Lithuanian spontaneous speech recognition

Students: Rita Beigaitė, Greta Ciganaitė, Agnė Keršytė, Povilas Pažėra
Supervisors: Jurgita Kapočiūtė-Dzikienė, Tomas Krilavičius[1]

Abstract. This paper presents the results obtained when building a continuous speech recognizer for the Lithuanian language. The recognizer was built using the HTK toolkit based on Hidden Markov Models (HMMs). The "Kalba Vilnius" interview-based speech database was used for training the model and the testing data was obtained from 4 other speakers. The triphones-level acoustic model was used for the system. The word error rate of the recognizer was always above 100%. There are a number of factors that contributed for such inaccurate recognition. However, the most likely factor is that both, the acoustic and the language models were created for automatic read speech recognition, thus were not spontaneous speech specific.
Key words: spontaneous speech, automatic speech recognition, Hidden Markov Model, the Lithuanian language.

I. INTRODUCTION

Nowadays, efficient automatic speech recognition (ASR) systems can reach very high accuracy for the isolated, small vocabulary speech. The accuracy decreases when the task changes from isolated to continuous speech with a large vocabulary. However, when the task is to recognize read (or prepared) speech (clearly pronounced, without any disfluencies and interjections), the recognition accuracy rates decrease, although not dramatically. The spontaneous speech, on the other hand, raises a significant challenge for ASR systems. In the everyday conversational or spontaneous speech, people often fill their pauses with hesitation sounds, mispronounce words or do not pronounce them enough to make them intelligible (e.g., mumble). Such features pose a problem for creating acoustic and language models required for the specific task of automatic spontaneous speech recognition (Furui et al., 2005). However, since most of our speech is spontaneous, the task of automatic spontaneous speech recognition becomes vital for the improving the human-computer interaction.

The development ASR for the Lithuanian language is still at a very early stage. Experiments with the Lithuanian ASR systems for prepared speech can achieve quite high accuracy rates, depending on the used speech corpora, vocabulary sizes, models, and tools. However, to the best of our knowledge, experiments on with the spontaneous Lithuanian speech have not been reported yet.

[1] The names of students and supervisors are listed in the alphabetical order.

II. RELATED WORK

Lithuanian language has only around ~3.2 million speakers world-wide, providing a very limited amount of speech data available and, thus, being one of the languages chosen by the IARPA BABEL research program (Harper, 2013; Lileikytė et al., 2015) that focuses on broadcast speech data for the development of ASR systems. Not surprisingly, Lithuanian has been used as a test language in various studies aiming to find effective low-resource training methods for such speech recognition (Mendels et al., 2015; Gales et al., 2015; Davel et al., 2015; Fraga-Silva et al., 2015; Grézl et al., 2015). One of the most successful attempts related to broadcast speech recognition was carried out within Qaero research program (Lileikytė et al., 2016), where only 3 hours of transcribed and 360 hours of untranscribed data used for training of semi-supervised acoustic model resulted in an extremely low word error rate (WER) of 18.3%. Similar work carried out by Estonian researchers that had adapted their own language ASR system to Lithuanian broadcast audio records (based on Kaldi ASR toolkit) gave even smaller WER, equal to 14.7%, though the amount of training data was significantly higher (90 hours of wide-band data from various Lithuanian TV, radio, and on-line media channels). In comparison, system created by "Google" was able to achieve an error rate of 40.74% (using 86 hours of recordings) (Sipavičius and Maskeliūnas, 2016). However, despite accurate recognition systems built in the studies described above, general Lithuanian speech recognition task is far from achieved, as all of them lacked spontaneous speech data (e.g., the data set used by "Google" included only 0.04% of spontaneous speech). Consequently, system capable of recognizing such speech is yet to be constructed.

III. DATA

For the training set of the spontaneous speech recognition system, "Kalba Vilnius" speech corpus was used, from which we selected 1 hour and 37 minutes of audio recordings. The recordings included 6 speakers (3 interviewers and 3 interviewers) in total. However, since the corpus was interview-based, the three interviewees uttered more words that the interviewers during every recording. The 6 speakers were all females. All of the audio from "Kalba Vilnius" was recorded using a dictaphone.

The testing data consisted of the recordings of 4 female speakers, recorded with different types of microphones (computer, tablet, and smartphone built-in microphones). The questionnaire of the "Kalba Vilnius" project was used when recording the testing data in order to avoid large vocabulary differences between training and testing sets. The recordings totaled to 55 minutes of speech.

A corpus of read speech was included later on, to test whether training on read speech and testing on continuous speech would improve the accuracy of the recognition. The corpus was composed of 88 minutes of speech from 1 female speaker, recorded on a laptop built-in microphone.

All of the recordings were done in a relatively quiet room.

IV. METHODOLOGY

In order to build a simple spontaneous speech recognizer, it was decided to use Hidden Markov Model Toolkit (HTK), which was originally developed at Cambridge University. The toolkit is written in C and consists of modules and tools used for speech analysis or training/testing hidden markov models (HMMs). The applicability of HTK is quite general (including natural language processing or other similar tasks), though it is most commonly used for speech recognition research.

The overall aim of an ASR system is to convert an acoustic signal representing the speech into a string that contains corresponding words. The conversion process could be split into three main stages: preparing the data (4.1), training the HMMs (4.2), and decoding (recognizing) the word strings (4.3) (Jurafsky and Martin, 2009, Ch. 9, p. 4). All of the stages are described in the subsections given below, followed by recognition evaluation description in the last one (4.4). The general architecture of an ASR system (with correspondence to HTK toolkit modules) is visualized in Figure 1:

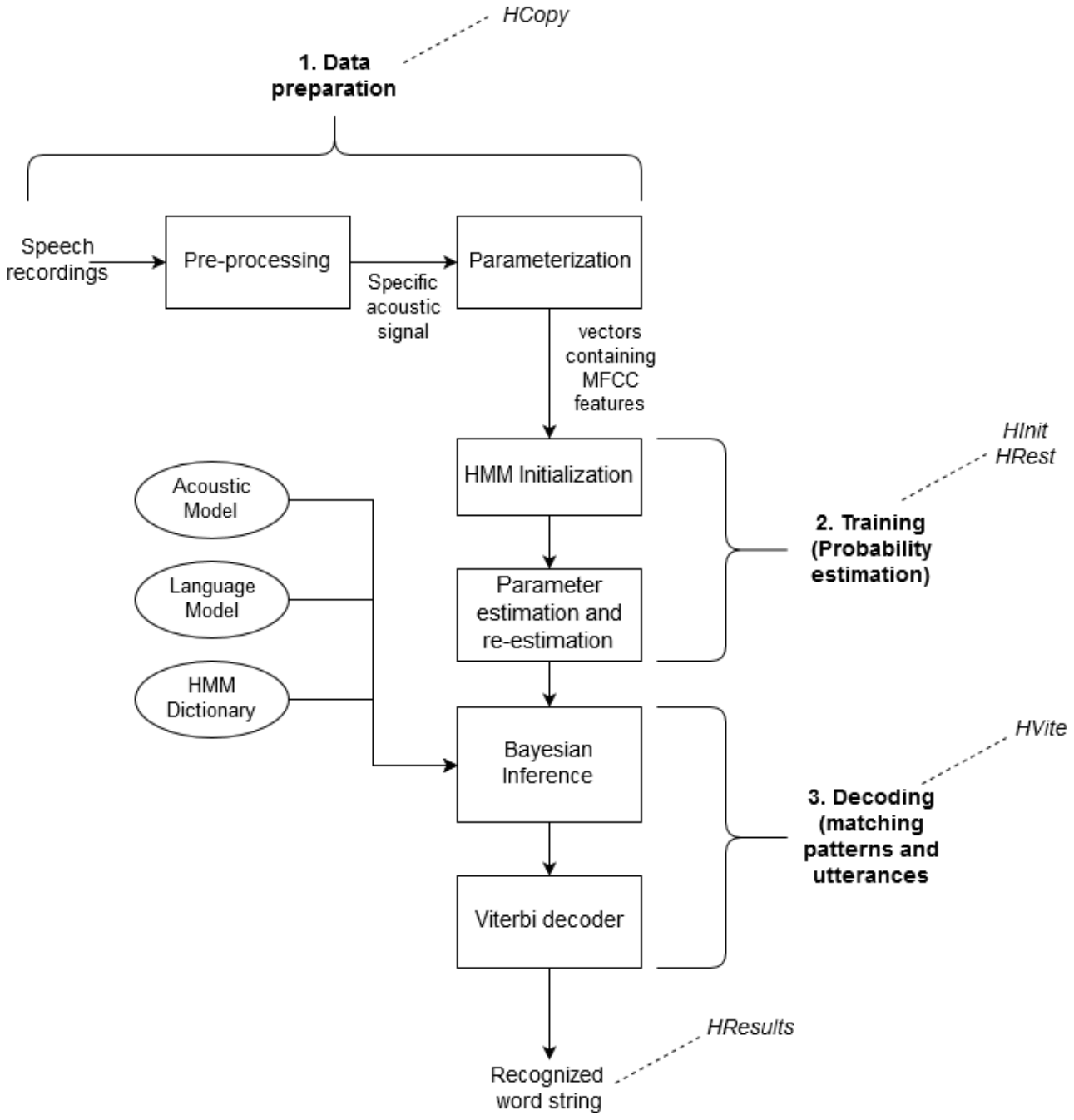

Figure 1. General ASR system's architecture. Main stages of recognition process are given in bold, and the related HTK modules are linked to them with dotted lines. The rectangles represent the main tasks done in each of the stages, and the circles display objects used for Bayesian inference procedure in the decoding step.

4.1. Data preparation

The first step of speech recognition process is to prepare the data, which includes its pre-processing and parameterization (also known as feature extraction). The pre-processing stage is simply recording the data, segmenting it into training and testing sets, and labeling it with related strings of word in order to test the recognizer's accuracy in the later recognition steps. The more important part here is the data parameterization, during which, a sequence of spectral features of the

input waveform are extracted in the form of vectors for every small window of the given speech signal (in HTK done using module *HCopy*). In order to achieve recognition of high accuracy, these features should represent the phonetic content of the speech precisely, also being able to exclude data that is not relevant for recognition (such as phase or pitch of the signal). Most commonly used representation for spectral features of speech (as well as the one used in the present study) is Mel Frequency Cepstral coefficients (MFCCs) (Jurafsky and Martin, 2009, Ch. 9, p. 7). The process of extracting the MFFC features is visualized as a flow chart in the Figure 2 and described in more detail below. In the present study, the parameterization was performed using HTK toolkit command *HCopy*.

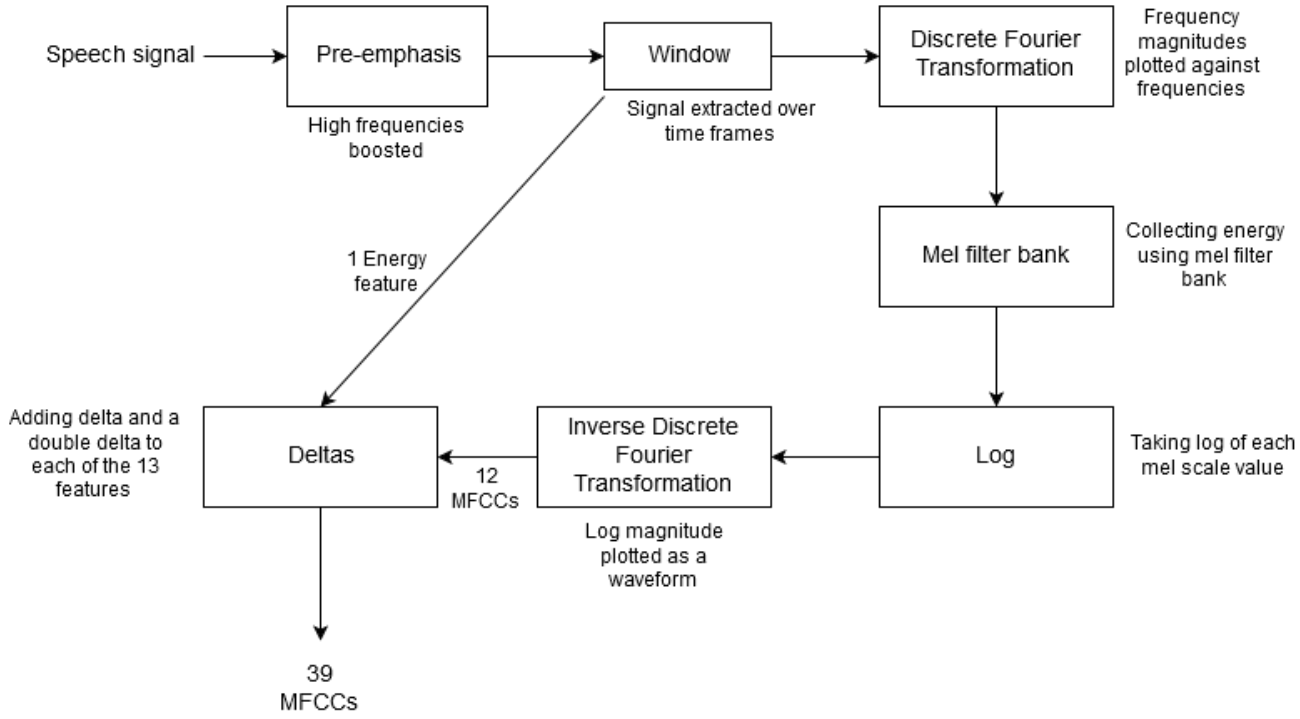

Figure 2. The process of MFCC features extraction.

The MFCC parameterization process is started with speech signal ***pre-emphasis***, where the energy amount is increased for the higher frequencies of the signal with a high-pass filter. That is done to account for the spectral tilt caused by the glottal pulse, which results in higher amount of energy contained in the lower frequency bands of the spectrum obtained from voiced segments of speech (Jurafsky and Martin, 2009, Ch. 9, p. 13). Applying the high-pass filter then allows retrieving the information contained in the relatively suppressed higher frequency bands.

At this point, ***windowing*** procedure can be performed. Here, acoustic observations are extracted from the recorded speech for overlapping time frames (with an offset of around 10ms). The time frames, also called windows, are of equal length, typically about 25ms, allowing to

acquire information from a signal that is stationary enough to represent distinctive sound units forming the whole utterance (see Figure 3).

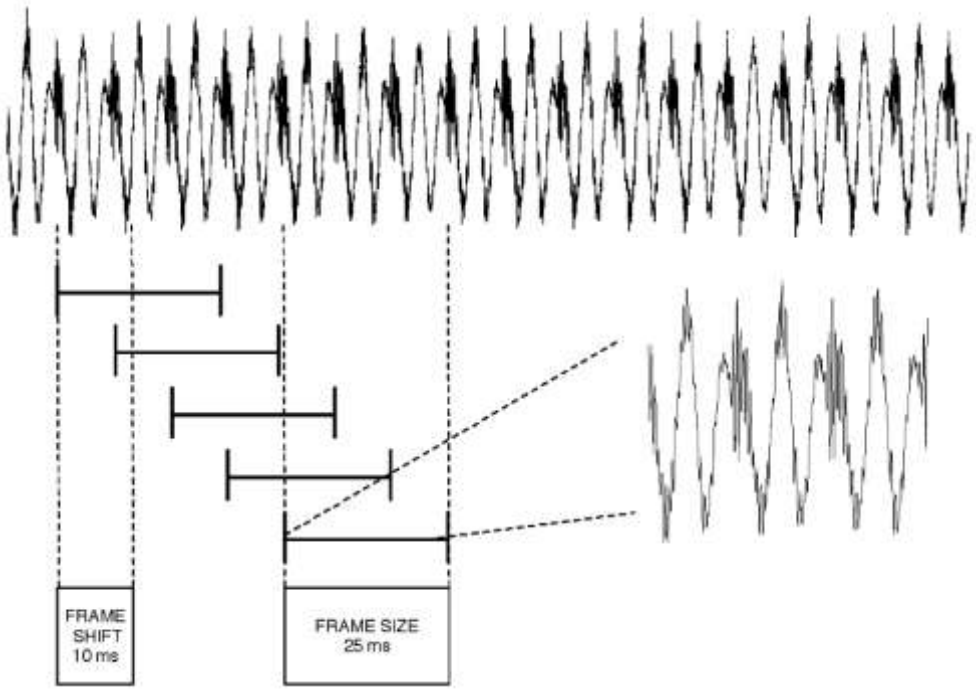

Figure 3. The windowing process. Here, a rectangular window of 25ms and offset of 10ms (Jurafsky and Martin, 2009, Ch. 9, p. 14).

Using a simple rectangular window does not suffice for the extraction of MFCC due to rough cutting of the signal which results in formation of sound clicks. Such clicks would impair the following procedures used in parameterization, specifically, Fourier analysis, thus, a different kind of window is usually used. A Hamming window makes the values of the signal smaller in the start and the end of the window, creating a fade out that solves the mentioned problem (see Figure 4).

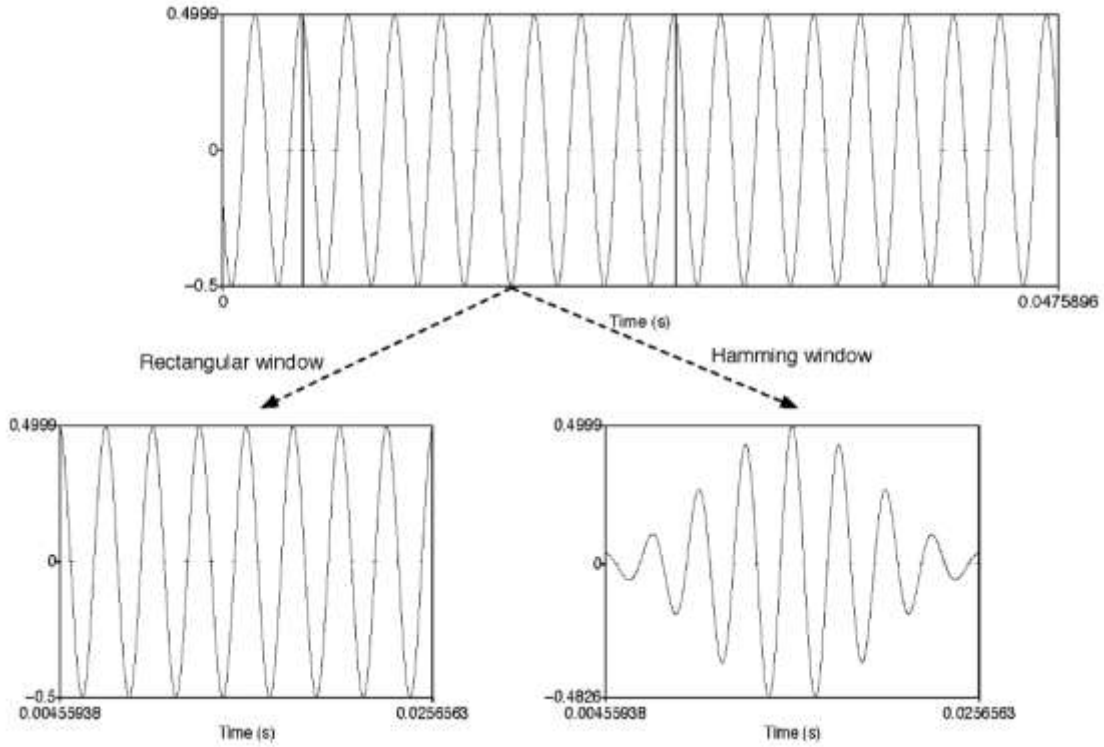

Figure 4. Using rectangular (left) and Hamming (right) windows for windowing.

All of the windowed signals are then transformed into complex numbers that represent the magnitudes and phases of the frequency components as in the original signal for all of the discrete frequency bands (see Figure 5). The procedure is called ***Discrete Fourier Transform (DFT),*** and in HTK it is implemented in the form of Fast Fourier Transform (FFT).

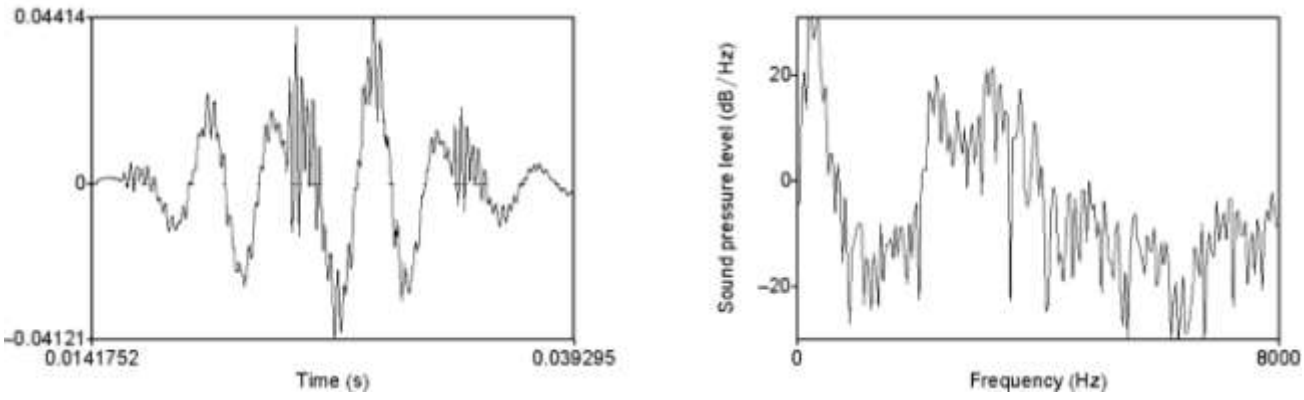

Figure 5. A signal (on the left) and its spectrum after DTF calculation (within a 25ms Hamming window) (Jurafsky and Martin, 2009, Ch. 9, p. 16).

The DFT procedure is extremely important, as knowing how much of energy is contained at each frequency band helps to distinguish different phonetic content. However, it is also known that the human ear perceives sounds of different frequencies differently. For example, it is less sensitive

to frequencies that are above 1000Hz. Consequently, it seems that modeling such property might improve the recognition even more, and that is accounted for in the next step of parameterization, mapping the DFT output ***onto mel scale,*** where perceptually equally distant pitch sounds are separated by the same amount of mel units. The procedure is done by creating a bank of band pass filters of non-uniform frequency band widths, where such non-uniformity is based on mapping between frequencies in Hertz and the mel scale (see Figure 6). To be specific, the mapping is linear below 1000Hz and logarithmic above, representing the perceptual inequalities of different frequencies. Applying this procedure allows to collect the relative amounts of energy required from each of the frequency bands.

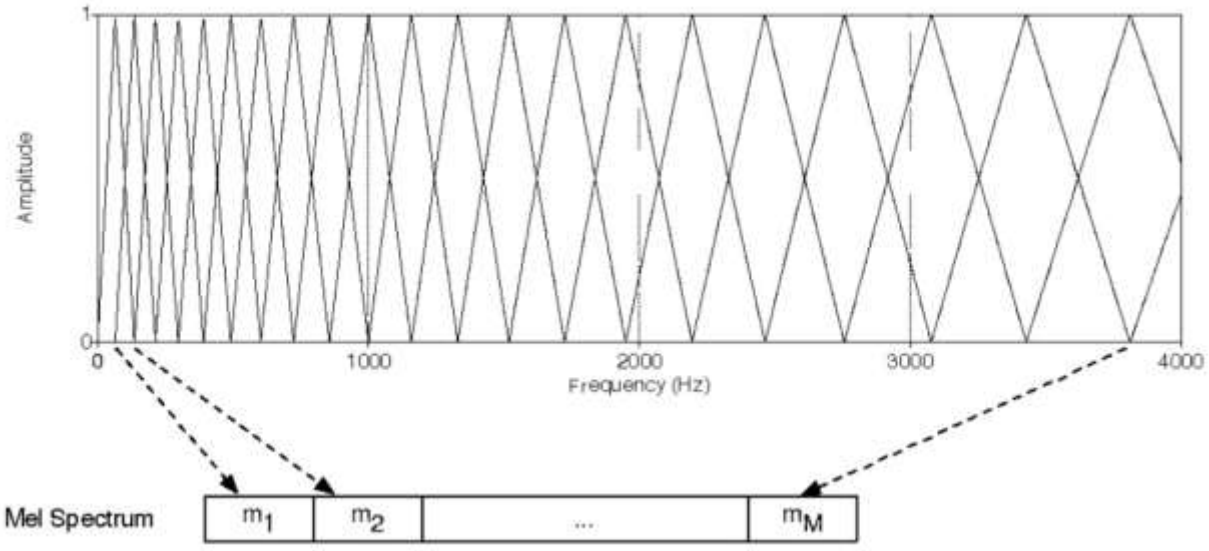

Figure 6. The Mel filter bank. Each triangular filter collects energy from a different frequency band, spreading them linearly below 1000Hz and logarithmically above (Jurafsky and Martin, 2009, Ch. 9, p. 17).

At this point, a ***log*** of each of the values acquired using the mel spectrum is taken, making the upcoming feature estimation (performed in the later stages of recognition) less prone to differences of input waveform, where such difference may occur due to simple reasons, such as varying distances from the speaker to the microphone used for recordings of the data.

During the next parameterization step, the ***inverse DFT*** of the log magnitude of the signal is taken (Jurafsky and Martin, 2009). The procedure is based on the notion that the speech waveform is created by a glottal source waveform with some specific fundamental frequency, which is the passed through the vocal tract that has filtering characteristic (depending on its shape). The filter data contains features useful for recognition, and the main idea of this procedure is to segment it from recognition-irrelevant source data. That results in a spectrum, where the first values are most definitive of the filter and, thus, carrying most significance. Typically, 12 of such cepstral values

are taken (Gruhn et al., 2011), with ***energy*** of the frame added as a 13th feature due to its correlation with phone identity.

The final step of parameterization finalizes the feature set by adding ***delta*** and ***double delta*** values (corresponding to velocity and acceleration, respectively) for each of the 13 features acquired before, which allows to account for the change of spectral features over time due to speech signal variation from frame to frame. The finalized set then consists of 39 MFCC features for each of the windowed signals. For the whole speech segment, that is a sequence of 39-dimensional feature vectors corresponding to recognition-relevant spectral information.

4.2. Training

After the data is parameterized, the HMM model can be trained. The key components of the process are described in detail below:

- The **model**. The HMM model was selected due to its computational simplicity. The model is regarded as computationally simple as each observation's probability is only dependent on the state that the observation is in at a certain point in time without taking into account the state that it was in before or the state that it transitions to after. Moreover, the transition probabilities only depend on only the most recent preceding state.

  The prototype HMM model we used consisted of 5 states, of which the first and the last states are "dummy", meaning that they do not emit any observations. Thus, the first state is always of probability 1 and the last state is always of probability 0. All of the transition probabilities of the model are represented in a transition probability matrix. Each emitting state has an emission probability of it generating a particular observation.

  Since the phonemic model does not account for any between-words silences (due to the fact that there are no silences within words), a short pause (silence) HMM was also created, consisting of 3 states, of which only one was an emitting state. This was done using the HTK HHEd tool.
- The **acoustic model**. The goal of the acoustic model is to calculate the probability of the observed spectral feature vectors given the phonemes (141 phonemes were used in our system) (Jurafsky and Martin, 2014). The HMM calculates Gaussian probability density functions for each feature vector and stores them in the HMM states. In computing the probabilities for the sequences of observations, an assumption of statistical independence between observations has to be made, which regards observations coming from the same HMM state as independent from one another.

  There are many possible paths (transitions) in the HMM that can generate the same sequence of observations (because self-transitions are also possible). This is why the model is called "hidden" - since there are many possible paths for the same observation sequence, the path that the model

made is unknown. However, this poses a problem for the model since in order to estimate the parameters of each HMM state, we need to know the observation sequences generated by that state.

This problem is solved somewhat indirectly in HTK. Firstly, all of the observations are aligned uniformly with the states (i.e. each observation is aligned with only one state). This process is called uniform segmentation (Young et al., 2000).

Secondly, a dynamic programming algorithm which is able to store intermediate probabilities is applied to align the states with the observations in the most likely way. The algorithm is used iteratively (i.e. each time after the computation is done, the output of it is used as the input of the next cycle). This process is applied to achieve the maximum probability of the HMM states' parameters.

We used the HTK HInit tool to perform the initialization of the models described above.

- **Training**. After the initialization of the models, the system can be trained. The training in HTK is done using Baum-Welch re-estimation. Baum-Welch is an Expectation Maximization algorithm, which operates in two steps. In the expectation calculation step, the transition and emission probabilities are computed given the initial models. In the following step (maximization), the parameters are re-estimated by using the probabilities calculated in the expectation step. The algorithm operates iteratively until, ideally, convergence is reached.

  Training was done using the HERest tool provided by the HTK.

4. 3. Decoding

The HMMs are able to count the probabilities of specific acoustic observations happening given some linguistic unit (e.g., phoneme or word), though that is not enough for classification of the sounds. The real recognition happens in the step after training, called decoding, where the sequence of states that matches the sequence of observations the best is found. This process requires 3 sources of information:

- A **language model**, which typically is an N-gram or fixed grammar that provides the prior probability of a specific linguistic unit (e.g., phoneme, word, or sentence), *P(Unit)*. In the present study, the language model was based on a vocabulary consisting of 3742 words (from the training set). The phonetic transcriptions of the words were also obtained, using grapheme-to-phoneme algorithm written by prof. Gailius Raškinis.
- An **HMM dictionary**, having probabilities of certain acoustic observations being realized as specific strings of linguistic units, *P(Observations|Unit).*
- An **acoustic model** that contains the probabilities of certain units being realized as a specific sequence of acoustic observations, *P(Observations).*

The information acquired from these sources is then used to derive the sound classification metric, which is the probability of some linguistic unit given an acoustic observation, *P(Unit|Observation)*, by applying the Bayes' rule:

$$P(Unit|Observations) = \frac{P(Observations|Unit)P(Unit)}{P(Observations)}$$

At this point, a Viterbi algorithm is employed, which searches through the space of all possible sequences and chooses the most likely one as the result of the recognition. As the search space is huge, it is often used with additional procedures, such as pruning (e.g., checking only the observation sequences that are likely to match the given unit), that would reduce the computational complexity of the process (Gruhn et al., 2011).

In HTK the decoding is performed in two stages. Firstly, manually written grammars are converted into finite state language models using the ***HParse*** module, and later, ***HVite*** is used to load a single network consisting of mentioned language model, trained HMMs, and a transcription dictionary. Here, the Viterbi algorithm is employed then in the form of token passing, where every token corresponds to a possible path via the model. The tokens are moving through states along the HMM arcs and store the path that they went through so far, as well as the total probabilities of such partial paths. After reaching the last state, the token holding highest probability is chosen. The process efficiency is increased by discarding tokens whose total probability is below some pre-set threshold at a specific partial path (Young et al., 2006). Finally, a set of labels containing the strings of recognized speech units is obtained.

The training and decoding processes for a simple isolated digit recognition task, as performed in HTK, are given in the Figure 7 (the general architecture of the procedures used is similar to that of bigger utterances, such as sentences).

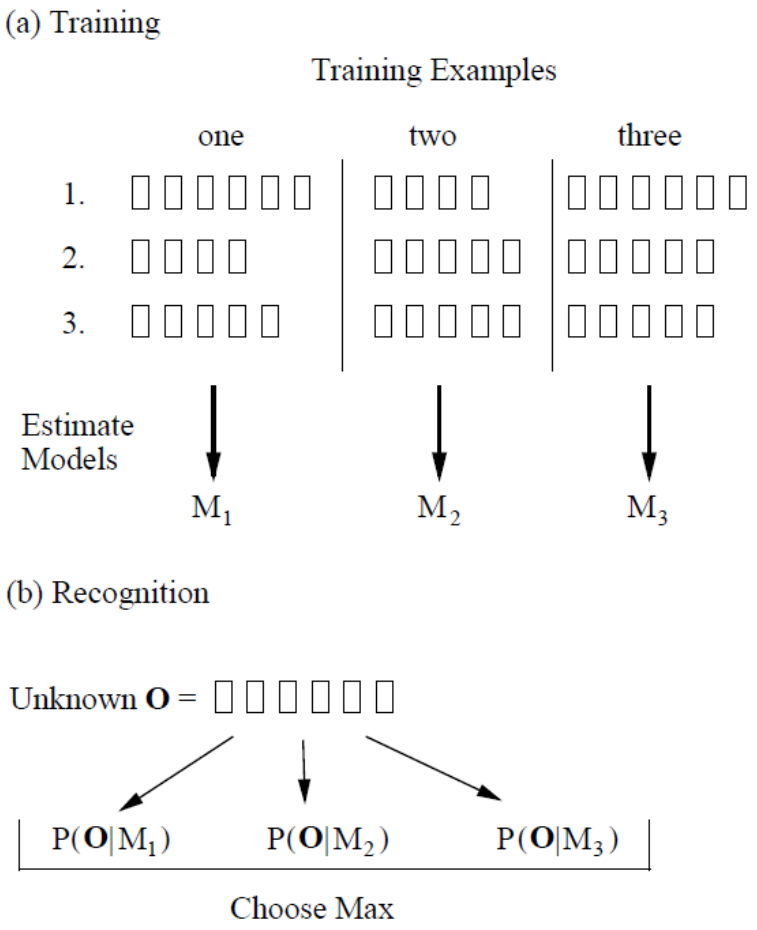

Figure 7. Training (a) and decoding (b) stages as performed in HTK for isolated digit recognition. The general framework remains the same for recognition of various linguistic units. (Young et al., 2006, p. 5)

4.4. Evaluation

The main types of mistakes made by recognition systems are substitutions, deletions, and insertions, where the type of the error is determined by comparison between the recognition and the actual transcription (hand-labeled). If the linguistic unit recognized is different from the actual one, the error type is substitution, if the unit is just omitted, it is deletion, and if the recognized unit was not actually spoken (a false positive), it is insertion. The error rate is then calculated by summing the count of all the errors and dividing the obtained number by the amount of entries at all.

In HTK, the module responsible for the analysis of the recognition is ***HResults,*** which aligns and matches the output given by *HVite* with the correct transcriptions, as well as reports the performance in terms of recognizer's accuracy and the amount of errors for each possible type.

V. EXPERIMENTS AND RESULTS

Experiment No. 1.

In this experiment the system was trained and tested on the spontaneous speech. The dictionary, which was used in the recognition phase, consisted of 3742 distinct words which occurred in the audio files used for training.

All four distinct audio files were tested. An average of ~20 minutes was required for the execution of the speech recognition system. The results of the experiment are presented in Table 1.

Table 1. The results of a small scale automatic speech recognition system when it was trained and tested on spontaneous speech.

Speaker	Microphone type	Uttered words	Recognised words	Recognised words, %	Substitutions	Deletions	Insertions	WER, %
Speaker 1	Computer (built-in)	769	70	9,10	697	2	1851	331,6
Speaker 1	Mobile phone (built-in)	769	44	5,72	725	0	1477	286,3
Speaker 1	Tablet (built-in)	769	50	6,50	719	0	1526	291,9
Speaker 2	Computer (built-in)	291	12	4,12	279	0	608	304,8
Speaker 2	Mobile phone (built-in)	291	20	6,87	271	0	1071	461,2
Speaker 3	Computer (external)	1151	82	7,12	1068	1	2520	311,8
Speaker 4	Mobile phone (built-in)	916	33	3,60	880	3	801	183,8

The percentage of correctly recognized words was extremely low (it did not reach even 10% for any of the speakers). The total word error rate (WER) of each test was significantly above 100%, with the lowest WER being ~183.8%. Such low WERs are largely caused by enormous number of insertion and substitution errors. However, the system demonstrated a very poor performance in general. There was no significant change in WER even when the value of the insertion penalty was increased when carrying out the testing using the HVite tool. The poor performance of the system might be due to a number of factors:

- **Microphone types.** The microphones used in the training set were dictaphones whereas the testing was done on other type of microphones. Different types of microphones vary in the amount of distortion they add to the sound signal (Wolf & Nadeu, 2014), thus when the training and testing data sound signals varied in the amount and type of the added distortion, the recognition rates of the system were almost certain to decrease. However, this problem could not be rectified due to insufficient dictaphone resources available to the authors.
- **Accent types.** The pronunciation of words (and, therefore, phonemes) varies with accent even for speakers of the same language. Our training data set consisted of Lithuanian-Russian and Lithuanian-Polish bilinguals as well as monolinguals from the same city. Thus, there was already quite a lot of accent variability in the training corpus. However, the testing data set included speakers from another city of Lithuania, thus including another dialect. Such variability in pronunciations was also a definite factor to impair the recognition significantly. However, accent variability was also unavoidable due to a lack of speech corpus accessible to us.

- **Training data size.** It has been shown that speech recognition is better when the system is trained on a large training set as large sets diminish the affect of individual acoustic particularities (e.g., the particularities of individual speakers' pronunciations, effects of microphone types, etc.). Our training set consisted of only ~97 minutes of speech and only 6 speakers, thus the training set was extremely small.
- **Speaker mismatch.** Higher rates of speech recognition are always achieved in speaker-dependent systems (i.e. when the training and testing is done on the same speaker). However, our system was used for speaker-independent recognition, thus somewhat lower recognition rates were to be expected.

Experiment No. 2.

This experiment was done in two parts, for differing speech types in the training data set. The experiments were executed to examine whether the huge amount of substitution and insertion errors can be reduced by changing the recognition dictionary. One audio recording, a minute in length, was used for testing. 93 words were uttered in the recording.

- The first part of this experiment was carried out when the system was trained on spontaneous speech. The following dictionary changes were made to the system trained on spontaneous speech:
    - Firstly, the initial spontaneous speech dictionary was used. Afterwards, all meaningless words (filled hesitation pauses like 'mmm' and unfinished words) were excluded from the dictionary. Thus, the amount of words in the dictionary decreased from 3742 to 3616.
    - The words uttered in the test data audio file were added to the initial dictionary. The new dictionary consisted of 3762 distinct words. Afterwards, all meaningless words were excluded from the newly created dictionary. The amount of the deleted meaningless words was 130, thus the total amount of words in the dictionary was 3632.
    - Only the words uttered in the testing data were left in the dictionary. The total amount of words was 76. Afterwards, 3 meaningless words were excluded of this dictionary.

  Table 2 presents the results of the spontaneous speech recognition system when the dictionary was altered and the system was trained on spontaneous speech.

  The results show that using only the words uttered in the testing data increases the percentage of correctly recognized words and decreases the WER. This is to be expected as the dictionary in this case is specified to the task at hand (i.e. recognizing only particular words). In all cases (when the dictionary consist of only the training words, only the testing words and words from both samples), excluding meaningless words decreases the WER as less insertion errors are

made but the amount of correctly recognized words decreases as well. This is probably due to the fact that the words that are inserted most are also recognized best.

Table 2. Variation in WER of a spontaneous speech recognition system trained and tested on spontaneous speech

Dictionary	Uttered words	Recognised words	Recognised words, %	Substitutions	Deletions	Insertions	WER, %
Words from the training data	93	7	7,53	86	0	183	289,2
Excluding meaningless words	93	5	5,38	88	0	77	177,4
Words from the training and testing data	93	7	7,53	86	0	183	289,2
Words from the training and testing data excluding meaningless words	93	5	5,38	88	0	77	177,4
Words from the test data	93	18	19,35	75	0	134	224,7
Words from the test data excluding meaningless words	93	14	15,05	78	0	55	143

- Another part of this experiment was done when the recognizer was trained on the read speech. The following dictionary changes were made for the system trained on the read (prepared) speech:
    - Firstly, the whole prepared speech dictionary was used. The amount of words in the dictionary was 3051.
    - The words uttered in the testing data were added to the dictionary. The new dictionary consisted of 3089 words.
    - Only the words uttered in the testing data file were left in the dictionary. The total amount of words was 76. Afterwards, 3 meaningless words were excluded from the dictionary.

  Table 3 presents the results of the spontaneous speech recognition system when the dictionary was altered and the system was trained on the read speech.

  The table shows that, overall, training the system on prepared speech and testing it on spontaneous speech results in lower WERs and significantly lower insertion errors. This is most likely caused by the fact that read speech contains less interjections, vocalized hesitations, etc, which are the words most likely to be inserted. The pattern of the results with dictionary variations was very similar to that of the previous experiment.

Table 3. Variation in WER of a spontaneous speech recognition system trained on read speech and tested on spontaneous speech.

Dictionary	Uttered words	Recognised words	Recognised words, %	Substitutions	Deletions	Insertions	WER, %
Words from the training data	93	3	3,23	90	0	81	183,8
Words from the training and testing data	93	6	6,45	85	2	88	188,1
Words from the test data	93	18	19,35	74	1	37	120,4
Words from the test data excluding meaningless words	93	14	15,05	73	6	13	98,9

VI. DISCUSSION AND CONCLUSIONS

In the present study, it was attempted to build a large vocabulary speech recognition system for Lithuanian language. The recognizer's accuracy was found to be extremely low, though the system could possibly be improved drastically by application of a set of certain measures.

We hypothesize that enormously high amount of substitution and insertion errors happened due to oversimplified language model which should help to control for such errors. In the present study, the used model was a fixed grammar, and the probabilities of each word happening were uniform. Not surprisingly, a lot of uncommon insertions and substitutions happened, where most were found to be shorter words, implying that the recognizer would insert words sounding similar to parts of longer words (e.g., instead of recognizing the right word, three smaller words that sound similar to the word of interest when combined are inserted). That could be easily avoided using an N-gram language model, which would eliminate such errors by not allowing the recognizer to pick such unlikely sequences. To put it simply, it would allow to model the words' context-dependency. What is even more, a similar approach could be used for modeling context-dependency in a phonemic level, and combining both of them should decrease the amount of substitution and insertion errors even more.

Another improvement could be done in the parameterization process. For example, applying cepstral mean subtraction procedure (CMS) could possibly remove linear filter effect that may happen due to different microphones used for the recordings of training and testing data, leading to more effective training and decoding stages of recognition (Gruhn et al., 2011).

The decoding process could be made more effective by using a more extensive scoring function when rating the possible candidates for recognition result during the Viterbi algorithm implementation. For example, adding an additional source for information, such as a language model of higher-order, or trying to minimize the Bayes risk (while regarding all possible transcriptions)for every step of Viterbi, instead of choosing the source sentence with maximal probability, would allow to take into account more context when making the decision for the best match string. However, these are much more computationally demanding, and that should have to be accounted for (Gruhn et al., 2011).

Finally, even though the HMM models are fairly simple and easy to use, as well as being able to provide a framework that is abstract enough for representation of temporal aspects of speech, they are often build following many simplifying assumptions that are not precisely reflective of how humans recognize speech. Therefore, it was hypothesized, that for large vocabulary speech recognition tasks, some other techniques might be more effective. Specifically, deep learning has been found to be quite efficient and less sensitive to data, often able to result in better recognition for small amounts of training data. Therefore, the idea of changing the general framework of the system is also considered for the future work.

ACKNOWLEDGEMENTS

We acknowledge prof. Gailius Rašknis (Vytautas Magnus University) for converting our dictionary words into their phonetic transcriptions.

REFERENCES

Davel M., E. Barnard, C. v. Heerden, W. Hartmann, D. Karakos, R. Schwartz, and S. Tsakalidis, Exploring minimal pronunciation modeling for low resource languages. *Interspeech*, 2015.

Fraga-Silva T., A. Laurent, J.-L. Gauvain, L. Lamel, V.-B. Le, and A. Messaoudi. Improving data selection for low-resource STT and KWS. *ASRU*, 2015.

Furui S., M. Nakamura, T. Ichiba, and K. Iwano. Why is the recognition of spontaneous speech so hard? *International Conference on Text, Speech and Dialogue*, 2005, pp. 9-22.

Gales M. J., K. M. Knill, and A. Ragni. Unicode-based graphemic systems for limited resource languages. *ICASSP*, 2015.

Grézl F., E. Egorova, and M. Karafiát. Study of large data resources for multilingual training and system porting. *SLTU*, 2016.

Gruhn, R. E., W. Minker, and S. Nakamura. *Statistical Pronunciation Modeling for Non-Native Speech Processing*. Springer Publishing Company, Incorporated. 2011.

Harper M. The BABEL program and low resource speech technology, *ASRU*, 2013.

Jurafsky D., and J. Martin. *Speech and Language Processing, 2nd edition*. Chapter 9, 2009.

Jurafsky, D., and J. H. Martin. *Speech and language processing*. Pearson. 2014.

Lileikytė R., A. Gorin, L. Lamel, J.-L. Gauvain, and T. Fraga-Silva. Lithuanian broadcast speech transcription using semi-supervised acoustic model training. *SLTU*, 2016.

Lileikytė R., L. Lamel, and J.-L. Gauvain. Conversational telephone speech recognition for Lithuanian. *SLSP*, 2015.

Mendel G., E. Cooper, V. Soto, J. Hirschberg, M. Gales, K. Knill, A. Ragni, and H. Wang. Improving speech recognition and keyword search for low resource languages using web data. *Interspeech*, 2015.

Sipavičius D. and R. Maskeliūnas, “Google” Lithuanian Speech Recognition Efficiency Evaluation Research. *Information and Software Technologies*, 2016, vol. 639, pp 602-612.

Wolf M., and C. Nadeu. Channel selection measures for multi-microphone speech recognition. *Speech Communication*, 2014, vol. 57, pp. 170-180.

Young S. J, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book*. Microsoft Corporation and Cambridge University Engineering Department, 2006.

Young S., D. Kershaw, J. Odell, D. Ollason, V. Valtchev and P. Woodland. *The HTK Book Version 3.0*, 2000.