. Later, it was discovered that the cause of this result is overly aggressive pruning procedure, which significantly degraded larger models.

4.3. Language Model Size

4.3.1. Language Model Pruning Problem

From literature (Huang et al., 2001) it is known that for English 3-gram models significantly outperform 2-gram model in speech recognition tasks. The improvement from 4-gram is usually not as big. It was decided to compare 2-gram models with 3-gram models for Latvian.

Table 18*. 2-gram vs 3-gram language model comparison*

N-gram order	Size, MB	RAM, GB	WER, %
2	50	1.33	36.58
2	100	2.21	35.94
3	50	1.68	37.39
3	100	3.48	**35.50**

Results of comparison are shown in **Table 18**. The evaluation was performed on the EvalWebNews test corpus using acoustic models LSRC-nnet2. The language models were trained

following WebNews-LM-1 recipe, but vocabulary size was increased to 200K and different pruning parameters were used for 100MB version. Surprisingly, the difference between 2-gram and 3-gram is very small and 2-gram model even outperforms 3-gram model when language model size is 50MB.

Next, the trigram hit rate was calculated on transcripts of EvalWebNews. Hitrate value indicates what percentage of words were scored using 3-grams, 2-grams or 1-grams entries in a n-gram language model. It was found that 3-gram hit rate from 100MB 3-gram model on the EvalWebNews corpus is only 9%. In other words, language scores of only 9% of words were calculated using 3-grams.

These findings can be interpreted as follows:

- The model is pruned so hard, that 3-grams are almost not used at all.
- It is possible that pruning process filters out too many "useful" 3-grams.
- The current language models are too small and are the main cause of high error rate.
- It is necessary to find a solution how to work with larger, preferably not pruned language models.

This result leads to a logical question: why so small models were used in the first place?

First reason is the size of the full model trained on WebNews corpus – about 20GB in ARPA format and vocabulary is 2.8M surface forms. Sphinx tools that prepare language model for Sphinx engine cannot handle model of such size. It is even bigger problem in case of Kaldi framework, the construction of decoding FST takes 13GB of RAM for 100MB model. So, it is necessary to limit the vocabulary and prune the model.

Second reason is the RAM usage during decoding. Using 100MB language model with Sphinx requires about 3.3GB of RAM during decoding and 3.6GB with Kaldi. The primary machine that was used for experiments had 4GB of RAM, so working with larger models was impossible.

The solution is to use two-pass decoding approach. In the first pass a small pruned language model (called "decoding" LM) is used during decoding to produce lattices (instead of single best hypothesis). Then, in the second pass, a large language model (called "rescoring" LM) is used to rescore these lattices.

Number of questions can arise when considering using this method:

- How is the accuracy and performance affected when using rescoring?
- What is the optimal size for smaller decoding language model?
- What is the optimal size for large rescoring language model?

- What is the memory usage during rescoring with large language models?

In order to answer these questions a number of experiments with models of different sizes were performed. 2-gram and 3-gram models were used for 1-pass and rescoring was performed with 3-gram models. EvalWebNews test corpus and LSRC-nnet2 acoustic models were used. The results are summarized in **Table 19**.

It can be seen, for example, that WER difference between decoding with 3-gram language model pruned to 1000 MB and using the same model to rescore lattices produced by 200 MB 3-gram model is only about 0.5 % absolute. However, the RAM usage is much smaller for 2-pass decoding: 6.65 GB in the 1st pass, and 0.62 GB in the 2nd pass. That is comparing to 30.59 GB when using 1-pass decoding. That illustrates that the 2-pass decoding is efficient method to work with large n-gram models and the reduction of the accuracy can be acceptable.

***Table 19**. Experiments with two-pass decoding*

Decoding LM		**Rescoring LM**	**WER, %**		**RAM, GB**		**Time, min**	
Order	**Size, MB**	**Size, GB**	**1st-pass**	**2nd-pass**	**1st-pass**	**2nd-pass**	**1st-pass**	**2nd-pass**
2	50	1	36.58	31.73	1.33	0.35	20:02	0:34
		2		30.03		0.67		0:40
		3		29.31		0.99		0:54
		4		28.8		1.28		1:05
		5		28.54		1.59		1:11
		11 (full)		**27.69**		1.99		0:54
2	100	1	35.94	32.41	2.21	0.35	20:25	0:34
2	200	1	35.77	32.41	3.28	0.35	20:46	0:34
3	50	1	37.39	32.07	1.68	0.38	21:47	0:34
		2		30.8		0.67		0:40
		3		29.9		0.99		0:54
		4		29.35		1.28		1:05
		5		29.05		1.59		1:11
		11 (full)		28.12		1.98		1:20
3	100	0.2	35.5	34.14	3.48	0.38	21:20	0:34
		0.4		32.91		0.38		0:40
		0.6		31.72		0.38		0:54
		0.8		31.59		0.38		1:05
		1		31.12		0.38		1:11
		11 (full)		**27.65**		1.98		1:20
3	200	0.4	34.06	31.78	6.65	0.62	21:59	0:40
		0.6		31.5		0.62		0:54
		0.8		31.29		0.62		1:05
		1		30.74		0.62		1:09
3	400	-	32.78	-	12.49	-	22:53	-
3	600	-	30.95	-	18.47	-	24:36	-
3	800	-	30.86	-	24.35	-	24:45	-
3	1000	-	30.23	-	30.59	-	25:47	-

Moreover, rescoring allows to outperform 1-pass decoding in terms of WER while using reasonable amount of RAM. For example, rescoring with full 11 GB unpruned language model consume about reasonable 2 GB of memory and achieve WER of 27.65.

Also, it can be seen that decoding with smaller language models and rescoring with large models is faster than 1-pass decoding with large language model. Actually, rescoring lattices takes only about 1 minute for given test corpus (EvalWebNews).

4.3.2. Higher Order N-gram Models

Introduction of 2-pass decoding allowed to use higher order N-gram models. Experiment on rescoring with unpruned 4-gram language model were also performed (see **Table 20**). Decoding was performed using 2-gram model pruned to 50MB and LSRC-nnet2 acoustic model.

Table 20. *Rescoring with 4-gram model*

Order of rescoring model	WER, %	RAM, GB
3	27.69	1.9
4	27.48	5.2

WER improvement was very small, however memory usage increased more than 2.5 times. Therefore, after analysing all results and trade-off between performance, WER and memory usage, the following configuration was selected for a baseline:

- A 2-gram model pruned to 100MB for decoding and producing lattices.
- A full not-pruned 3-gram model for rescoring lattices.

Both language models were trained on the WebNews monolingual text corpus with vocabulary limited to 200K most frequent words (surface forms). For shortness this LM is called WebNews-LM-2 in other parts of this thesis. To enable acronym recognition, an exception list containing acronyms and other similar tokens is added to the filtering and preprocessing steps. The updated and retrained language model is called WebNews-LM-2-ABBR.

Later, similar experiments were performed using 5-gram LM with improvements from Section 0 and TDNN acoustic models (unfinished version of AUG2-TDNN). The results in **Table 21** showed very little improvement on Saeima-test and no improvement on EvalWebNews.

Table 21. *Rescoring with 5-gram model*

Order of model		WER, %	
Decoding	Rescoring	EvalWebNews	Saeima-test
2	3	12.9	8.8
2	5	12.9	8.7
3	5	12.9	8.5
4	5	13.3	8.7

Additionally, oracle WER on lattices obtained in the decoding step was evaluated (see **Table**

22). Oracle WER shows the best possible WER that can be obtained by rescoring these 1-pass lattices.

Table 22. *Lattice oracle WER*

Decoding LM order	EvalWebNews	Saeima-test
2	5.9%	9.4%
3	6.0%	9.3%
4	6.3%	9.6%

Results show that all tested configuration are almost equal (except 4-gram decode, 5-gram rescore combination which is little bit worse). We can conclude that the previously chosen configuration is still optimal and allows to achieve low WER while consuming less memory during 1-pass decoding.

4.4. Automatic Spell-checking of the Monolingual Corpus

During previous experiments, it was also found that ASR can sometimes output misspelled or non-existing words. This happens because the corpus that is used for training language model, is automatically crawled from the Web. It contains "noise": misspelled words, words in other languages, code and "garbage". Some of this noise is already filtered, but some cases are very difficult, like misspellings or non-existing words, that look just like regular words.

One possible solution is to use spell-checker. For example, one could use spell-checker to correct spelling errors in training sentences and filter out sentences that cannot be repaired. In practice, however, spell-checker is not perfect and can introduce new errors, mark correct words as errors (because they are not in spell-checker vocabulary) etc. This could lead to removing too many good sentences and words and training a weaker model.

The proposed solution is to filter out sentences that contain N or more errors using automatic spell-checker, therefore several experiments were performed to find the best filtering strategy, different vocabularies were also tested.

The experiments were performed using WebNews corpus (version 2016), which was prepared by crawling news web articles. The acoustic model used was LSRC-nnet2 and language modelling procedures for this experiment are as described in 4.3 for WebNews-LM-2.

The goal of these experiments was to find the best filtering strategy for training corpus of decoding 2-gram model. Two different types of vocabularies were tested: (1) small 200K vocabulary and (2) large 1M vocabulary. Both vocabularies were created by taking most frequent words in the corpora. The idea is that by taking subset of most frequent words, misspellings and noise are filtered out. Also, it should be noted that after filtering out all sentences that contained words unrecognized by the spell-checker, the vocabulary is reduced to 900K. So, vocabularies larger than 1M were not evaluated.

Table 23. *Filtering LM training corpus with spell-checker*

Vocabulary size	Max allowed errors	Sentences after filtering	WER, %
200K	0	24M	34.54
200K	1	35M	34.84
200K	2	40M	35.81
200K	3	42M	35.86
200K	4	43M	36.32
200K	5	43M	36.28
200K	Unlimited	45M	36.58
900K	0	24M	**29.22**
1M	1	35M	30.67
1M	2	40M	31.35
1M	3	42M	31.43
1M	4	43M	31.60
1M	5	44M	31.73
1M	Unlimited	47M	35.25

The results of experiments are presented in **Table 23**. The best result (WER is 29.22%) was obtained when using large 900K vocabulary and not allowing any errors. Allowing errors resulted in gradual increase of WER. But using 200K vocabulary resulted in much higher error rate, because of increased out-of-vocabulary rate. Rescoring with 3-gram LM trained on the same filtered corpus, allowed to achieve significant WER improvement, from 27.69%, to 23.61%.

Even better result was achieved by performing discriminative SMBR training of acoustic models and adding adaptations for acronyms by providing an exception list for the spell-checker and other filters, WER was reduced to 20.31%.

In previous experiments the same filtering strategy was used for both decoding and rescoring language models. This was done for several reasons:

- Experimentation with 2-gram model is more interactive as results are available much quicker.
- Vocabularies are automatically synchronized if the same strategy is applied when training 3-gram LM.
- There was a hypothesis that, applying the same strategy to 3-gram LM will also be optimal.

The next batch of experiments verifies last hypothesis and also tests new idea: allowing language model training corpora to have sentences with 1 error (as marked by spell-checker), but later limiting vocabulary to words that are recognized by spell-checker only. The intuition behind this idea is that, when filtering whole sentence because of only one error, useful information is lost. So, finer filtering can be achieved if such sentences are left in training corpus, but after training vocabulary is limited only to words that are recognized, i.e. only n-grams that are recognized as correct are selected. Results of this batch of experiments are presented in **Table 24**.

Table 24. *Different filtering strategies for 2 and 3-gram models*

Max errors in 2-gram training corpus	Max errors in 3-gram training corpus	Max errors in vocabulary	WER, %
0	0	0	20.31
1	1	1	21.33
0	1	0	19.76
1	1	0	**19.63**
2	2	0	20.87

The results support the hypothesis when no special vocabulary filtering is used, however when we filter vocabulary separately, different strategy achieve the best WER. Allowing 1 error in sentences during 2-gram and 3-gram training, but allowing no errors in vocabulary enabled to achieve WER of 19.63%. Increasing number of errors resulted in increase of WER, so it can be concluded that optimal value for spelling error filtering is 1.

As a by-product, processing with spell-checker also allows to perform true-casing. All words are lowercased and checked by the spell-checker. If after lowercasing the word is marked as error, then it is changed back to the initial form. True-casing is performed without context, so spell-checker does not know if word is first in the sentence. There is also a list of exceptions, like acronyms, for which lowercasing is never performed. Applying this method instead of corpus statistics-based method in 4.1 allowed to make a small WER improvement from 19.63% to 19.46%. The resulting language model is called WebNews-LM-3.

4.5. Sub-word Language Model

N-gram models treat inflected forms of the same word as separate independent words and does not recognize their similarity. For a closed vocabulary system, this means:

- If an inflected form is not presented in the training corpus, then it will not be recognized correctly.
- The full vocabulary of such a LM will contain about a million or more surface forms. The number of n-grams will be more than 200 million for 3-gram model. This leads to high memory and computational resource requirements (e.g. section 4.3) or to high OOV rates and increased perplexity if vocabulary is limited or model is pruned.
- Estimation of model of this size requires a huge amount of training data in order to get reliable probability estimates for all possible surface forms.

One of possible solutions for inflected form modelling is decomposition of whole words into smaller units, called sub-words. These smaller units are selected to be common for a large number of words. Sub-word based search vocabularies and language models can reduce the OOV rate of a speech recognition system (Salimbajevs & Strigins, 2015c).

Using sub-word vocabulary requires the following steps to be taken:

- **Decomposition**: The original words need to be decomposed into smaller sub-word units. The units need to be common for many words, so that the new sub-word vocabulary size is clearly smaller than that of the whole word vocabulary.
- **Pronunciation Generatio**n: In this step, sub-word unit pronunciations are being added to the speech recognition engine. In general, deducing the pronunciation of a sub-word unit from the pronunciation of a whole word is often challenging and even impossible in some cases. However, Latvian has a strong correspondence between written form and phoneme sequence, and this makes it possible to use a grapheme-based approach in this step.
- **Language Model Training**: A new language model needs to be trained for recognition of sub-word units. A model is usually trained on the same text corpus that was used for deriving the vocabulary.
- **Word Reconstruction**: After decoding, the recognized sub-words need to be recombined in order to obtain a valid word sequence.

4.5.1. Word Decomposition

One approach to decomposing words into sub-word units is to use probabilistic machine learning methods. In this work, *Morfessor* 2.0 is used (Creutz & Lagus, 2005; Virpioja et al., 2013) – a family of methods for unsupervised learning of morphological segmentation. The *Morfessor* model is trained on a text corpus, and then this or another corpus is segmented using this model. The result is a corpus made from sub-word units, which can be used to train an n-gram language model and derive a vocabulary of noticeably smaller size and which has almost zero OOV rate on original WebNews (see **Table 25**).

***Table 25**: OOV rate comparison*

Method	**Size**	**OOV, %**
Baseline	100K	11.2 %
Baseline	200K	8.7 %
Baseline	400K	7.2 %
Baseline	900K	5.9 %
Morfessor	76K	**<0.01 %**

Other methods include: modified stemmers, byte-pair encoding (Sennrich et al, 2016) or morphological analysers, that can split words into grammatically correct morphemes. In this work, un addition to Morfessor author also test byte-pair encoding approach (BPE). This method is based

does not try achieving morphological segmentation and instead relies on a data compression algorithm (Gage, 1994). This method represents the most frequent words with single units, but less frequent words are represented by combining multiple more frequent units (morphemes, syllables, characters). For example, frequent word "and" is encoded as single unit "AND", but some rare word like "solander" can encoded as "sol + and + er".

Both approaches are tested in this work because of the following reasons:

- Stemmer approach have been shown to be less effective (Salimbajevs & Strigins, 2015c).
- Morfessor and BPE showed very similar results for other morphologically complex languages (Smit et al., 2017).
- Both Morfessor and BPE are purely data-driven and there is no need for complex proprietary morphological analysers which can do segmentation. Also, it is not clear if such grammatically correct segmentation is useful for speech recognition.

4.5.2. Word Reconstruction

The output of the sub-word speech recognizer will be a sequence of sub-word units, thus, regardless of the chosen method and units, it is important to be able to reconstruct words from the sub-words to produce readable text.

Table 26. *Four methods of marking sub-word units so that the original word sequence can be reconstructed*

Marking style	**Example**
Boundary tag	<w> kā <w> pie dzīv ojums <w>
Left-marked	kā pie +dzīv +ojums
Right-marked	kā pie+ dzīv+ ojums
Left+right-marked	kā pie+ +dzīv+ +ojums

After word decomposition is important to mark the boundaries between words. This can be done by introducing a word boundary tag (like <w>) or marking sub-words (by adding prefixes and/or suffixes) to indicate their position in a word (examples shown in **Table 26**). All these markings satisfy the requirement that the word text can be reconstructed in a trivial manner.

The choice of the marking style is not just a matter of taste, because it affects the efficiency of the ASR system. For example, using word boundary tags increases the number of tokens in a sentence and requires a higher n-gram order in language modelling. Also, a default ASR decoder does not produce word boundary tags, as they don't have any pronunciation. However, using the marked sub-word approach increases the vocabulary of the ASR but has less tokens in the segmented sentences.

Tags and markings can be restored in a raw output of ASR by using a separate hidden-event

language model (Stolcke et al., 1998). This model is trained on a corpus in which the places where the words were decomposed are treated as "*hidden events*" and are marked accordingly. Applying it to a sequence of sub-word units produces a sequence of the most likely sub-word units and connector tags from which full words can be reconstructed.

However, a much better results can be achieved if sub-word reconstruction is integrated directly into decoding progress (Smit et al., 2017). This method has at least two major advantages:

1. Pronunciation of sub-word units is modelled using correct context-dependent phones (which can be difficult to implement if reconstruction is performed separately from recognition).
2. Decoder is forced to produce valid output sequences. Systems that do not apply these restrictions will not be able to make an unambiguous decision how to reconstruct words from the sub-word sequence. This also improves the performance by making the search space smaller.

Smit el. al. (2017) performed a comparison between different styles of sub-word marking and found, that (1) difference between different marking styles is very small, (2) for all but one experiment the "left-right" style of marking sub-words was the most effective. Considering these results, in the following experiment left-right markings are used.

4.5.3. Evaluation

A Morfessor 2.0 tool and BPE methods were used to segment filtered WebNews corpus (section 0) and extract sub-word vocabularies with left-right markings. For Morfessor this resulted in vocabulary of 234K units and 130K sub-word units for BPE. Each word in WebNews corpus was split into 2 sub-word units on average.

Next, in order to evaluate the influence of sub-word vocabularies on the speech recognition task, the following procedures were performed:

- A 6-gram language model was trained from sub-word segmented and marked WebNews corpus. A higher order n-gram model is used because after splitting a single word can be longer than 1-gram. In these experiments the average splitting factor for both methods was about 2, so a 6-gram sub-word LM was chosen to catch approximately the same context as 3-gram word LM.
- From this large two smaller language models were created, 3-gram model for decoding and 6-gram model for rescoring. Both models are pruned, so that the total n-gram count is similar to baseline 2-gram and 3-gram models.

- Using above mentioned 3-gram model and AUG2-TDNN acoustic models the decoding graph for sub-word recognition is created similarly to Smit el. al. (2017).
- A WebNews-LM-3 language model with AUG2-TDNN acoustic model is used as baseline.

The evaluation was performed for non-adapted general domain ASR systems. During evaluation baseline full word, Morfessor and BPE sub-word ASR systems were compared with each other on EvalWebNews, EvalGeneral and Saeima-test test corpora.

Table 27. *Evaluation of general domain sub-word ASR*

Test corpus	**WER, %**		
	Baseline	**Morfessor sub-words**	**BPE sub-words**
EvalWebNews	10.3	10.2	**10.1**
EvalGeneral	10.4	9.8	**9.6**
Saeima-test	8.3	7.9	**7.8**

The evaluation results presented in **Table 27** show that sub-word ASR performs better than baseline full word ASR. Also, it can be seen that BPE sub-word approach outperforms the Morfessor based approach.

Next, the performance of sub-word recognition was evaluated (see **Table 28**). The results show that, because sub-word language models use higher order n-grams the decoding a graph is more complex, thus the decoding performance is slower. However, the difference is minimal and rescoring time remained unchanged. Therefore, it can be concluded that sub-word approach for Latvian ASR is practical and allows for WER improvements.

Table 28. *Evaluation of sub-word ASR performance*

Test corpus	**Process**	**Baseline ASR**	**Sub-word ASR**
EvalWebNews	Decoding	**184 seconds**	189 seconds
	Rescoring	16 seconds	16 seconds
Saeima	Decoding	**560 seconds**	586 seconds
	Rescoring	21 seconds	21 seconds

Finally, a small OOV analysis was performed on EvalWebNews. This test set contains 36 words that are not in the WebNews (version 2014) corpus, so they can not be recognized by baseline ASR. Sub-word ASR correctly recognizes 14 of 36 (38.9%), therefore it can be concluded that sub-word approach is able to partially overcome sparsity created by inflected forms and improve the speech recognition quality.

5. DOMAIN ADAPTATION

5.1. Latvian Speech-to-Text Transcription Service

Although there exist many services and software that transcribe speech to text for widely used languages, such as English, German, etc., there is were no such service for the Latvian language before 2015 (also, it must be noted that in August 2017 Google released speech recognition for Latvian). One explanation for this might be that Latvian is spoken by only about 1.5 million people, which becomes a reason why big companies, who actively research and make ASR applications available for the public, do not develop ASR systems for Latvian. It should also be noted that Latvian is an under-resourced language, and there is not much language and acoustic corpora available for Latvian, for example, LSRC was created only in 2014. Another reason could be the fact that Latvian is a very different language and reusing models and methods from "big" languages is impossible.

In spite of these difficulties, there is quite strong motivation to develop a good performing LVASR system for Latvian, as it would allow people to use it in various areas, such as business, education, security, etc. So, after Latvian ASR achieved sufficient quality (WER of 19.4% on EvalWebNews, using WebNews-LM-3 and LSRC-nnet2-smbr models), the development of Latvian Speech-To-Text transcription service quickly started and in 2015 the service was published online (Salimbajevs & Strigins, 2015a).

The main purpose of this service was to enable regular users to automatically transcribe their speech recordings in Latvian, which was impossible before. The service accepts most of the popular audio formats and provides the transcription in a plain text or in a subtitle format (i.e., text with time marks). This service allows to test the Latvian LVASR system in some real-life scenarios, which is valuable both for potential future users of this technology and for developers, because it gives useful information about what people need to transcribe, which will help to develop more optimized speech recognition systems.

The transcription service is based on the Alumae Full-duplex Speech-to-text System for Estonian (Alumae, 2014) and Kaldi speech recognition toolkit. It consists of a web frontend, a single master server, and multiple workers, which can be used independently. For example, workers and master servers can be hosted on different machines. This makes it possible to easily scale up the system just by adding more servers and hardware. It is also possible to use the transcription service without the web frontend, i.e., create a desktop or mobile application that communicates directly with the master server. An overview of the service is presented in **Figure 11**.

The web frontend consists of two HTML\JS web pages. The first page enables the user to select a file, enter an email address, and submit this data to the transcription service. The other page is used

for viewing and downloading transcriptions. It also contains a feedback form for evaluation of user experience.

The master server is responsible for receiving files from users, converting them to a uniform format, and maintaining a job queue (one job represents one file). The master server is also responsible for sending notification e-mails about the transcription process, e.g., to notify the user that his file is successfully processed, and the transcription is ready.

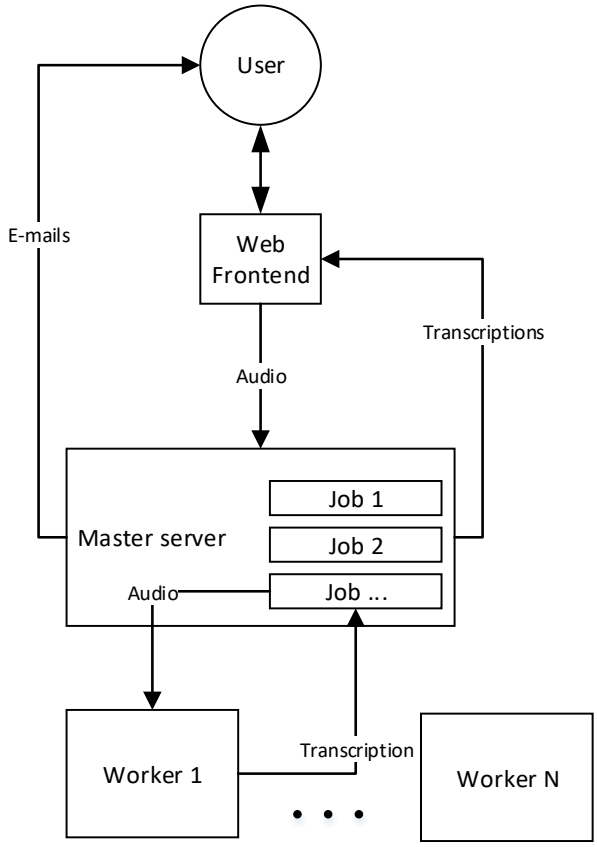

Figure 11. *Transcription service overview.*

Workers connect to the master server, receive jobs, and perform the actual transcription. Each worker processes only 1 job at a time. Each job consists of several stages:

- Segmentation and speaker diarisation;
- PLP feature and i-vector extraction;
- Decoding with 2-gram model;
- Lattice rescoring with 3-gram model;
- Converting end result to SRT and CTM files, which enhance the transcript with information on sentence and word level.

The transcriptions are then sent back to the master server where they can be accessed by users through the special URL that is sent in a notification e-mail.

LIUM SpkDiarization tool (Meignier & Merlin, 2010) is used for speaker diarisation, which is the process of partitioning an input audio stream into homogeneous segments according to the speaker identity, or in other words a combination of speaker segmentation and speaker clustering. The first aims at finding speaker change points in an audio stream. The second aims at grouping together speech segments on the basis of speaker characteristics. It can enhance the readability of an automatic speech transcription by structuring the audio stream into speaker turns.

The multimedia format support is achieved by using GStreamer multimedia framework[3], the developed web service is able to process all popular audio coding formats and also, in many cases, extract audio from popular video formats.

Because acoustic models were trained using Kaldi framework, it was logical to customize existing Kaldi speech decoder instead of implementing new solutions.

Not all recognition scenarios consist of processing long audio recordings, so the webservice also supports a "simple" job mode:

- Speaker diarisation is not performed.
- No email notifications are sent.
- CTM and SRT files are not generated, but the recognition is sent as a JSON response to the original request.

This mode is intended for short audio files and synchronous usage, e.g. for user interface purposes. For example, it used to recognize user answers in a mobile app "Reizrēķins", which teaches kids multiplication.

The first deployed ASR system consisted of LSRC-nnet2-smbr acoustic model and WebNews-LM-3 language model. The system showed WER of 19.46% on EvalWebNews test set. Analysis of errors shows that only 47% of misrecognized words in these test sets make utterances difficult or impossible to understand. E.g., 42% of errors are errors in word endings, and in most cases, it is easy for a human reader to recover correct meaning from such errors.

While these results were very far from perfect, that allowed any Latvian language speaker to try the speech recognition technology for their language for the first time and enabled the development of real-word practical applications. This led to the development of several domain adapted systems and mobile applications like (Balss[4] and Reizrēķins[5]).

[3] GStreamer. Open source multimedia framework https://gstreamer.freedesktop.org/
[4] Mobile app Tildes Balss. https://play.google.com/store/apps/details?id=com.tilde.tildesbalss&hl=en
[5] Mobile app Reizrēķins. https://play.google.com/store/apps/details?id=com.tilde.male.model.one&hl=en

Following the development of better acoustic models like AUG2-TDNN the Latvian Speech-to-Text transcription service was updated and WER on EvalWebNews was reduced to 10.3%

5.2. Dictation Task

Results described in Section 5.1 allowed to create first publicly available large vocabulary speech recognition service for Latvian, which can be used in some practical applications like speech recording indexing, preparing draft versions of speech transcripts and mobile applications with natural language interface. 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.

The first problem was solved by creating Latvian Dictated Speech Corpus (LDSC) in a joint effort of Tilde and Institute of Mathematics and Computer Science of University of Latvia (Pinnis et al., 2016). The goal of this work was to create a specific corpus that would: 1) better capture the speaker characteristics when dictating text to a computer and 2) contain spoken commands common to dictation scenarios (e.g., punctuation, formatting, special symbol, and action commands). The more details on this corpus are given in Section 0, which is based on publication by Salimbajevs (2016b).

The creation of the dictation corpus opened the possibility for tackling remaining tasks, which will be the main focus of next subsections.

5.2.1. Acoustic Model Adaptation

HMM-DNN acoustic models LSRC-nnet2 and LSRC-nnet2-smbr were used as a baseline in adaptation for dictation task.

In the dictation task, the ASR is expected to perform in real-time, generating a hypothesis as the audio is streaming. This means that the acoustic model should not be too complex and should be adapted for streaming data.

Fortunately, both LSRC-nnet2 and LSRC-nnet2-smbr were already reasonably small and trained with sliding window cepstral normalization and i-vector adaptation. If this would not be the case, the model should have been simplified (for example, by reducing number of hidden layers or neurons in them) and re-trained using sliding normalization.

5.2.2. Language Model Adaptation

For language model training, a 46 million sentence text corpus WebNews (version 2016) is used.

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.
- 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.
- Spell-checker is also used for true-casing.

As the result, the processed corpus contains 35M sentences, 674M tokens. On this text corpus, two n-gram language models are trained in the same way as described in 0, vocabulary is limited to 872K words (extracted from sentences containing no spelling errors):

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

This set of dictation language models is called "WebNews-Dict-LM" in the other parts of this work.

5.2.3. Dictation Software

Latvian Speech-To-Text Transcription Service (see section 5.1) was upgraded in order to integrate the dictation service. For this, new type of workers and new API were implemented. Dictation service API is based on WebSocket protocol and is more similar to Full-duplex Speech-to-text System for Estonian (Alumae, 2014).

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 one 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.

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 are 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 by Moattar & Homayounpour (2009).

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 & Homayounpour (2009).

5.2.4. Performance and Memory Usage

The effect of the rescoring optimizations is evaluated 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.

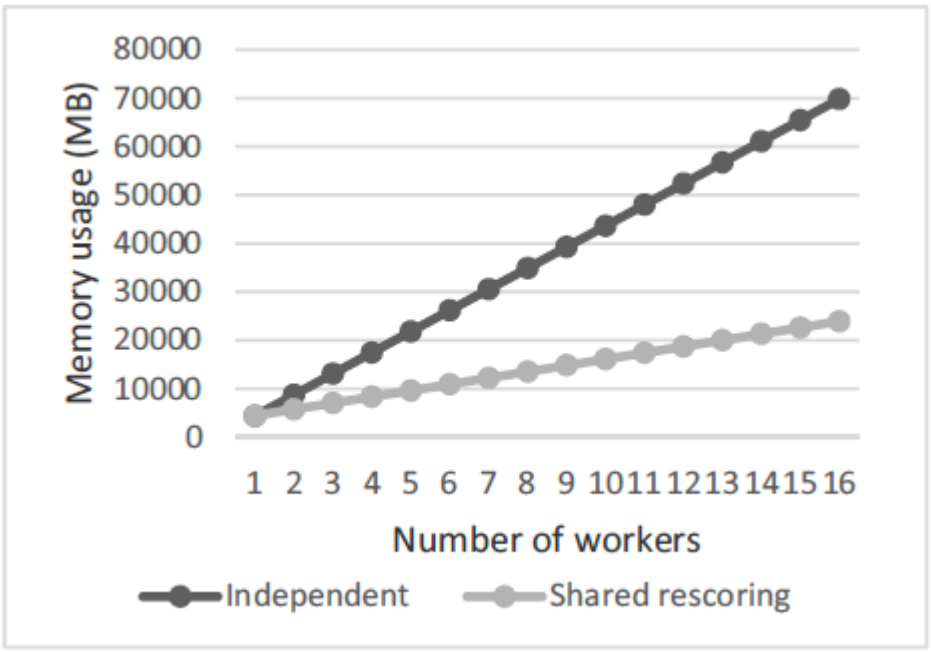

Figure 12. *Idle memory usage.*

Figure 12 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 rescoring method. Thus, a much simpler hardware can be used.

Memory usage during decoding was also evaluated. **Figure 13** shows peak memory usage when simultaneously performing 1-16 decoding jobs of the same speech recording. The numbers reported represent the amount of memory used for decoding only, excluding memory used for storing models. Results show that the memory usage for shared rescoring method is similar to independent rescoring.

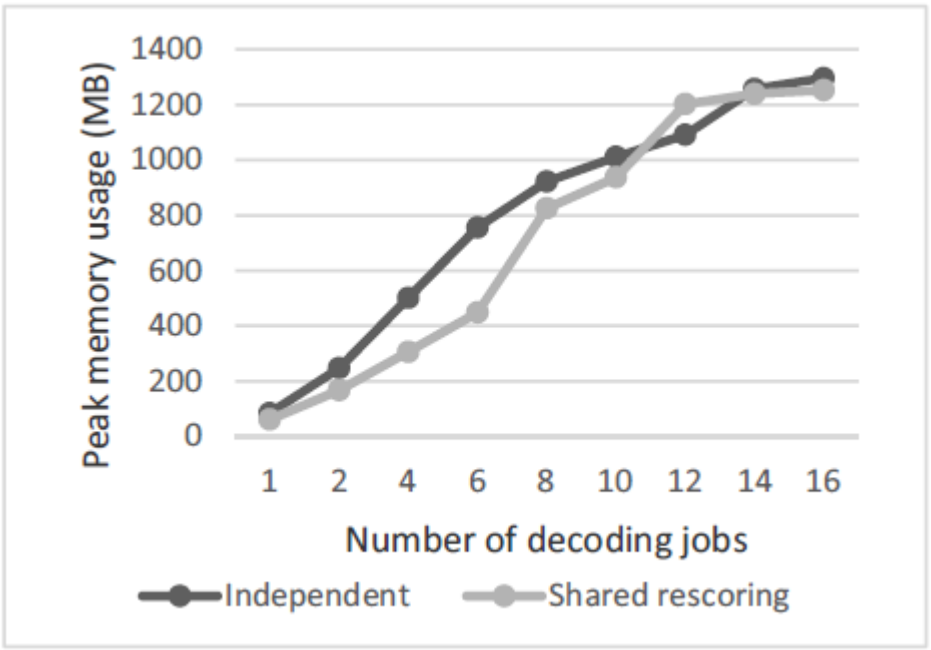

Figure 13. *Peak memory usage during decoding.*

Finally, the effect on real-time performance is evaluated (see **Figure 14**). 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 $61/60 \approx 1.02$.

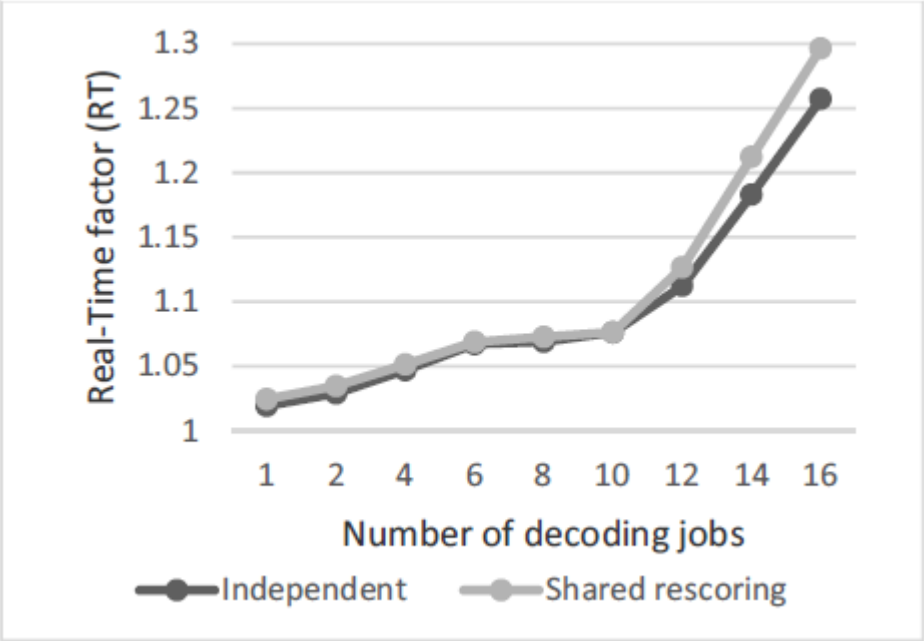

Figure 14. *Real-time performance evaluation.*

Figure 14 shows that the difference between independent workers and workers with shared rescoring is very small and only starts to appear when 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.

Also, it must be noted that the same file is streamed for all workers. That means all lattice rescoring requests happen at the same time, which can be interpreted as worst-case scenario for shared rescoring scheme.

5.2.5. Evaluation

This last subsection is devoted to evaluation of influence acoustic and language model adaptation on speech recognition accuracy.

First, the adapted HMM-GMM models were trained and evaluated on LDSC-test. The models were adapted by augmenting training data with 8 hours of dictated speech from LDSC. The same training procedure as for LSRC-tri3b was used. The results in **Table 29** show that augmenting training data with 8 hours of dictated speech allows to reduce the WER by relative 17% (or 7.36% absolute)

comparing to non-adapted LSRC-tri3b based system.

The effect of the special text corpus processing was also evaluated. Without this adaptation, it is difficult for both systems to recognise correctly commands and the resulting WER is very high. Even in the case when the adapted acoustic model is used, the WER still is 35.71%. By using the special text corpus processing alone, the WER can be improved by relative 30.69% (or 13.22% in absolute numbers). However, in this case, the improvement from using DSC in acoustic training is much smaller; the WER is reduced only by 6.53% (or 1.95% absolute).

Table 29. *Evaluation using HMM-GMM models*

Training set	Language model	WER
Baseline (100hr)	Adapted	29.85%
	Not adapted	43.07%
Augmented (108hr)	Adapted	**27.90%**
	Not adapted	35.71%

As next, the existing general transcription system (based on LSRC-nnet2-smbr, without LM adaptation and trained only on the 100 hour long LSRC data set) was evaluated on the one hour LDSC-test. The resulting high word error rate (WER) of 40.7% indicates that there is a significant difference between dictation and transcription tasks.

Next, HMM-DNN models were trained using maximum likelihood criteria and a comparison with LSRC-nnet2-smbr based ASR was performed (see **Table 30**). As in previous case, special text corpus processing for LM training gives significant improvement – 32.93% relative (or absolute 13.4%). By augmenting training data with dictated speech, the WER is further improved by 12.57% (or 3.43% absolute).

Table 30. *Evaluation using HMM-DNN models*

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

If this result is compared to a non-adapted LSRC-nnet2-smbr based system, then the overall improvement from both language model adaptation and acoustic data augmentation is at relative 41.36% (or 16.8% in absolute numbers). It is noticeable that there is a relatively small difference between baseline HMM-DNN and baseline HMM-GMM systems without LM adaptation. This is the result of a mismatch between training and testing conditions. However, when both LM adaptation and additional training data is used, the difference between HMM-DNN and HMM-GMM becomes large – 14.48% relative improvement (or 4.04% absolute).

Experiments with discriminative training were also performed, but no improvement were observed. On the contrary, the WER degraded by approximately 10% when compared to a non-discriminatively

trained system. For general domain transcription system WER difference on LDSC-test between LSRC-nnet2 and LSRC-nnet2-smbr was minimal, less than 0.1%. This is probably caused by the fact that the size of LDSC is relatively small in comparison with the remaining training data (the LDSC contribution is approximately 7.5%). Therefore, the acoustic models get more adapted to the 100 hours of non-dictation speech data.

Overall, the both LM and AM adaptation allowed to significantly improve recognition quality in the dictation scenario. The achieved WER of 23.9% shows that the adapted ASR is applicable for usage in the real-world applications, so the developed dictation ASR system was deployed on Latvian Text-to-Speech server.

Later, an improved dictation system was deployed, this system uses AUG2-TDNN acoustic model (Section 3.7) and achieves WER of 12.6% on LDSC-test.

5.3. Saeima Session Transcription

One of the domains in which Latvian speech recognition might be particularly useful, is transcription of debates, meetings and sessions. For instance, every session of Latvian Parliament, Saeima, must be transcribed and transcription should be made available for public as soon as possible. There are also many closed sessions, for which transcripts must be also manually prepared. This is a time consuming and tedious process that requires manual work of many people – transcription specialists and editors. Automatic speech recognition (ASR) can be very useful in this case because it can reduce the work of a human to only correcting recognition mistakes, adding punctuation and formatting. The ASR function can be complemented with automatic speaker diarization and punctuation restoration to further reduce the work.

This subsection presents a case study of domain adaptation of general domain Latvian speech recognition system (LSRC-nnet2-smbr with WebNews-LM-3) for transcription of Saeima sessions. However, the methods described here can be also applied to other domains.

5.3.1. Language Model Adaptation

In domain adaptation task, the first step is to get the evaluation set for that particular domain. Without such data set, it is impossible to prove that performed adaptations are actually helpful and give any improvements.

For this a 1-hour corpus from Saeima session recordings from 2014 to 2016 was collected and manually annotated. This corpus is called “Saeima-test” and described in Section 4.1.

General domain ASR achieved WER of 13.9% on this new test set. Such good result can be explained, by the fact that Saeima session recordings are usually high quality, speakers usually speak

in turns, microphone is directed and close to the speaker. It can be concluded, that general acoustic model is already suitable for transcription of Saeima session.

However, there should be still much room for adaptation of language model, as Saeima speakers use many specific language constructions, words and phrases that are not common in general speech, but are very frequent in the domain of Saeima speeches.

For this about 1M sentences from Saeima session transcripts were crawled from Saeima website (see Section 4.1). Collected corpus was processed in the same way as WebNews, except no filtering with spell-checker was performed, as Saeima transcripts are thoroughly checked before publishing and can be trusted. Two held-out sets (dev and test) were created from collected Saeima transcripts, each containing 10000 sentences, the rest were used to train 3-gram language model.

Table 31 shows the results of perplexity evaluation of 3 different language models on "test" held-out set. It can be seen that the best result is achieved by the interpolated model, which was created from 3-gram WebNews-LM-3 and 3-gram Saeima transcript LM by using second held-out set "dev" for finding the best interpolation weights.

Table 31. *Perplexity evaluation on held-out set of Saeima transcripts*

Language model (training corpus)	**Perplexity**
WebNews 3-gram	236.94
Saeima 3-gram	156.34
Interpolated 3-gram	126.56

After perplexity evaluation, the best model (interpolated) was split into two parts for integration into ASR:

- 2-gram pruned (to the size of about 120MB) LM for decoding;
- 3-gram unmodified LM for rescoring.

The set of this models is called "Saeima-LM" in other parts of this work.

The result of WER evaluation on 1hr Saeima evaluation set can be seen in **Table 32**. The adaptation was successful and WER was significantly reduced from 13.9% to 8.6%, which is 38% relative improvement.

Table 32. *Word error rate comparison between baseline and adapted systems*

ASR system	**WER, %**
Baseline	13.9
With adapted language model	8.6

This positive result allows to conclude that general domain speech recognition system was successfully adapted for the particular domain, transcription of Saeima sessions. The achieved recognition accuracy should be sufficient to make the work of Saeima session transcription specialists

much easier. When using AUG2-TDNN acoustic models instead of LSRC-nnet2-smbr the WER can be further improved from 8.6 to 5.9%.

5.3.2. Post-editing

Although there exist many specific audio transcription editors, Microsoft Word is a de-facto standard tool for preparing many types of documents. It is a familiar tool for millions of people. Moreover, in many scenarios, the final transcription document will be a Microsoft Word document. In case of Latvian language, many transcription specialists will not be familiar with transcription tools that include ASR functionality, because ASR for Latvian language is not available. However, they are familiar with Microsoft Word, so it is natural if the whole transcription process can be done in one tool.

As a part of domain adaptation for Saeima session transcription, some special modifications were implemented in the Speech-To-Text web service. These new features allow to produce ASR transcript as a special macro-enabled Word document with embedded audio that provides a convenient and familiar editing environment (Salimbajevs & Ikauniece, 2017).

The document is created from raw ASR transcript by an upgraded master server component of the Speech-To-Text web service. Also, a special version of web-service client web-page was implemented, it does not allow to view the transcript in browser, but it displays a link to the Word document that contains the transcription. The same link is also sent by e-mail.

Word document with transcript allows to play the embedded audio from any place in the document and highlights the words as they are being played (see **Figure 15**).

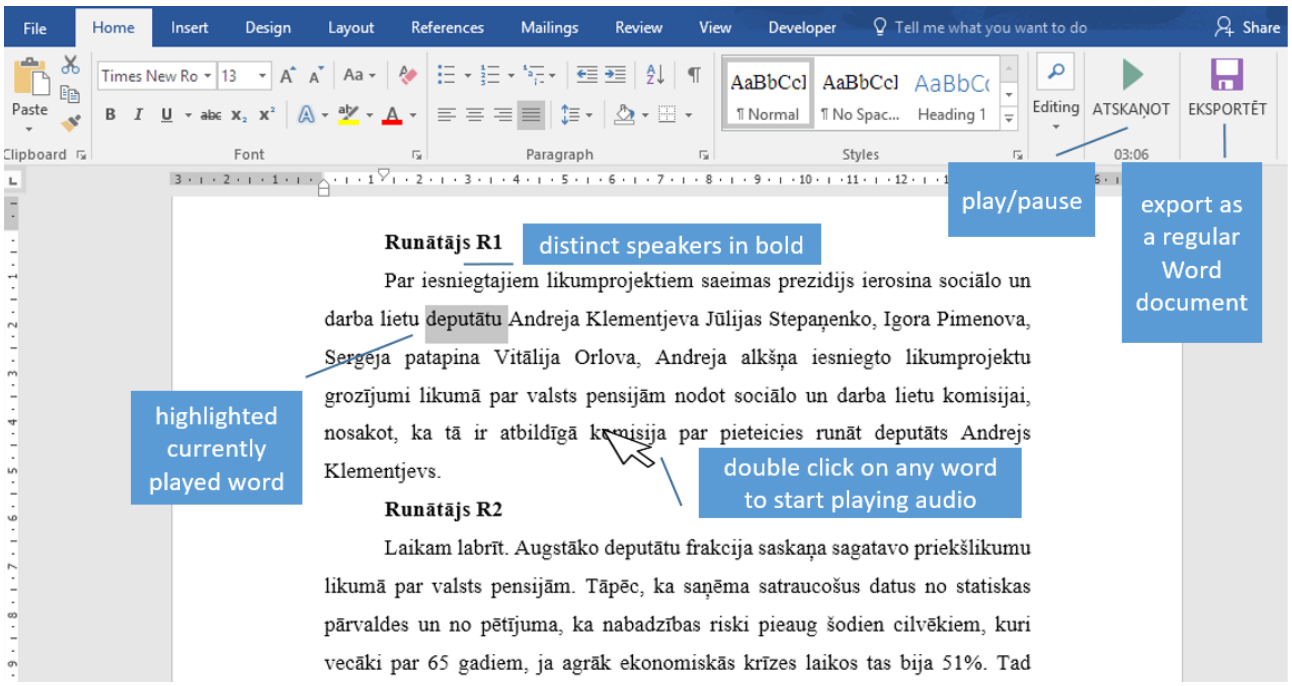

Figure 15. *Editing transcriptions in Microsoft Word.*

The document's text is divided into speakers (with speaker titles in bold) and speech transcripts (in formatted paragraphs with punctuation and formatting). Numbers are written in digit form, they are converted from words (as they appear in raw ASR transcript) using a finite state grammar.

Splitting into utterances and grouping by speakers is achieved by speaker diarization (performed by the LIUM SpkDiarization tool). Punctuation restoration is performed by the bidirectional LSTM neural network, which will be described in section 5.4. While both diarisation and punctuation restoration are not perfect and can produce a lot of errors, the readability of the document is greatly improved, allowing users to edit transcript more efficiently.

There are two custom buttons on the Word ribbon: the "play/pause" button plays and pauses the embedded audio, and the "export" button saves the document as a regular Word document without audio and macros. It is also possible to start playing the audio from any word by double clicking on it. There are also keyboard shortcuts for pausing and resuming the playback.

While the audio is being played, each current word is highlighted. If the user sees or hears a mistake, he can pause the playback by using the "pause" button or by clicking on the first incorrect word. He can then edit the erroneous segment as he would do in a regular Word document, and it will not break the audio and word alignment. The playback then can be resumed.

5.4. Punctuation Restoration

In some specific use cases, not only acoustic and language models should be adapted to the domain, but also the specific automatic post-processing of the transcripts should be implemented. For example, numbers and dates should rewritten with digits, URL and e-mail addresses should be

formatted. Also, some text level formatting might be necessary.

Some of these post-processing procedures can be fairly general and useful in many domains, but some can be very specific for a domain. They can be implemented in many different ways: most of the number conversion can be implemented as regular expressions, while others may require writing specific parsers.

This subsection will focus on punctuation restoration problem that is common for many domains and requires much more complex implementation than writing few regular expressions.

Since people usually do not pronounce punctuation when speaking, the output of generic automatic speech recognition system is a raw word sequence without any punctuation. This is sufficient in many cases, such as search queries, voice commands, or simple short informal messages, where word sequences are typically short. However, when audio recordings are longer, the transcript is difficult for humans to read and understand. Also, many of the natural language processing and understanding (NLU) tools are typically incompatible or perform badly with such raw input.

One or both of these issues might be important in some specific application scenarios, in which the adapted system should not only recognize spoken text, but also automatically recover punctuation (at least partially). One example of such domain is Latvian Saeima session transcripts. Adaptation of the language model significantly improved speech recognition quality, however even if session audio recordings are cut into 5 minute fragments, the raw ASR transcript is still hard to read and edit for humans.

There have been many previous studies on automatic punctuation restoration in speech transcripts. One of the most frequently proposed approaches for punctuation restoration is based on the so-called hidden event language model (LM), which uses a traditional N-gram statistical model trained on texts that include punctuation tokens (Stolcke et al., 1998). During decoding, the hidden event LM is used to predict where punctuation symbols should be inserted, based on n-gram probability of such sequences of words and punctuation.

Punctuation restoration can also be solved using conditional random fields (CRFs) (Lu & Ng, 2010; Lafferty et al., 2001) and recurrent neural networks (Tilk & Alumae, 2015).

Many approaches combine textual information with acoustic/prosodic features (Tilk & Alumae, 2015; Kolar et al., 2004; Huang & Zweig, 2002). However, this requires a corpus that simultaneously contains both acoustic/prosodic and punctuation annotations, which are rare.

This subsection will describe a punctuation restoration model for a Latvian dictation system. This subsection is based on publication by Salimbajevs (2016a). The focus will be on restoration of two of the most frequent and probably most important punctuation types – commas and periods. Other

punctuation marks will be mapped to these two and won't be recovered directly. Exclamation and question mark will be treated as period. Colon and semicolon will be treated as commas. All other punctuation will be filtered out and ignored.

The model used is a bidirectional long short-term memory model (BLSTM). BLSTM is a bidirectional recurrent neural network (BRNN) (Schuster & Paliwal, 1997) that contains long short-term memory cells (LSTM) (Hochreiter & Schmidhuber, 1997) in its hidden layers. Recurrent networks and LSTM recurrent networks have been used in many sequence labeling tasks in speech recognition and spoken language understanding (Yao et al., 2013; Yao et al., 2014). For example, forward LSTM and bidirectional LSTM can be used in acoustic modeling for phoneme classification (Sak, 2014; Senior et al., 2015) or in language modeling (Mikolov et al., 2010).

LSTM networks have already been used for punctuation recovery in punctuation-free tasks (Tilk & Alumae, 2015). However, there is a very little work on using bidirectional LSTM, and, there are no research on punctuation restoration in ASR transcripts for Latvian.

5.4.1. Data

Unfortunately, there is no Latvian corpus that contains both - punctuation and acoustic (e.g., pauses) annotations. Therefore, no acoustic features were used in this work, and the model is trained on purely textual information.

For this a 46 million sentence text corpus is used, which was collected by crawling Latvian web news portals (the same WebNews version 2016). The corpus contains about 905 million tokens, i.e., words and punctuation symbols. Because text was automatically collected, it may contain a noise, garbage, and spelling errors. Also, we need to replace punctuation and other special symbols with their respective full words (i.e., "comma", "period", "at sign", etc.), convert numbers from digits into written form, expand abbreviations (e.g., "km"), etc. To deal with these issues, the corpus is pre-processed as follows:

- The raw text is processed with natural language processing tools, which perform tokenization, garbage filtering (for example, mixed case tokens, non-alphanumeric tokens), number conversion (from digits to words and tries to find correct inflection), some abbreviation expansion, and true-casing.
- Optionally, the corpus can be stemmed to minimize vocabulary and model size.
- If the corpus is stemmed, a vocabulary is created from 100,000 of the most frequent stems. Otherwise, the vocabulary is created from 800,000 of the most frequent words, which are checked by a spell checker. Out-of-vocabulary (OOV) tokens are replaced with the "<UNK>" token.

- Next, punctuation symbols are replaced by the respective word forms and some more garbage processing is performed (incorrect punctuation sequences are filtered).
- Also, because the web news corpus is already segmented into sentences (this segmentation is not perfect, of course) during text collection, we randomly glue some sentences together to obtain utterances that contain 2 or more sentences. This is done in order to have training samples for cases where the model is expected to split an utterance into several sentences.

After processing, the text corpus contains 21 million utterances, 589 million words, 41 million punctuation commands, and 40 million punctuation symbols. From this processed corpus, two held-out sets are selected: 2000 utterances for validation during training and 3000 utterances for evaluation. All other data is used for model training.

During the development of the punctuation model, a small 1-hour long speech corpus of the debates in the Parliament of Latvia was collected (called Saeima-test, see Section 3.2). The corpus contains 439 segments, which were recorded by about 300 different speakers (estimated). It was collected for evaluation of ASR adaptation to Saeima domain(see previous section), but the interesting feature of this corpus is that its annotation contains punctuation. This makes it possible to perform punctuation restoration on the ASR output and to compare it with the reference annotation.

5.4.2. Hidden-event Language Model

The given task can be solved using the hidden-event LM approach (Stolcke et al., 1998), which uses a traditional N-gram language model trained on a corpus that contains regular text and so-called hidden-events. Applying this model to a raw sequence of words produces a sequence of the most likely words and hidden-events between them. In punctuation restoration case, hidden-events are punctuation marks.

The model is trained as follows:

- Punctuation marks in 21M utterance training corpus are annotated as hidden-events.
- A traditional 4-gram language model is trained on annotated corpus.

During decoding, the trained “hidden-event” LM will be used to produce a sequence of words and “punctuation-events” that can be unambiguously converted into a text with punctuation.

5.4.3. Bidirectional LSTM

The proposed solution is to use a bidirectional recurrent neural network with long short-term memory (BLSTM). The model that is needed for given task can be described as a classifier that will predict whether the current word in a word sequence corresponds to one of these 3 classes:

- A period must be inserted after this word.
- A comma must be inserted after this word.
- This word is a regular word and should be left as is.

Classes for other punctuation types can be added in a similar way if needed.

The BLSTM approach has several advantages over the classic hidden event N-gram LM, the LSTM recurrent model, and traditional feed-forward neural networks.

Probably, the most important problem of the N-gram model is weak generalization ability due to the data sparsity. For example, for a vocabulary of size 100,000 words, the number of all possible 3-grams is 10^{15}, but our training corpus has only about 50 million 3-grams. Many different smoothing methods were developed to deal with this issue. However, as it is known from language modelling, recurrent neural networks are much better at generalizing to unseen sequences (Mikolov et al., 2011a; Mikolov et al., 2011b).

The generalization problem can also be solved by using simple feed-forward neural networks. However, they share a common weakness with the N-gram models, as their context size is limited to a fixed number of tokens.

On the other hand, RNNs can theoretically use an unlimited previous context for their decisions. In practice, however, the ability of pure RNNs to remember long-term dependencies is limited, so RNNs are combined with LSTM units to make remembering long contexts possible.

Bidirectional LSTM RNN further extends RNN LSTM models by allowing to use not only the previous context of the word but also the next context, which can be important for the correct punctuation.

5.4.4. Model Architecture

The BLSTM neural network for punctuation restoration consists of two LSTM layers, an input layer, and an output layer. The architecture of described BLSTM model is shown in **Figure 16**.

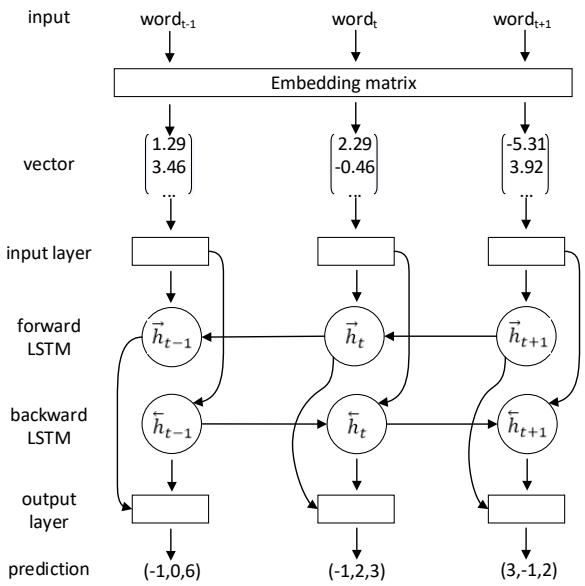

Figure 16. *BLSTM network for punctuation restoration*

Since the input consists of words, a special embedding matrix is used to translate each word to a 128-dimensional vector. This matrix consists of 100,000 (in case of stems) or 800,000 (when using full word forms) rows and 128 columns; each row vector represents a word from the vocabulary. The matrix is initialized with random values from uniform distribution in the range of (-1, 1) and then is trained together with the whole network.

Word vectors are then fed into the input layer of BLSTM that is common for both forward and backward 128-dimensional hidden LSTM layers. The backward LSTM layer $\overleftarrow{h}_t$ has a recurrent connection with the hidden layer $\overleftarrow{h}_{t-1}$ from the previous word, and the forward LSTM layer $\vec{h}_t$ is connected accordingly to the hidden layer activations $\vec{h}_{t+1}$ from the next word. Baseline LSTM units with forget gates and without peephole connections are used in this work. Hidden layer activations $\vec{h}_t$ and $\overleftarrow{h}_t$ are then passed to the softmax output layer, which calculates the probabilities of each class. To sum up, the forward pass of punctuation BLSTM can be described with the following equations (1-6):

$$x_t = embedding(word) \quad (1)$$

$$inp_t = (W_I x_t + b_I) \quad (2)$$

$$\overrightarrow{h_t} = LSTM(inp_t, \overrightarrow{h_{t+1}}) \quad (3)$$

$$\overleftarrow{h_t} = LSTM(inp_t, \overleftarrow{h_{t-1}}) \quad (4)$$

$$\overleftrightarrow{h_t} = concat(\overrightarrow{h_t}, \overleftarrow{h_t}) \quad (5)$$

$$pred(t) = softmax\left(W_o \overleftrightarrow{h_t} + b_O\right) \quad (6)$$

where W_O and W_I are weight matrices, b_O and b_I are bias vectors for output and input layers respectively, inp_t is input layer activations, *embedding* is a lookup function in an embedding matrix,

x_t is a word embedding vector, and *word* is the word we are trying to classify.

The model is implemented using TensorFlow (Abadi et al., 2015) framework and trained by optimizing cross-entropy using the Adam method (Kingma & Ba, 2015). All weights are initialized with random values from normal distribution with mean 0 and standard deviation 1. During training, weights are updated in "mini-batches" of 128 sequences. The learning rate at the start of the training is 0.0005 and is halved each time when there is no improvement on the validation set (adopted from (Tilk & Alumae, 2015) and Mikolov et al., 2011c). Training is stopped when the learning rate has been decreased 3 consecutive times and no improvement has been observed, or it is stopped when the maximum number of 20 epochs has been reached.

5.4.5. Evaluation

Both hidden-event and BLSTM models were evaluated by three different metrics:

- Precision for each class and overall precision.
- Recall for each class and overall recall.
- F1 for each class and overall F1.

First, the evaluation is performed on a held-out dataset of 3000 utterances; the results are presented in **Table 33**. The baseline hidden-event LM is outperformed by the proposed BLSTM model in all metrics by a large margin.

Table 33. *Results of the evaluation on a held-out test data.*

Metric	**Hidden-event**	**BLSTM**
Precision		
comma	0.726	**0.826**
period	0.600	**0.798**
Recall		
comma	0.444	**0.655**
period	0.155	**0.713**
F1		
comma	0.551	**0.731**
period	0.246	**0.753**

The biggest improvement is gained in recall; the difference is especially large (0.713 compared to 0.155) for periods. Noticeable improvement is also seen in precision, from 0.726 to 0.826 (14% relative) for commas and from 0.600 to 0.798 (33% relative) for periods.

Next evaluation was performed on a Saeima-test speech corpus. This time, raw ASR transcripts were used as input for both models. The acoustic model used was LSRC-nnet2-smbr and language model was WebNews-LM-3.

First, word error rate (WER) is measured, and commas and periods are calculated as word

tokens. Then, the alignment from the WER calculation is used to evaluate precision, recall, and F1 (results in **Table 34**).

Table 34. *Results of the evaluation on an ASR output.*

Metric	**Hidden-event**	**BLSTM**
Precision		
comma	0.709	**0.845**
Period	0.732	**0.789**
Recall		
comma	0.497	**0.646**
period	0.149	**0.642**
F1		
comma	0.585	**0.732**
period	0.247	**0.708**
WER	28.25%	**15.68%**

Again, when compared with the hidden-event model, improvement is gained in all metrics, and the biggest difference is seen in recall for periods (0.642 compared to 0.149). Overall, the proposed model achieved F1-scores of 0.732 for commas and 0.708 for periods, which is very close to the numbers obtained on a held-out set.

To conclude the model evaluation showed promising results. BLSTM outperformed the hidden-event LM and achieved F1-scores of 0.732 for commas and 0.708 for periods on raw ASR transcripts. Subjectively evaluation of processed transcripts also shows that although there are still many punctuation marks missing, the readability of the transcript is greatly improved, making working with it much easier.

At the time this was the state-of-the-art punctuation restoration model for Latvian. Recently, it was outperformed by using a self-attention neural network instead of BLSTM (Varavs & Salimbajevs; 2018).

5.5. Street Address Recognition

The aim of this domain adaptation experiment is to create ASR for recognition of Latvian street addresses and to assess the effect of adaptation. Since Latvian addresses have a specific format, a customized ASR is needed to recognize and process this format. Also, the vocabulary of general domain ASR system is not likely to contain all street names and addresses, so it is necessary to create new customized vocabulary that complements it with streets and addresses of Latvia.

Such ASR system would be useful for many specific cases, such as voice control in the different navigation applications for smartphones or submitting meter reading to an IVR system, where the address of client and meter reading should be named.

5.5.1. Data

For language model training a corpus of the Latvian addresses was collected. For this, a web-crawler was developed, which processed the Latvian address list on the website VisaLatvija.lv[6]. The corpus contains 873363 addresses. Each address is normalized, so that all addresses are in the same format:

```
<street name> <house number> <building>, [city|pagasts|district].
```

The street name must include the word “iela”, “gatve” etc. But the number must be written in words and not in digits, and could include the words “A”, “B” or “korpuss” (housing or bulding), when necessary.

A test set (“Address-test”) of 136 audio recordings (16 MB, 8 minutes 44 seconds, 506 words) was collected to evaluate the quality of address recognition. The audio data were recorded by 35 different speakers (12 women and 23 men). 81 recordings were done in office rooms, and 55 entries in cars (with and without background music).

Recording was performed using different smartphones and have different original formats. Subsequently, all test set entries were converted to WAV format with 16 kHz sampling rate. An 8 kHz version was also prepared.

5.5.2. Adaptation Method and Post-processing

Speech recognition was adapted in two ways:

- LSRC-nnet2-smbr model was trained on artificial narrowband data by downsampling the original LSRC.
- 4-gram language model was built from address corpus.

This experiment uses a narrowband acoustic model because:

- The intended usage domain of this ASR is IVR and mobile phones.
- Telephone networks typically use 8 kHz sampling rate when transmitting audio.
- 8 kHz is the default sampling rate on many mobile phones and supported by even very old models.

Once the ASR has the recognized the address, it must be converted to the normalized output form. This is necessary, because the output of the speech recognizer is a raw word sequence (e.g. “Vienības gatve seventy-five A” instead of “Vienības gatve 75A”).

[6] Travel portal VisaPasaule.lv. http://visapasaule.lv/adreses/riga

Therefore, prior to displaying or passing the output to the other programs (e.g. Waze), it is necessary to perform post-processing of the ASR output - translate the number words into digits and normalize address components (for example, convert “korpuss” to “k-”, “75 A” to “75A” etc).

For post-processing, a formal grammar was developed which describes how to convert ASR output to address (fragment of this grammar is shown in **Figure 17**). From this grammar, a finite state transducer was built and incorporated into the ASR as post-processing step.

```
<address> = <street> ":<space>" <number> (<mod>) (":<space>" <area>) |
":<space>" <city> | ... ;
<street> = <word> ":space" ("iela" | "gatve" | ...);
<number> = <number1> | <number2> | <number3> ;
<mod> = <letter> | "korpuss:k-" <number> ;
...
// divciparu vesels skaitlis
<number2> = :1 <teens> | <tens> (:0 | <digit>);
// vārdu saraksts
<digit> = viens:1 | divi:2 | trīs:3 | četri:4 | pieci:5 | seši:6 |
septiņi:7 | astoņi:8 | deviņi:9 ;
<teens> = desmit:0 | vienpadsmit:1 | divpadsmit:2 | trīspadsmit:3 |
četrpadsmit:4 | piecpadsmit:5 | sešpadsmit:6 | septiņpadsmit:7 |
astoņpadsmit:8 | deviņpadsmit:9 ;
<word> = "Abavas:Abavas" | "Brīvības:Brīvības" | ...
```

Figure 17. *Fragment of address post-processing grammar.*

5.5.3. Evaluation

The results of the evaluation are summarised in **Table 35**. As a baseline, general domain ASR is used with narrowband LSRC-nnet2-smbr acoustic and WebNews-LM-3 language models. First, the test set downsampled to 8 kHz and evaluation of non-adapted narrowband ASR was performed. The obtained WER is 25.95%.

Next, the test set was decoded with the customized ASR that used a 4-gram language model built from addresses. The table shows that the WER of the adapted 8kHz ASR (12.38%) is a little better than WER of 16 kHz ASR (13.37%). Both are significantly better than non-adapted ASR.

Table 35. *Results of WER evaluation for address recognition task.*

ASR	**Test set**	**WER, %**
General, 8 kHz	Office rooms and cars	25.95
Adapted, 8 kHz	Office rooms only	8.65
	Cars only	17.02
	Office rooms and cars	12.38
Adapted, 16 kHz	Office rooms only	8.65
	Cars only	19.15
	Office rooms and cars	13.37

The table also shows that the subset from entries made in the office rooms is recognized more

precisely than the subset from the recordings in the car.

When using ASR with the adapted language model, a significant improvement in WER is obtained. There are no significant differences between 8 kHz or 16 kHz acoustic models on the test set used in the experiment, but results are slightly better for 8 kHz acoustic model. It can be seen that for audio recordings in the car, the 8 kHz model provide a lower WER, which might be explained that the narrowband acoustic model is more robust, as it's trained on lower quality data.

The achieved results allow to conclude, that after some small improvements, the address recognition system developed during this research can be used in real-life applications. After switching to AUG2-TDNN acoustic models WER was reduced from 12.38% to 7.94% and the system was deployed to Latvian Speech-To-Text webservice, where it can be accessed from mobile application Tildes Balss.

6. RESULTS

This section summarizes final evaluation results of models and systems that were developed during author's research and described in detail in previous sections.

6.1. Speech Recognition Evaluation

The speech recognition systems are evaluated by decoding some evaluation recordings and then comparing the recognized text with reference transcription and calculating WER. Evaluation data should be accurately annotated and represent the intended usage scenario (e.g., for evaluation of telephone speech recognition the evaluation should be performed on real telephone speech data).

The results of the evaluation of the general domain ASR system for Latvian can be found in **Table 36**.

Table 36. *Evaluation of general domain ASR*

Evaluation data	WER, %
EvalWebNews	10.3
EvalGeneral	10.4
Saeima-test	8.3

In the scope of the thesis a number of domain adapted systems were developed and evaluated on corresponding domain test sets. The results of such evaluations are presented in **Table 37**.

Table 37. *Evaluation of domain adapted ASR systems*

Domain	Evaluation data	WER, %
Dictation	LDSC-test	12.6
Saeima	Saeima-test	5.9
Street addresses	Address-test	7.9

To deal with infective and morphologically rich nature of Latvian language, experiments with sub-word recognition were performed.

The results in **Table 38** show that sub-word ASR performs better than baseline full word ASR. The evaluation was performed for non-adapted general domain ASR systems. Also, it can be seen that BPE sub-word approach outperforms the Morfessor based approach.

A small OOV analysis was performed on EvalWebNews. This test set contains 36 words that are not in the WebNews (version 2014) corpus, so they can not be recognized by baseline ASR. Sub-word ASR correctly recognizes 14 of 36 (38.9%), therefore it can be concluded that sub-word approach is able to partially overcome sparsity created by inflected forms and improve the recognition quality.

Table 38. *Evaluation of sub-word ASR*

Evaluation data	WER, %		
	Baseline	Morfessor sub-words	BPE sub-words
EvalWebNews	10.3	10.2	**10.1**
EvalGeneral	10.4	9.8	**9.6**
Saeima	8.3	7.9	**7.8**

In August 2017 speech recognition for Latvian passed an important milestone, as Google added Latvian speech recognition to their services. In September 2017 and again in May 2018, author performed an evaluation of this system on the EvalWebNews corpus and got a word error rate of 50.6%. Quick analysis showed that much of the errors are deletions, which happen because Google ASR skips words if its confidence is too low. If we ignore the deletions the WER can be calculated as 36.2%.

Around that time UK company Speechmatics also started to provide ASR service for Latvian. The evaluation of this system on EvalWebNews resulted in WER of 25.2%. Word deletion problem was not found in this case.

The best ASR system developed in this work significantly outperforms Google and Speechmatics ASR and achieves WER of 10.1% on the same dataset.

6.2. Punctuation Restoration Evaluation

First, the evaluation of BLSTM punctuation restoration model was performed on a held-out dataset of 3000 utterances from WebNews; the results are presented in **Table 39Table 33**.

Table 39. *Results of the evaluation on a held-out test data.*

Metric	Comma	Period
Precision	0.826	0.798
Recall	0.655	0.713
F1	0.731	0.753

Next evaluation was performed on a Saeima-test speech corpus using raw ASR transcripts (results in **Table 40**). The acoustic model used was LSRC-nnet2-smbr (section 3.4 in full thesis) and language model was WebNews-LM-3 (section 4.4 in full thesis).

Table 40. *Results of the evaluation on an ASR output.*

Metric	Comma	Period
Precision	0.845	0.789
Recall	0.646	0.642
F1	0.732	0.708

To conclude the model evaluation showed promising results. BLSTM model achieved F1-scores of 0.732 for commas and 0.708 for periods on raw ASR transcripts. Subjectively evaluation of processed transcripts also shows that although there are still many punctuation marks missing, the readability of the transcript is greatly improved, making working with it much easier.

At the time this was the state-of-the-art punctuation restoration model for Latvian. Recently, it was outperformed by using a self-attention neural network instead of BLSTM (Varavs & Salimbajevs; 2018).

CONCLUSIONS

This research is the first large-scale work dedicated to the modelling of Latvian language for automatic speech recognition. It covers both theoretical and practical aspects of building speech recognition system for a small, morphologically rich and under-researched language, such as Latvian. Although the research in this work is focused on issues related to speech recognition for Latvian, the same methods and guidelines can be applied to other languages. For example, following the same guidelines author successfully developed large vocabulary automatic speech recognition system for Lithuanian (Salimbajevs & Kapočiūtė-Dzikienė, 2018).

The principal difference between the work in this dissertation and other work on speech recognition has been the investigation of almost all aspects of ASR for Latvian language, including research on acoustic and language models, pronunciation modelling, sub-word speech recognition, data collection and augmentation, inverse text normalization (punctuation restoration) and practical system development.

The work began with an initial HMM-GMM acoustic models (which are now considered almost outdated), stub language model and examination of different grapheme-to-phoneme models. The phoneme set and grapheme-to-phoneme model were selected based on speech recognition experiment results. The best result was achieved using graphemes as phonemes, 4 diphthongs and simple mapping algorithm.

Then, models, training and evaluation data were gradually refined. State-of-the-art HMM-DNN acoustic models for Latvian were trained using 100h Latvian Speech Recognition Corpus and allowed to greatly improve the recognition quality over initial HMM-GMM models.

Several evaluation sets were created: 0.38 hr and 2.5 hr general domain evaluation sets (EvalWebNews and EvalGeneral), 1 hr dictated evaluation set (LDSC-test) and 1 hr evaluation set from Saeima session recordings (Saeima-test).

Introduction of 2-pass decoding allowed to use unpruned language models with reasonable RAM usage. A very large vocabulary (900K units) language model was trained using text corpus collected on the Web.

A special text corpus processing and filtering method together with special treatment of acronyms allowed to greatly improve the quality of language model. Combining trained HMM-DNN acoustic and improved language models allowed to create a baseline general domain large vocabulary speech recognition for Latvian, which achieved WER of 19.4%.

Several domain adaptation experiments were performed in order to understand how to design speech recognition systems and adapt them to specific applications like dictation, Saeima session

transcription and street address.

A practical application of the research resulted in the public online Speech-To-Text service for Latvian and integration of audio transcription and dictation functionality into a Tildes Birojs software package. The ASR web-service also provides a speech recognition for a mobile app Reizrēķins (available in Google Play store), which teaches kids multiplication. In 2015 these were the first services of this kind for Latvian language. The successful practical implementation proves the research hypothesis 3.

In the scope of development of practical application experiments were performed on processing the raw output of speech recognition systems. A bidirectional LSTM neural network was trained for punctuation restoration in ASR transcripts. At the time this was the state-of-the-art punctuation restoration model for Latvian. This model served as a baseline in (Varavs & Salimbajevs; 2018), in which a self-attention neural network (also known as Transformer model) was applied to punctuation task and achieved new state-of-the-art accuracy. Therefore, the research hypothesis 4 has been proven.

In parallel with practical ASR development, a research on improving general recognition quality continued. The proposed automatic data collection method allowed to obtain additional 186 hours of speech using partially correctly annotated data found on the Web. Using this data in training allowed to achieve significant improvements in WER, from 19.4% on EvalWebNews to 16.9% and from 8.6% to 7.2% on Saeima session transcription task (using Saeima-LM).

New data allowed to train more advanced neural network models, which can model long range temporal context. Time-delay neural networks (TDNN) are used in this thesis. These models achieve almost state-of-the-art recognition accuracy (in experiments for other languages) but require less data and are much faster to train and tune, which make them very practical in cases with limited data and hardware resources like this work.

As initial decisions on grapheme-to-phoneme modelling were made using stub LM and most simple HMM-GMM acoustic models, it was decided to repeat some of the G2P experiments on LSRC-tri3b model and check if previous decisions are still the most optimal. It was found out that pure grapheme-based method, that maps letters and phonemes one-to-one, works the best.

This data acquisition method can be combined with data augmentation, which increases size of training data multiple times by adding distorted copies of original utterances. In this work the total size of training data is increased 6 times (from 294 hours to 1764 hours) by performing speed perturbation, lowpass filtering and reverberation. Data augmentation helped to improve overall robustness of the Latvian ASR and also reduce WER on EvalGeneral from 10.5% to 10.1%.

To conclude, both data augmentation and automatic data collection from the Web turned out to

be very useful methods for improving the acoustic models for Latvian speech recognition models. Both methods allowed to significantly improve the word error rate on different testing corpora. And finally, there is still a room for future improvement as there exist many other sources of inaccurately annotated data on the Web and other databases of noises and room impulse responses. Adopting those may reduce WER even further.

Improved acoustic model was integrated into general domain and domain adapted ASR systems, that were deployed on a Latvian Speech-To-Text webservice. The new acoustic model allowed to improve word error rate of all systems: (1) from 16.9% to 10.3% on EvalWebNews, (2) from 7.2% to 5.9% on Saeima session transcription task Saeima-test (using Saeima-LM), 3) from 23.9% to 12.6% on dictated speech from LDSC-test (using WebNewsDict-LM) and 4) from 12.4 to 7.9% on street address recognition set Address-test (using special street address LM).

General domain, dictation and street address recognition systems can be accessed by using mobile app Tildes Balss, which is first mobile app that enable Latvian language speakers to speak to their phone in their language, dictate text message and voice-enter street addresses.

To deal with infective and morphologically rich nature of Latvian language, experiments with sub-word recognition were performed. The idea is to divide word surface forms into shorter units and perform recognition on sub-word level. The number of such units is much smaller, and they are much more frequent. The best results were achieved using byte-pair encoding (BPE) inspired segmentation method. A small but consistent improvement over baseline word-level ASR was observed: from 10.3% to 10.1% on general domain EvalWebNews corpus, from 10.4% to 9.8% on EvalGeneral and from 8.3% to 7.8% on debates domain Saeima-test corpus (results for general non-adapted LM).

In August 2017 speech recognition for Latvian passed an important milestone, as Google added Latvian speech recognition to their services. In September 2017 and then again May 2018 author performed an evaluation on the EvalWebNews corpus and got a word error rate of 50.6%. Quick analysis showed that much of the errors are deletions, which happen because Google ASR skips words if its confidence is too low. If we ignore the deletions the WER can be calculated as 36.2%. Author also performed evaluation of Speechmatics ASR for Latvian, which also appeared around 2017. Evaluation was performed in September 2018 and resulted in WER of 25.2% on EvalWebNews. Word deletion problem was not found for Speechmatics system. For comparison, the best ASR system developed in this work significantly outperforms both Google and Speechmatics ASR and achieves WER of 10.1% on the same dataset.

The positive evaluation results on different test corpora and comparison with Google's and Speechmatics systems allow to conclude that the research hypotheses 1 and 2 has been successfully proven.

We can conclude that results of the thesis work prove that effective models for Latvian language have been developed, so the aim of the research is achieved. The thesis also provides guidelines and answers for many research questions that can arise when developing speech recognition for previously unresearched languages. The experimental evaluation results and research approbation shows that all research hypotheses have been successfully proven. The goal of the thesis has been reached and all objectives have been completed.

However, in answering many of the questions that were posed, this thesis has itself prompted many more. These will hopefully provide the stimulus for further research in this area.

For example, neural network language models recently have been constantly showing improvements for many languages. Therefore, there is a strong motivation in adopting this methodology for Latvian ASR. However, a large vocabulary of infected words is a difficult challenge, as it results in huge word embedding and soft-max layers, that makes convergence of the models difficult and also requires computational resources for training and inference.

This problem can be possibly solved by using sub-word units, which is another perspective direction for future work. However, sub-word decomposition makes text sequences many times longer and requires an advanced model that can handle such long-distance contexts. Experiments performed in this work showed that sub-word ASR can outperform word-level ASR if n-gram models are used. While there is a performance penalty, the recognition speed is adequate and practical on modern hardware.

Another hot-topic is end-to-end speech recognition and character-based models. Considering that Latvian has highly phonemic orthography and rich morphology, these methods can be potentially a very effective and elegant solution. The analysis of recent research suggests that end-to-end approach greatly simplifies the training of speech recognition models and have a great potential for infective languages as it allows for lexicon-free systems (solves large vocabulary problem). However, such methods require much larger amount of training data (the method proposed in this work can help to acquire more data) and currently are unable to achieve state-of-the-art results of classic systems if used without classic language model (which neutralizes the lexicon-free advantage).

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. In *None* (p. 19). Retrieved from http://download.tensorflow.org/paper/whitepaper2015.pdf

Alumae, T. (2014). Full-duplex Speech-to-text System for Estonian. In *Human Language Technologies-The Baltic Perspective: Proceedings of the Sixth International Conference Baltic HLT 2014* (Vol. 268, p. 3).

Anastasakos, T., McDonough, J., & Makhoul, J. (1997). Speaker adaptive training: a maximum likelihood approach to speaker normalization. *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, *2*, 0–3. http://doi.org/10.1109/ICASSP.1997.596119

Anguera, X., Luque, J., & Gracia, C. (2014). Audio-to-text alignment for speech recognition with very limited resources. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH'2014)* (pp. 1405–1409).

Atal, B. S. (1976). Automatic Recognition of Speakers from Their Voices. *Proceedings of the IEEE*, *64*(4), 460–475. http://doi.org/10.1109/PROC.1976.10155

Auziņa, I., Pinnis, M., & Darģis, R. (2014). Comparison of rule-based and statistical methods for grapheme to phoneme modelling. In *Human Language Technologies--The Baltic Perspective-Proceedings of the Sixth International Conference Baltic HLT 2014*.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. (2016). End-to-End Attention-based Large Vocabulary Speech Recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (pp. 4945–4949). http://doi.org/10.1007/s13398-014-0173-7.2

Bahl, L., Brown, P., de Souza, P., & Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on* (Vol. 11, pp. 49–52). http://doi.org/doi: 10.1109/icassp.1986.1169179

Bahl, L. R., de Soutza, P. V., Gopalakrishnan, P. S., Nahamoo, D., & Picheny, M. A. (1991). Context dependent modeling of phones in continuous speech using decision trees. In *HLT '91 Workshop on Speech and Natural Language,*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*(Feb), 1137–1155.

Brown, P. F., De Souza, P. V., Mercer, R. L., Della Pietra, V. J., & Lai, J. C. (1992). Class-Based n-gram Models of Natural Language. *Computational Linquistics*, *18*(1950), 467–479.

Butzberger, J., Murveit, H., Shriberg, E., & Price, P. (1992). Modeling Spontaneous Speech Effects in Large Vocabulary Speech Recognition Applications. In *In Speech and Natural Language Workshop*. Morgan Kaufmann.

Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 2016–May, pp. 4960–4964). http://doi.org/10.1109/ICASSP.2016.7472621

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv Preprint arXiv:1406.1078*.

Chorowski, J., & Jaitly, N. (2016). Towards better decoding and language model integration in sequence to sequence models. *arXiv Preprint arXiv:1612.02695*.

Clark, A. (2007). Learning deterministic context free grammars: The Omphalos competition. *Machine Learning*, *66*(1), 93–110. http://doi.org/10.1007/s10994-006-9592-9

Clark, A., & Lappin, S. (2010). Unsupervised Learning and Grammar Induction. In *The Handbook of Computational Linguistics and Natural Language Processing* (pp. 197–220). http://doi.org/10.1002/9781444324044.ch8

Creutz, M., & Lagus, K. (2005). *Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. Publications in Computer and Information Science, Report A81, Helsinki University of Technology*.

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, *20*(1), 30–42. http://doi.org/10.1109/TASL.2011.2134090

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, *20*(1), 30–42. http://doi.org/10.1109/TASL.2011.2134090

Darģis, R. (2014). Fonētiskās vārdnīcas un valodas modeļa izstrāde latviešu valodas runas apstrādei (ENG: he development of phonetic dictionary and language model for latvian speech processing).

Darģis, R., & Znotiņš, A. (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).

Davis, K. H., Biddulph, R., & Balashek, S. (1952). Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, *24*(6), 637–642.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(4), 788–798. http://doi.org/10.1109/TASL.2010.2064307

Deng, L., Li, J., Huang, J. T., Yao, K., Yu, D., Seide, F., … Acero, A. (2013). Recent advances in deep learning for speech research at Microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 8604–8608). http://doi.org/10.1109/ICASSP.2013.6639345

Ferguson, J. D. (1980). Hidden Markov analysis: an introduction. *Hidden Markov Models for Speech*, *1*, 8–15.

Fook, C. Y., Muthusamy, H., Chee, L. S., Yaacob, S. Bin, & Adom, A. H. Bin. (2013). Comparison of speech parameterization techniques for the classification of speech disfluencies. *Turkish Journal of Electrical Engineering & Computer Sciences*, *21*(Sup. 1), 1983–1994.

Fosler-Lussier, E., & Morgan, N. (1999). Effects of speaking rate and word frequency on pronunciations in conversational speech. *Speech Communication*, *29*(2), 137–158. http://doi.org/10.1016/S0167-6393(99)00035-7

Gage P., A new algorithm for data compression, *The C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

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). Retrieved from http://hdl.handle.net/10062/2521

Goldwater, S., Jurafsky, D., & Manning, C. D. (2010). Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, *52*(3), 181–200. http://doi.org/10.1016/j.specom.2009.10.001

Goryainova, M., Grouin, C., Rosset, S., & Vasilescu, I. (2014). Morpho-Syntactic Study of Errors from Speech Recognition System. In N. C. (Conference Chair), K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, … S. Piperidis (Eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA).

Graves, A. (2008). *Supervised Sequence Labelling with Recurrent Neural Networks*. Universitat Munchen.

Graves, A., Fernandez, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *Proceedings of the 23rd International Conference on Machine Learning*, 369–376. http://doi.org/10.1145/1143844.1143891

Haeb-Umbach, R., & Ney, H. (1992). Linear discriminant analysis for improved large vocabulary continuous speech recognition. *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, *1*, 13–16. http://doi.org/10.1109/ICASSP.1992.225984

Hazen, T. J. (2006). Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Proceedings of the 9th International Conference on Spoken Language Processing (INTERSPEECH'2006)* (Vol. 2006, pp. 1606–1609).

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, *87*(4), 1738–1752. http://doi.org/10.1121/1.399423

Hermansky, H., Ellis, D. P. W., & Sharma, S. (2000). Tandem connectionist feature extraction for conventional HMM systems. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 3, pp. 1635–1638). http://doi.org/10.1109/ICASSP.2000.862024

Hochreiter, S., & Schmidhuber, J. J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1–32. http://doi.org/10.1162/neco.1997.9.8.1735

Huang, J., & Zweig, G. (2002). Maximum entropy model for punctuation annotation from speech. *Interspeech*, 917–920. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.2152&rep=rep1&type=pdf

Huang, X., Acero, A., & Hon, H.-W. (2001). *Spoken language processing: a guide to theory, algorithm, and system development*. *Prentice Hall*. Retrieved from http://books.google.com/books?id=reZQAAAAMAAJ&pgis=1

Huang, X., Baker, J., & Reddy, R. (2014). A Historical Perspective of Speech Recognition. *Commun. ACM*, *57*(1), 94–103. http://doi.org/10.1145/2500887

Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M., & Rudnicky, A. I. (2006). Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings* (Vol. 1, p. I-185-I-188). http://doi.org/10.1109/ICASSP.2006.1659988

Jelinek, F., Bahl, L., & Mercer, R. (1975). Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, *21*(3), 250–256.

Jiampojamarn, S., Kondrak, G., & Sherif, T. (2007). Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. In *HLT-NAACL* (Vol. 7, pp. 372–379).

Juang, B. H., Chou, W., & Lee, C. H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, *5*(3), 257–265. http://doi.org/10.1109/89.568732

Juang, B.-H., & Rabiner, L. R. (2006). Automatic speech recognition – a brief history of the technology development. *Encyclopedia of Language and Linguistics*, 806–819. http://doi.org/10.1016/B0-08-044854-2/00906-8

Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, 1–13.

Kingsbury, B. (2009). Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (pp. 3761–3764). http://doi.org/10.1109/ICASSP.2009.4960445

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. http://doi.org/10.1109/ICASSP.1995.479394

Ko, T., Peddinti, V., Povey, D., & Khudanpur, S. (2015). Audio augmentation for speech recognition. In *INTERSPEECH* (pp. 3586–3589).

Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., & Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (pp. 5220–5224).

Kolář, J., Švec, J., & Psutka, J. (2004). Automatic punctuation annotation in Czech broadcast news speech. In *SPECOM´ 2004* (pp. 319–325). Retrieved from http://www.kky.zcu.cz/cs/publications/KolarJ_2004_Automaticpunctuation

Lafferty, J., McCallum, A., & Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, *8*(June), 282–289. http://doi.org/10.1038/nprot.2006.61

Lamel, L. (2012). Multilingual Speech Processing Activities in Quaero: Application to Multimedia Search in Unstructured Data. In *Baltic HLT* (pp. 1–8).

Lee, K. F., Hon, H. W., & Reddy, R. (1990). An Overview of the SPHINX Speech Recognition System. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *38*(1), 35–45. http://doi.org/10.1109/29.45616

Li, J., Yu, D., Huang, J. T., & Gong, Y. (2012). Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM. In *2012 IEEE Spoken Language Technology Workshop (SLT)* (pp. 131–136). http://doi.org/10.1109/SLT.2012.6424210

Lu, W., & Ng, H. T. (2010). Better Punctuation Prediction with Dynamic Conditional Random Fields. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, (October), 177–186. Retrieved from http://www.aclweb.org/anthology/D10-1018

Meignier, S., & Merlin, T. (2010). LIUM SpkDiarization: an open source toolkit for diarization. In *CMU SPUD Workshop* (Vol. 2010).

Mermelstein, P. (1976). Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, *116*, 374–388.

Miao, Y., Zhang, H., & Metze, F. (2014). Towards Speaker Adaptive Training of Deep Neural Network Acoustic Models. *Interspeech*, *XX*(September), 2189–2193. http://doi.org/10.1109/TASLP.2015.2457612

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent Neural Network based Language Model. *Interspeech*, (September), 1045–1048.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Černocký, J. (2011a). Empirical Evaluation and Combination of Advanced Language Modeling Techniques. *Interspeech*, (August), 605–608.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., & Khudanpur, S. (2011b). Extensions of recurrent neural network language model. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 5528–5531. http://doi.org/10.1109/ICASSP.2011.5947611

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., & Černocký, J. (2011c). RNNLM --- Recurrent Neural Network Language Modeling Toolkit. *Proceedings of ASRU 2011*, 1–4.

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

Moore, R. C., & Lewis, W. (2010). Intelligent selection of language model training data. In *ACL 2010 - 48th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference* (pp. 220–224). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84859981825&partnerID=tZOtx3y1

Nadas, A. (1983). A Decision-Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood. IEEE Trans. on Acoustics, Speech and Signal Processing, 1983, 4, pp. 814-817.

Ney, H., Essen, U., & Kneser, R. (1994). On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech & Language*, *8*(1), 1–38. http://doi.org/10.1006/csla.1994.1001

Novak, J. R., Minematsu, N., & Hirose, K. (2012). WFST-based Grapheme-to-Phoneme Conversion : Open Source Tools for Alignment , Model-Building and Decoding. *10th International Workshop on Finite State Methods and Natural Language Processing*, (1), 45–49.

Oparin, I., Lamel, L., & Gauvain, J.-L. (2013). Rapid development of a Latvian speech-to-text system. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 7309–7313). http://doi.org/10.1109/ICASSP.2013.6639082

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 2015–Augus, pp. 5206–5210). http://doi.org/10.1109/ICASSP.2015.7178964

Pappu, A., Misu, T., & Gupta, R. (2016). Investigating critical speech recognition errors in spoken short messages. In *Situated Dialog in Speech-Based Human-Computer Interaction* (pp. 71–82). Springer.

Peddinti, V., Manohar, V., Wang, Y., Povey, D., & Khudanpur, S. (2016). Far-Field ASR Without Parallel Data. In *INTERSPEECH* (pp. 1996–2000).

Pinnis, M., & Skadiņš, R. (2012). MT adaptation for under-resourced domains-what works and what not. In *Frontiers in Artificial Intelligence and Applications* (Vol. 247, pp. 176–184). http://doi.org/10.3233/978-1-61499-133-5-176

Pinnis, M., Auziņa, I., & Goba, K. (2014). Designing the Latvian Speech Recognition Corpus. In *Proceedings of the 9th edition of the Language Resources and Evaluation Conference (LREC'14)* (pp. 1547–1553).

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).

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., … Vesely, K. (2011). The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding* (pp. 1–4). http://doi.org/10.1017/CBO9781107415324.004

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., … Khudanpur, S. (2016). Purely sequence-trained neural networks for ASR based on lattice-free MMI. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (Vol. 8-12-NaN-2016, pp. 2751–2755). http://doi.org/10.21437/Interspeech.2016-595

Povey, D., & Woodland, P. C. (2002). Minimum phone error and I-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on* (Vol. 1, p. I--105).

Prahallad, K., & Black, A. W. (2011). Segmentation of monologues in audio books for building synthetic voices. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(5), 1444–1449.

Rehm, G., & Uszkoreit, H. (2012). META-NET White Paper Series: Europe's Languages in the digital age. Springer, Heidelberg etc.

Rouvier, M., Dupuy, G., Gay, P., Khoury, E., Merlin, T., & Meignier, S. (2013). An open-source state-of-the-art toolbox for broadcast news diarization. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (pp. 1477–1481).

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH)*, (September), 338–342.

Salimbajevs, A., & Pinnis, M. (2014). Towards Large Vocabulary Automatic Speech Recognition for Latvian. In Human Language Technologies – The Baltic Perspective (pp. 236–243). IOS Press.

Salimbajevs, A., & Strigins, J. (2015a). Latvian Speech-to-Text Transcription Service. In INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015 (pp. 723-725)

Salimbajevs, A., & Strigins, J. (2015b). Error Analysis and Improving Speech Recognition for Latvian Language. In Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria (pp. 563–569).

Salimbajevs, A., & Strigins, J. (2015c). Using sub-word n-gram models for dealing with OOV in large vocabulary speech recognition for Latvian. In B. Megyesi (Ed.), Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Institute of the Lithuanian Language, Vilnius, Lithuania (pp. 281–285). Linköping University Electronic Press / ACL.

Salimbajevs, A. (2016a). Bidirectional LSTM for Automatic Punctuation Restoration. In I. Skadina & R. Rozis (Eds.), Human Language Technologies - The Baltic Perspective - Proceedings of the Seventh International Conference Baltic HLT 2016, Riga, Latvia, October 6-7, 2016 (Vol. 289, pp. 59–65). IOS Press. http://doi.org/10.3233/978-1-61499-701-6-59

Salimbajevs, A. (2016b). Towards the First Dictation System for Latvian Language. In I. Skadina & R. Rozis (Eds.), Human Language Technologies - The Baltic Perspective - Proceedings of the

Seventh International Conference Baltic HLT 2016, Riga, Latvia, October 6-7, 2016 (Vol. 289, pp. 66–73). IOS Press. http://doi.org/10.3233/978-1-61499-701-6-66

Salimbajevs, A., & Ikauniece, I. (2017). System for speech transcription and post-editing in Microsoft Word. In INTERSPEECH 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017 (pp. 825–826).

Salimbajevs, A. (2018). Creating Lithuanian and Latvian Speech Corpora from Inaccurately Annotated Web Data. In LREC 2018, 11th International Conference on Language Resources and Evaluation, Miyazaki, Japan, May 7-12, 2018.

Salimbajevs, A., & Kapočiūtė-Dzikienė, J. (2018). General-purpose Lithuanian Automatic Speech Recognition System. In Human Language Technologies - The Baltic Perspective - Proceedings of the Seventh International Conference Baltic HLT 2018, Tartu, Estonia, September 27-29, 2018.

Saraçlar, M., Nock, H., & Khudanpur, S. (2000). Pronunciation modeling by sharing Gaussian densities across phonetic models. *Computer Speech & Language*, *14*(2), 137–160. http://doi.org/10.1006/csla.2000.0140

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. http://doi.org/10.1109/78.650093

Schwartz, R., Nguyen, L., Kubala, F., Chou, G., Zavaliagkos, G., & Makhoul, J. (1994). On Using Written Language Training Data for Spoken Language Modeling. In *Proceedings of the Workshop on Human Language Technology* (pp. 94–98). Stroudsburg, PA, USA: Association for Computational Linguistics. http://doi.org/10.3115/1075812.1075830

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using Context-Dependent Deep Neural Networks. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (pp. 437–440). http://doi.org/10.1124/pr.110.003533.nome-disease

Seltzer, M. L., & Acero, A. (2007). Training Wideband Acoustic Models Using Mixed-Bandwidth Training Data for Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(1), 235–245. http://doi.org/10.1109/TASL.2006.876774

Senior, A., Sak, H., & Shafran, I. (2015). Context dependent phone models for LSTM RNN acoustic modelling. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (Vol. 2015–Augus, pp. 4585–4589). http://doi.org/10.1109/ICASSP.2015.7178839

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016). Berlin, Germany.

Smit, P., Virpioja, S., & Kurimo, M. (2017). Improved Subword Modeling for WFST-Based Speech Recognition. In *Proc. Interspeech 2017* (pp. 2551–2555). http://doi.org/10.21437/Interspeech.2017-103

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, *147*(1), 195–197.

Steinberger, R., Eisele, A., Klocek, S., Pilos, S., & Schlüter, P. (2012). DGT-TM: A freely available Translation Memory in 22 languages. *Lrec*, 454–459. Retrieved from http://www.mt-archive.info/LREC-2012-Steinberger.pdf

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, *8*(3), 185–190.

Stoyanchev, S., Salletmayr, P., Yang, J., & Hirschberg, J. (2012). Localized detection of speech recognition errors. In *2012 IEEE Workshop on Spoken Language Technology, SLT 2012 - Proceedings* (pp. 25–30). http://doi.org/10.1109/SLT.2012.6424164

Stolcke, A., & Droppo, J. (2017). Comparing Human and Machine Errors in Conversational Speech Transcription. In *Proc. Interspeech 2017* (pp. 137–141). http://doi.org/10.21437/Interspeech.2017-1544

Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Hakkani, D., Plache, M., … Lu, Y. (1998). Automatic detection of sentence boundaries and disfluencies based on recognized words. *ICSLP*. http://doi.org/10.1.1.53.7127

Strigins, J. (2015). Valodas modeļi latviešu runas atpazīšanas sistēmai (ENG: Language models for Latvian speech recognition system).

Su, H., Li, G., Yu, D., & Seide, F. (2013). Error back propagation for sequence training of Context-Dependent Deep Networks for conversational speech transcription. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (pp. 6664–6668). http://doi.org/10.1109/ICASSP.2013.6638951

Tiedemann, J. (2009). News from OPUS — A Collection of Multilingual Parallel Corpora with Tools and Interfaces. *Recent Advances in Natural Language Processing*, *V*, 237–248.

Tilk, O., & Alumäe, T. (2015). LSTM for punctuation restoration in speech transcripts. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (Vol. 2015–January, pp. 683–687).

Tu, K., & Honavar, V. (2008). Unsupervised learning of probabilistic context-free grammar using iterative biclustering. *Grammatical Inference: Algorithms and Applications*, 224–237. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-88009-7_18

Varavs, A., & Salimbajevs, A. (2018). Restoring Punctuation and Capitalization Using Transformer Models. 6th International Conference on Statistical Language and Speech Processing (SLSP 2018), (in press).

Vasilescu, I., Adda-Decker, M., & Lamel, L. (2012). Cross-lingual studies of ASR errors: paradigms for perceptual evaluations. *Lrec 2012 - Eighth International Conference on Language Resources and Evaluation*, 3511–3518.

Vesely, K., Ghoshal, A., Burget, L., & Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Interspeech* (pp. 2345–2349).

Virpioja, S., Smit, P., Grönroos, S.-A., & Kurimo, M. (2013). Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. *Aalto University Publication Series SCIENCE + TECHNOLOGY*, *25/2013*.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *Acoustics, Speech, and Signal Processing, IEEE Transactions on*. http://doi.org/10.1109/29.21701

Whittaker, E. W. D. (2000). *Statistical Language Modelling for Automatic Speech Recognition of Russian and English*. University of Cambridge.

Wohlin, C., Höst, M., & Henningsson, K. (2003). Empirical research methods in software engineering. In Empirical methods and studies in software engineering (pp. 7–23). Springer.

Wu, J., & Chan, C. (1993). Isolated word recognition by neural network models with cross-correlation coefficients for speech dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(11), 1174–1185.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., … Zweig, G. (2016). Achieving human parity in conversational speech recognition. *arXiv Preprint arXiv:1610.05256*.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., & Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)* (pp. 189–194). http://doi.org/10.1109/SLT.2014.7078572

Yao, K., Zweig, G., Hwang, M., Shi, Y., & Yu, D. (2013). Recurrent Neural Networks for Language Understanding. *Interspeech*, (1), 104–108. http://doi.org/10.13140/2.1.2755.3285

Yu, D., Xiong, W., Droppo, J., Stolcke, A., Ye, G., Li, J., & Zweig, G. (2016). Deep convolutional neural networks with layer-wise context expansion and attention. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (Vol. 8-12-NaN-2016, pp. 17–21). http://doi.org/10.21437/Interspeech.2016-251

Yu, G., Russell, W., Schwartz, R., & Makhoul, J. (1990). Discriminant analysis and supervised vector quantization for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on* (pp. 685–688). http://doi.org/10.1109/ICASSP.1990.115850

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., & Courville, A. (2017). Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv Preprint arXiv:1701.02720*.

Znotiņš, A., Polis, K., & Darģis, R. (2015). Media Monitoring System for Latvian Radio and TV Broadcasts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

APPENDICES

1. appendix. Papers Selected for Review and Analysis

Authors	Title	Venue\Journal	Researched language
G Raškinis, D Raškinienė	Building medium-vocabulary isolated-word lithuanian hmm speech recognition system	Informatica, 2003	Lithuanian
P Kasparaitis	Lithuanian speech recognition using the English recognizer	Informatica, 2008	Lithuanian
T Alumäe	Large vocabulary continuous speech recognition for Estonian using morphemes and classes	Text, Speech and Dialogue, 2004	Estonian
I Oparin, L Lamel, JL Gauvain	Rapid development of a Latvian speech-to-text system	IEEE Internataional Conference on Acoustics, Speech and Signal processing, 2013	Latvian
M Filipovič, A Lipeika	Development of HMM/Neural Network-Based Medium-Vocabulary Isolated-Word Lithuanian Speech Recognition System	Informatica, 2004	Lithuanian
A Lipeika, J Lipeikienė, L Telksnys	Development of isolated word speech recognition system	Informatica, 2002	Lithuanian
T Alumäe, E Meister	Estonian Large Vocabulary Speech Recognition System for Radiology.	Baltic HLT, 2010	Estonian
T Alumäe, L Võhandu	Limited-Vocabulary Estonian Continuous Speech Recognition System using Hidden Markov Models	Informatica, 2004	Estonian
A Ragni	Initial Experiments with Estonian Speech Recognition	16th Nordic Conference of Computational Linguistics NODALIDA-2007	Estonian
G Bartisiute, K Ratkevicius	Speech server based Lithuanian voice commands recognition	Elektronika ir Elektrotechnika, 2012	Lithuanian