PAN ARCHIVES OF ACOUSTICS
Vol. **39**, No. 4, pp. 501–509 (2014)

DOI: 10.2478/aoa-2014-0054

System for Automatic Transcription of Sessions of the Polish Senate

Krzysztof MARASEK, Danijel KORŽINEK, Łukasz BROCKI

Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warszawa, Poland; e-mail: {kmarasek, danijel, lucas}@pjwstk.edu.pl

(*received January 28, 2014; accepted August 20, 2014*)

This paper describes research behind a Large-Vocabulary Continuous Speech Recognition (LVCSR) system for the transcription of Senate speeches for the Polish language. The system utilizes several components: a phonetic transcription system, language and acoustic model training systems, a Voice Activity Detector (VAD), a LVCSR decoder, and a subtitle generator and presentation system. Some of the modules relied on already available tools and some had to be made from the beginning but the authors ensured that they used the most advanced techniques they had available at the time. Finally, several experiments were performed to compare the performance of both more modern and more conventional technologies.

Keywords: large vocabulary speech recognition, language modelling, transcription, transliteration, subtitles.

1. Introduction

Transcription of Parliamentary speeches is a very common domain in Automatic Speech Recognition (ASR) (Lööf *et al.*, 2006; Pražák *et al.*, 2006; Kos *et al.*, 1996). It is also a first stepping stone in many languages for several reasons: laws of many countries require official parliamentary meetings to be transcribed, thus there is both demand for such a solution and ample supply of publicly available training data.

If we look for examples in other domains that are acoustically similar, the closest would be meeting transcription: there is a possibility of interruptions from other speakers and several people could be speaking concurrently. On the other hand, the vocabulary is very likely to be completely different as compared to Parliamentary speeches. A better overlap, with regards to the vocabulary, may be found in the transcription of speeches or certain kinds of lectures, but considerably different acoustic conditions would be present there, with almost no interruptions and a single speaker. Thus, it can be seen that the chosen task is quite unique. While developing such a system, data from various domains had to be used but the data from the task itself were mainly depended upon.

The state of speech recognition in Polish is still very weak as compared to other languages (Miłkowski, 2012), even though it is improving at a fast pace and Polish should not be considered as an under-resourced language for very long. Several research projects have emerged in the last couple of years dealing with the topics of automation in the telephony environment (Marasek *et al.*, 2009), transcription of legal documents (Demenko *et al.*, 2008), and, recently, speech-to-speech translation in different settings (Marasek, 2012). Commercially, there have been a few local start-ups and a few attempts by world market leaders but none has yet achieved real adaptation of LVCSR in the field, with the exception of a few (Google and Apple/Nuance) which include ASR as a free service with their existing products. This makes it quite difficult for companies and organizations which might seek to use such a technology and find that it either does not fulfill their needs or is too expensive for them to pursue. It has been our goal for many years to contribute to the research on Polish ASR and help bring it to a level of being both useful and attainable by the larger country audience.

2. Data collection and preparation

The main data for this project were acquired by virtue of an agreement between the Polish Senate and the authors' university. The law in Poland mandates that all official parliamentary meetings need to be recorded, transcribed, and publicized. Most of this work is still being done manually with the recordings done with standard recording equipment and then transcriptions done by professional transcription specialists hired and trained specifically for this purpose. The audio quality is often quite poor – the desktop microphones are omni-directional which introduces a lot of environmental distortion (reverberation caused by room materials, background noise), and lossy compression is often used to store the files.

Recently, it has become more difficult to find such transcription specialists and meet the ever growing demand for information in an efficient and speedy manner. These days, all the Senate and Parliamentary meetings are available online in both audio/visual and textual form. Having the technology to automate this process, at least as an aid to the existing workforce, could help both cut costs and make the task more sustainable in the future. Additionally, one could implement a technique called "respeaking", which has been shown as extremely successful in certain situations (Romero-Fresco, 2011). This method works by using an adapted speaker-dependent system to recognize a single, trained professional speaker who repeats the audio that needs to be transcribed. Extremely high accuracy rates (>99%) can be achieved using this method, even with real-time transcription.

A subset of the Senate proceedings' audio recordings with the matching transcripts was chosen for the purpose of training both the acoustic and language model. Initially, data provided directly by the Office of the Senate was used but starting from the beginning of 2013, many recordings have been released on the Senate website and the website of the national television network's channel TVP Parlament. These recordings were of a considerably higher quality, so they were added to our data set. In addition to that, the Polish Parliament data available on the same websites and in much larger quantities was also used.

The preparation of the corpora was a very time-consuming task because the official transcripts were not an exact description of the audio. They were written in a way so as to render grammatically correct written text without changing the semantics of the statement. This was done by the official transcribers in order to improve the presentation of the written PDF document but made it slightly less useful for acoustic training. That is why the data had to be re-transcribed by a group of contract workers. They also removed fragments of audio with high background noise, double-speak and incomprehensible speech, to simplify the transcription process. A large portion of the data was also annotated with information about a speaker (name, gender) to aid the normalization and adaptation properties of the acoustic models.

Alltogether, about 95 hours of recordings from both the Senate and the Parliament, with 488 different speakers (mostly male) were prepared. Out of that, 10 speakers were chosen randomly for the test set (roughly 2 hours). All the recordings have been saved as uncompressed 16-bit linear audio sampled at 16 kHz.

Acquiring the sufficient quantity of text data turned out to be slightly more complicated. The electronic text corpora are difficult to obtain for Polish because of poor attempts of digitizing written works and restrictive copyright laws. Furthermore, even though some text corpora exist, they have very limited access, forcing new researchers to recreate their own data sets each time.

A reasonably large initial set of transcripts was acquired with the help of the Office of the Senate amounting to about 5 million words. This corpus was then expanded with the transcripts of Parliamentary sessions, committee meetings, and the publicly available portion of the National Corpus of Polish (NKJP) (Przepiórkowski *et al.*, 2012). This data needed to be segmented and normalized to expand all the numbers and abbreviations to their spoken form according to the grammatical rules. Corpus normalization is usually done by hand but with the data of this size it was necessary to create automatic tools for word form agreement. This was solved using the authors' own decoder trained on manually assigned grammatical rules and manually expanded numbers. Tests showed around 10% error rate. The system first uses a list of word mappings and specialized algorithms to expand all the necessary tokens into all the possible expansions, including all the grammatical forms of the individual words. Next, a Viterbi-style decoder is used to find the best sequence of word forms using a combination of 3 language models (word, stem, and grammar class). More details about the system can be found in (Brocki *et al.*, 2012b).

An attempt to use a collection of legal documents graciously provided by a large publisher of such works was ultimately abandoned because the level of normalization needed in such texts was too great to achieve reasonable results. All experiments involving this data decreased performance making it unsuitable for our use. It is worth noting that such documents consist of a very large amount of abbreviations and the sentence structure does not correlate in any way with the spoken language (outside of the courtroom, perhaps). The only benefit would be to improve the vocabulary, but without a proper context such models would not work very well.

To summarize, most of the experiments were performed on a corpus containing roughly 145 million words. The corpus is tokenized, in the lower case, and all the numbers and abbreviations expanded to their full, spoken, grammatically correct forms.

3. Transcription system implementation

The transcription system is based on ASR, which is defined as a task of recognizing a sequence of words based on an audio recording of speech, see (Jelinek, 1997; Mori, 1998; Rabiner, 1989).

To properly implement the transcription system, several components needed to be developed:

1) phonetic transcription system, also known as a grapheme-to-phoneme (G2P) converter,
2) voice activity detector (VAD),
3) acoustic model (AM),
4) language model (LM),
5) decoder,
6) presentation and subtitling module.

In Fig. 1, a simple overview of the relations between the system components is presented. The audio signal, shown in the top left, is processed by a standard Feature Extraction module (FE) to produce frames of acoustic features. These frames are then filtered using the VAD module and fed into the AM. The information gathered from both the AM and LM is processed by the Decoder to generate a stream of words. The words are determined by a vocabulary (VOCAB) which is mapped to the phonetic output of the AM using a G2P module. Finally, the words are processed by a subtitling module (SUB) producing subtitles which can be presented on a video stream. Following is the description of the individual modules in more detail.

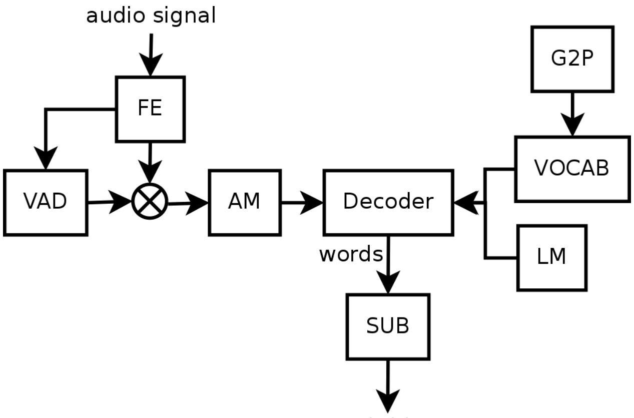

Fig. 1. General system overview containing all the components described in Sec. 3.

3.1. Grapheme-to-phoneme conversion

The G2P conversion is an important bridge between the sounds of speech and how we are used to reading and writing words on paper. This procedure is important both in the training phase (to convert the training transcripts into a description of recognizable speech sounds) and during a normal use (to convert the recognized sounds into readable text).

The phonetic alphabet chosen for this project is a variant of the Polish SAMPA (Wells, 2013), modified to contain only alphabetic characters (so symbols like the apostrophe and tilde were replaced with the letters i and n, respectively, see Fig. 2). The only actual difference is the lack of the "N" symbol which was found to carry little benefit when modelling speech purely for ASR purposes. It is, however, important to mention the multiple pronunciations per word arising from either Speaker dependent variability or coarticulation effects arising from context. Homographs are thus naturally solved at the LM level with the dictionary containing all the word variants. Homophones that are also heterographs are a slightly greater issue and, depending on the decoder, may be resolved in several ways (e.g., using disambiguation symbols). Homonyms are not of concern for pure LVCSR tasks.

senackiej	s e n a ts k j e j
senackim	s e n a ts k i m
senackimi	s e n a ts k i m i
senacką	s e n a ts k o m
senacką	s e n a ts k on
senat	s e n a t
senatach	s e n a t a x
senatem	s e n a t e m
senator	s e n a t o r
senatora	s e n a t o r a
senatorach	s e n a t o r a x
senatorami	s e n a t o r a m i

Fig. 2. An example of a phonetic dictionary as a result of the G2P conversion process.

The G2P system in the described project is a fast C++ stream processor based on a set of rules written as graphs. Each word or phrase is expanded to a graph form (FSA) based on these rules and then a dictionary is generated by traversing each node in the graph. Even though various systems for other languages already exist, for Polish such a task can be solved fairly efficiently by a finite set of rules, with the expressiveness of a regular language. The only problem using this approach are names, foreign and other atypical words. Such words were transcribed using a manually built dictionary which converts them to a form that can be transcribed using the rules mentioned before. The final system was tested and manually corrected using the lexicon generated from the training corpus, to minimize the potential errors that can occur for this particular task. Finally, the system consisted of a set of 972 basic rules and 4802 word replacement rules due to exceptions. With regards to the acutal performance of the system, it is generally assumed that the system performs accurately, with respect to phonetic rules, when given correctly normalized input in Polish. The only errors that occur are either due to errors in the preprocessing or because of foreign or foreign sounding words. As a simple test, a transcript from a fairly recent Senate meeting (July 23, 2014) was processed and after manually checking the resulting dictionary, only a dozen of the 12k words in the dictionary were mistranscribed. These were exclusively English words like "board of trustees" and "cashback" or English-derived spelling in words like "stricte".

3.2. Audio processing and voice activity detection

The audio pre-processing and parameterization were done similarly for all the tested solutions in this paper. A standard 39 feature vector (12 Mel-Frequency Cepstral Coefficients (MFCCs) + energy with delta and acceleration coefficients) was extracted from the signal windowed at 100 frames per second with frames being 25 ms long. Each of the programs implements its own feature extraction front-ends but they are all greatly influenced by HTK (Young *et al.*, 2002), and, therefore, the differences between them should be minimal.

Audio data is usually not homogeneous and contains mixed fragments of speech, music, background noise, and silence. As mentioned previously, distinguishing between these different types of audio is very important for the performance of the transcription system, especially when performing online recognition. Literature shows that acoustic modelling using Artificial Neural Networks usually outperforms equivalent Gaussian Mixture Models (GMMs) in accuracy (Robinson *et al.*, 1996). That is why the authors decided to use an LSTM RNN (Graves *et al.*, 2004) to perform the VAD audio segmentation. The same feature set mentioned above was used to train the RNN to recognize two classes: speech and non-speech. The output of the network was additionally smoothed using an excitation integrator in order to produce speech fragments of the minimum length of 20 frames (Brocki *et al.*, 2006). Such a system achieved 93.5% frame-level accuracy with the system tuned in, so that most errors occur in non-speech fragments.

Other available systems were also tested for the purpose of VAD. Shout (Huijbregts, 2008) is a speech recognition toolkit with a focus on speaker's diarization and adaptation. It contains a series of simple tools that perform various steps including: VAD, speaker diarization, VTLN (see Subsec. 3.3), model adaption, and, finally, recognition. The VAD program contains a simple, pre-trained GMM based acoustic model that detects two classes (speech and non-speech). This tool performed well for the Senate speeches analyzed in the project.

Several of the decoders that were tested in the project also contained a built-in VAD functionality. Julius (Lee *et al.*, 2001) contains several methods of VAD: using a simple audio analysis (level threshold and zero-crossing), using an AM which contains a model for silence and using an external GMM model. The first two methods were compared and both performed very similarly. For Julius, either of the methods is very much advised as it allows the decoder to segment the speech into coherent fragments, drastically improving performance with very long audios, as it is the case with Senate speeches. Kaldi (Povey *et al.*, 2011) performs silence detection only using the provided acoustic model and, given the extensive model adaptation tools, performs very well on such audio data. More details on the decoders are given in Subsec. 3.5.

3.3. Acoustic modelling

Depending on the system used, various AMs were created. For the GMMs used in both Julius and Kaldi, a simple 5-state (including start and end) topology was used to model triphones. For Julius, HTK tools were used to create tied-state triphones with both tree and clustering based state tying. The Kaldi project provides its own set of training tools with many state-of-the-art training and adaptation methods included in the basic program: cepstral-mean normalization (CMN), linear discriminant analysis (LDA) and maximum likelihood linear transformation (MLLT) feature transformation, vocal tract length normalization (VTLN), subspace gaussian mixture modelling (SGMM), feature space maximum likelihood linear regression (fMLLR) adaptation, training using maximum mutual information (MMI) criterion, and speaker adaptive training (SAT). Details on the performance of these methods are described in the experiment section.

CMN is a typical processing step for cepstral features which stands for subtracting the mean and removing the variance from the training data. The cepstral mean is usually regarded as the source of bias in the data arising from the differences in the transmission channel, whether it be intrinsic to the speaker or the environment where the recording took place. The subtraction of the cepstral mean is roughly equivalent to the de-convolution of the data with the transmission channel profile (Young *et al.*, 2002).

LDA is a common machine learning algorithm most often used to compress multi-dimensional feature space through a linear combination. This method is slightly more useful when faced with a feature space with many hundreds dimensions of various types and sources and can easily cause an increase in WER (word error rate). A slightly more robust approach uses MLLT (Psutka, 2007) which ties transformations to particular HMM (Hidden Markov Model) states and uses likelihood criteria to optimize the solution.

VTLN is a simple spectrum level processing technique that commonly uses a single parameter to modify the frequency envelope of the signal in order to compensate for the changes in pitch between speakers (which arise due to the difference of the length of the vocal tract) (Eide, Gish, 1996).

SGMM is a technique for acoustic modelling in which all phonetic states share a common GMM structure and their means and mixture weights vary in a subspace of the total parameter space (Povey *et al.*, 2010). This method creates more compact models which often perform better, especially with small amounts of training data.

Finally, fMLLR and MMI are common GMM adaptation methods. The former uses a set of linear transformation of the means and variances of the GMM models to reduce the difference between the model and the adaptation data set. Unlike the more traditional MAP (maximum a posteriori) approach, MLLR can adapt faster using less data. MMI is a slightly different adaptation approach that uses a discriminative approach to training GMM models. As such, it requires exact aligned training data to plug into its cost function. This alignment is usually not available ahead of time as making time aligned training data by hand is unfeasible at such quantities, but it is rather derived from running the decoder iteratively on data, slightly improving its performance along the way. This method can also utilize various objective functions, thus, the fMMI method will optimize both feature and model space errors, while the boosted MMI (also known simply as BMMI) is modified to boost the paths that contain more errors. The details on these methods can be found in (Veselỳ *et al.*, 2013).

Apart from the already available tools, an RNN based bi-directional LSTM (BLSTM) acoustic model was created by the authors. Such models are different from the aforementioned GMM models in that they contain variable length context, which allows them to efficiently model the dynamic nature of the data. The model was trained using a modified Back-Propagation Through Time (BPTT) routine which required the authors to reduce the training set to about 40 hours, due to the system complexity and time constraints (Graves, Schmidhuber, 2005).

Attempts were also made to use the new deep learning architectures, which was demonstrated in the combination of the Deep Belief Network (DBN) and BLSTM acoustic model. The initial results were very promising but the complexity of the system made it impossible to utilize on real-world data, for now. Similarly, some improvements were noticed using the recently developed DBN components in the KALDI toolkit (Veselỳ *et al.*, 2013). The tools used there implement the standard approach to DBN training as described in (Hinton *et al.*, 2006), by pre-training a set of Restricted Boltzman Machine (RBM) layers and then using standard error backpropagation on all the layers. The error can be calculated using several criteria: cross-entropy between the predicted and reference label distribution, MMI for discriminative training (just like the already mentioned above), and sMBR (state-level Minimum Bayes Risk) which is similar to MMI but works at a more granular level. An additional benefit to the toolkit is that it implements the costly training algorithms on the GPU (graphic card processor) using the Nvidia CUDA toolkit, which gave a 45-fold speedup compared to a single thread on the CPU. For typical LVCSR tasks, the training can still take weeks, even when several GPUs are used.

3.4. Language modelling

Much of the work in the project was spent on developing language models and evaluating their influence on speech recognition performance. Two kinds of models have been developed: traditional N-gram and RNN-based models. For N-grams, several aspects were tested: context length (N = 2..5), interpolation or back-off, smoothing methods (Kneser-Ney, Witten-Bell), dictionary size, and the quality was measured using perplexity on the test set. The best results were achieved with interpolated models with Kneser-Ney smoothing, which is consistent with results for other languages mentioned in the literature (Kneser, Ney, 1995).

One of the greatest challenges was the size of the models. For example, a 5-gram model trained on our corpus with a dictionary containing 308337 words, contains about 46 million word sequence probabilities, which is obviously impractical for real use. One solution is model pruning by minimal threshold entropy change for the chosen n-grams. Setting the threshold value to 10^{-8}, 10^{-9} eliminates about 50% of uncommon n-grams, with a relatively small increase in perplexity. Such a model can be used in practice, even though it is still quite large, consuming as much as 1 GB of memory.

Several programs for N-gram model creation were tested: SRILM (Stolcke *et al.*, 2002), IRSTLM (Federico *et al.*, 2008), and MITLM (Glass *et al.*, 2009). The most robust and best quality results were achieved using the SRILM package.

An ANN based connectionist language model was also tested during the project (Brocki *et al.*, 2012a). Such a model uses a dictionary that maps a vector of features C to each word, containing d real values. To calculate the probability of the next word in the sequence, a vector of features $x = (Cw_{t-n}, Cw_{t-n+1}, \ldots, Cw_{t-1})$ is fed into the ANN, which consists of a sequence Cw_t concatenated together, where w_t is a word at time t in the word sequence. Therefore, this vector describes a context of length n. The output layer contains as many neurons as there are words in the dictionary with an additional output for out-of-vocabulary (OOV) words. A softmax activation function is used in the output layer, which allows expressing the output as word occurrence probability, i.e., preserves the stochasticity property. The ANN is trained to minimize the log-likelihood using gradient descent. The gradient can be calculated using the well-known back-propagation algorithm, although modified to include not only the neural network synaptic weights but also the matrix containing the C vectors of word features. The process of training such a model is quite time consuming but in (Brocki, 2010b) the author suggests a parallelization scheme that allows using the model in practice (although the training still takes several weeks).

The connectionist model was trained in the following way: each word was assigned a 50-tuple real valued vector representing the input to the ANN. These values were initially set randomly in the range -0.01 to 0.01. The ANN was trained using a gradient descent algorithm with a momentum set to 0.99 and learning rate set to 10^{-7}. The final network outperformed the best n-gram model in perplexity by about 24% (Brocki, 2010a; Brocki *et al.*, 2014).

3.5. Decoders

During the project, the authors tried to adapt both their existing decoder (Koržinek, Brocki, 2007; Brocki *et al.*, 2008) and other open projects available online to the task of Polish Senate speech transcription. The authors' connectionist decoder was compared to Julius (Lee *et al.*, 2001) and Kaldi (Povey *et al.*, 2011) with respect to accuracy, speed, and system requirements.

The simplest and unsurprisingly most popular solution is the Julius decoder. It is a mature project with a very convenient open license which already got a lot of attention in Polish research circles. The system utilizes AMs trained using the well known HTK (Young *et al.*, 2002) toolkit and LMs available in the standard ARPA format. The decoder is a classic multi-pass Viterbi style decoder capable of outputting both lattices and confusion networks alongside the normal time-aligned output. It has remarkably low system requirements (less than 100 MB per channel) and is quite easy to set up and use on almost any platform (Windows, Linux, Mac) without needing extra external libraries.

At the time of writing this paper, Kaldi was a rather new but active project. It is a modern Weighted Finite-State Transducer (WFST) based toolkit including tools for everything from AM training to final output decoding in many ways (online, offline, time-aligned, lattices) with many tools available for model retraining, segmentation, and adaptation. It includes even some basic Artificial Neural Network (ANN) and Deep Neural Network (DNN) models for both AM and LM use. The only tools not available directly in the toolkit are those for creating standard N-gram LMs, but many are easily available and supported by the system (Perl scripts for using some of the tools are also available). Kaldi is much more advanced than Julius, and unsurprisingly, performs much better but at a much greater memory cost. Each channel can take well over 1 GB of memory and the setup is a bit more difficult, but it functions very well as an online decoder. It is very stable and works efficiently in real-time. It utilizes several external libraries, which makes it a challenge to set up on anything but Linux (at the time of writing this paper).

The authors' decoder is a connectionist system that uses a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) for the AM and a Viterbi decoder to output single or n-best output. The decoder uses a special LM comprising of 3 models: a word model using 3-grams, a lemma model using 5-grams, and a grammar class model using 7-grams. The three models are combined using linear interpolation optimized using an Evolutionary Strategy (Michalewicz, 1996) that minimizes perplexity. The feature front-end implements a dynamic normalization scheme and a VAD along the standard 39 feature extraction mentioned in the previous chapter. The system is very easy to set up but takes a lot of resources (at least 1 GB memory) and a long time (several weeks) to fully train. Nonetheless, it can perform quite accurately in real-time, even on an average home computer. Due to memory optimization techniques utilized, it is currently working only on Windows.

3.6. Subtitling and data selection

The presentation system chosen for the project was meant to aid the already existing system present on the new website of the Polish Senate. The website features video recordings and transcripts as separate documents. The authors' system can be thus used in two potential use-cases:

1) as a fully functional transcription tool – to display automatic (and somewhat inaccurate) transcripts, until the manual transcripts become available (usually 1–2 days after the actual meeting),

2) as a subtitle alignment tool – to assign subtitle timecodes to existing manual transcripts, once they become available.

The latter of the two tasks may seem as much more trivial but is actually quite challenging. The biggest problem is that the manual transcripts are not an accurate representation of what is actually spoken during the meetings. The transcripts are corrected in order to be easier to read in PDF form but omit many nuances of speech, like word repetitions, interruptions, and paralinguistic sounds. That is why, an alternative time-alignment technique is used: first the audio is recognized using a language model trained on the vocabulary that is known to exist in the audio and as a second step, text-to-text alignment is used to determine which portions of the audio have been correctly matched with the text. Next, this same method is used recursively on all the incorrectly aligned segments until a perfect alignment is reached (Katsamanis *et al.*, 2011). Given enough time, this method always converges to the forced time-alignment, although this may not always be our goal. If the transcripts that we are trying to align contain errors, we may decide to stop the forced alignment early and accept the ASR output instead. This may ultimately prove more accurate than fully forced time-alignement.

There are also many technical issues related to subtitle generation. This is usually handled by the specialized software designed specifically for use in television or movie subtitling. Such software contains many heuristics to determine how to optimally fit the information on the screen in order to make it easy to follow for the viewer. The system in this paper implemented only the basic heuristics: the number of characters per cue, minimum duration between two consecutive cues, and duration of cue after the speech is over. More advanced heuristics would be difficult to implement, because the output of the ASR is a stream of words that lacks any sentence boundaries, punctuation, or capitalization. This problem also affects Machine Translation and it is planned for the future to fix it with additional modules for punctuation and capitalization.

The final system also has the ability to export the result in WebVTT (Hickson, 2012) format. This is a W3C standard for describing subtitles used on websites. It is easy to deploy with existing streaming solutions, especially if HTML5 is used. This method was successfully tested by the authors on several modern browsers (e.g., Google Chrome).

4. Experiments

A comparison between the authors' system, Julius, and Kaldi was performed. All three systems described in the previous sections were trained on the same feature sets. This comparison is fairly imprecise as these systems are vastly different in their capabilities and training approaches. The values here are what can be regarded as the baseline for each system, given the same training data. Evaluation was perfomerd on an independent set of about 50 minutes in length with 10 different speakers.

Table 1. Comparison of different systems trained on the same data. The vocabulary was set to 90 k words and the LM was trained on a corpus of 60 million tokens.

System	WER
Julius/w 5-gram LM	34.62
Authors' hybrid system/w tripple LM	32.68
Kaldi/w 3-gram LM	30.06

The authors' connectionist system surpasses the traditional Hidden Markov Model (HMM) based one in both accuracy and resource consumption. Even though it works in real-time on a typical laptop computer, it consumes up to 2 GB of memory, whereas Julius performs at a fraction of that memory footprint. On the other hand, to achieve such results, Julius works only one third of real-time. Kaldi is capable of matching and exceeding the authors' system performance in real-time but has similar memory requirements as that system.

Many experiments were performed on the Kaldi system in order to establish the lowest error rate. These experiments included the following: different acoustic data training sets, different methods for training AMs, different language data training sets, context length of LMs, entropy pruning in LMs, dictionary size, Out-Of Vocabulary (OOV) words inclusion or exclusion from the training data, decoder parameter tuning. These experiments were tested on the set containing 10 different speakers (approx. 2 hours of data).

Table 2 shows a typical training procedure spanning over several AM optimization techniques available in Kaldi. Table 3 presents the evolution of the system given more data. The BASELINE system was trained on the initial 28 hours of Senate data and a vocabulary of 60 k words. In EXP1, more data from various online sources was added to improve the language model. By expanding the vocabulary to 211 k words in EXP2, the error rate was reduced quite significantly. Adding the

Table 2. Comparison of training methods in Kaldi on an initial training set containing only Senate recordings and transcripts.

Method	WER
Initial trigram	37.37
+LDA/MLLT	34.37
+MMI	32.01
+MPE	32.55
+SAT(fMLLR)	33.51
+MMI	31.81
+fMMI	29.85
+BMMI	29.69
+SGMM	32.39

Table 3. Several iterations of the system. The WER was reduced by adding more data as they became available.

Step	Triphone	+SAT	+fMMI
BASELINE	34.37	33.51	29.85
EXP1		30.89	25.69
EXP2	25.88		21.3
EXP3	23.96		19.96
EXP4	22.46		**19.56**

data from the NKJP corpus helped further reduce the WER in EXP3 and adding the full audio corpus, as shown in EXP4, brought the overall error rate to the final level of 19.5%.

The best results using Kaldi (WER = 19.5%) were achieved with AMs trained using fMMI criterium on a dictionary of 211 thousand words (giving ca. 350 thousand phonetic forms – the increase is mostly due to coarticulation rules that allow many words to be pronounced in multiple ways) and a 5-gram language model with the entropy pruning threshold set to 10^{-9}.

Preliminary experiments using DBNs were also performed. Initial experiments using the authors' connectionist decoder showed an improvement of 5% accuracy but due to the limitations of the system, the experiments were done on a much smaller corpus. Further experiments were therefore performed using the tools available in the Kaldi toolkit. A network was trained on the small, initial training set (as in Table 2) and the output of the fMLLR acoustic models was used as the input to the network. The WER was reduced to 29.51%, which is an improvement of 4% to the initial result. Note that this is the simplest training method available in Kaldi and further experiments need to be performed to fully utilize the toolkit (especially the MMI training methods).

5. Discussion and conclusion

Some observations can be made from these experiments:

- modern AM training methods can considerably improve the error rate,
- increasing the training data set size generally improves performance but not consistently; i.e., there are certain training methods that achieve higher gain with more data than others,
- a strong influence of the LM on the final performance was noticed during the experiments; it should be noted, however, that increasing the dictionary size does not always improve performance because new words may occur too sparsely in the training corpora to be modelled to a satisfactory level,
- very large dictionaries have to be reduced to be usable but pruning causes considerable increase in perplexity; it is distinctive, however, that the pruning of Polish data needs to be about two times smaller than English because of the rich inflection present in the language,
- a notable problem remaining is the lack of language data, specifically corpora of spoken language, available for Polish, instead of the literary or "smoothed" data that is very common; this makes the LMs poorly correlated with the actual spoken utterances, since they lack many of the common spontaneous speech artifacts – repetitions, hesitations, pauses, and mispronunciations; the relatively (compared to other languages) small domain text corpora reflect the poor recognition results.

This paper described the research behind creating the first Polish LVCSR transcription system of Senate speeches. The authors made effort to use the best tools available at the time to achieve this goal. Initial attempts were slightly discouraging, especially when relying on the HTK workflow and the Julius decoder. Fortunately, the transition to the Kaldi toolkit helped to achieve reasonable results. While the best result (WER 19.5%) may seem worse than what is reported for other languages, the reasons behind it are pretty clear: lack of training data, especially language data, introduces high LM perplexity and OOV rates. Combined with the fact that inflected languages have much larger vocabulary sizes makes Polish quite a challenge to overcome. The authors hope that this small step will encourage others to contribute and help bring the state of Polish ASR to the levels of the overall international research.

Acknowledgments

The work was sponsored by a research grant from the Polish Ministry of Science and Higher Education, no. N516 519439.

References

1. Brocki L. (2010a), *Koneksjonistyczny model języka polskiego*, [in:] XII International PhD Workshop OWD 2010.
2. Brocki L. (2010b), *Koneksjonistyczny Model Języka w Systemach Rozpoznawania Mowy*, PhD thesis, Polish-Japanese Institute of Information Technology.
3. Brocki L., Koržinek D., Marasek K. (2006), *Recognizing connected digit strings using neural networks*, [in:] Text, Speech and Dialogue, pp. 343–350, Springer.
4. Brocki L., Koržinek D., Marasek K. (2014), *Improved factorization of a connectionist language model for single-pass real-time speech recognition*, [in:] Foundations of Intelligent Systems, Andreasen T., Christiansen H., Cubero J.-C., Raś, Z., [Eds.], volume 8502 of Lecture Notes in Computer Science, pp. 355–364, Springer International Publishing.
5. Brocki Ł., Koržinek D., Marasek K. (2008), *Telephony based voice portal for a university.*
6. Brocki Ł., Marasek K., Koržinek D. (2012a), *Connectionist language model for polish*, [in:] Intelligent Tools for Building a Scientific Information Platform, pp. 243–250, Springer.
7. Brocki Ł., Marasek K., Koržinek D. (2012b), *Multiple model text normalization for the polish language*, [in:] Foundations of Intelligent Systems, pp. 143–148, Springer.

8. Demenko G., Grocholewski S., Klessa K., Ogórkiewicz J., Wagner A., Lange M., Sledzinski D., Cylwik N. (2008), *Jurisdic: Polish speech database for taking dictation of legal texts*, [in:] LREC.

9. Eide E., Gish H. (1996), *A parametric approach to vocal tract length normalization*, [in:] Acoustics, Speech, and Signal Processing, 1996, ICASSP-96, Conference Proceedings., 1996 IEEE International Conference on, volume 1, pp. 346–348, IEEE.

10. Federico M., Bertoldi N., Cettolo M. (2008), *Irstlm: an open source toolkit for handling large scale language models*, [in:] Interspeech, pp. 1618–1621.

11. Glass J.R., Hsu B.-J. *et al.* (2009), *Language modeling for limited-data domains.*

12. Graves A., Eck D., Beringer N., Schmidhuber J. (2004), *Biologically plausible speech recognition with lstm neural nets*, [in:] *Biologically Inspired Approaches to Advanced Information Technology*, pp. 127–136, Springer.

13. Graves A., Schmidhuber J. (2005), *Framewise phoneme classification with bidirectional lstm and other neural network architectures*, Neural Networks, **18**, 5, 602–610.

14. Hickson I. (2012), *Webvtt. living standard*, World Wide Web Consortium.

15. Hinton G.E., Osindero S., Teh Y.-W. (2006), *A fast learning algorithm for deep belief nets*, Neural Computation, **18**, 7, 1527–1554.

16. Huijbregts M.A.H. (2008), *Segmentation, diarization and speech transcription: surprise data unraveled.*

17. Jelinek F. (1997), *Statistical methods for speech recognition*, MIT press.

18. Katsamanis A., Black M., Georgiou P.G., Goldstein L., Narayanan S. (2011), *Sailalign: Robust long speech-text alignment*, [in:] Proc. of Workshop on New Tools and Methods for Very-Large Scale Phonetics Research.

19. Kneser R., Ney H. (1995), *Improved backing-off for m-gram language modeling*, [in:] Acoustics, Speech, and Signal Processing, 1995, ICASSP-95, 1995 International Conference on, vol. 1, pp. 181–184, IEEE.

20. Koržinek D., Brocki Ł. (2007), *Grammar based automatic speech recognition system for the polish language*, [in:] Recent Advances in Mechatronics, pp. 87–91, Springer.

21. Kos M., Vlaj D., Kacic Z. (1996), *Sloparl-slovenian parliamentary speech and text corpus for large vocabulary continuous speech recognition.*

22. Lee A., Kawahara T., Shikano K. (2001), *Julius – an open source real-time large vocabulary recognition engine.*

23. Lööf J., Bisani M., Gollan C., Heigold G., Hoffmeister B., Plahl C., Schlüter R., Ney H. (2006), *The 2006 RWTH parliamentary speeches transcription system*, [in:] INTERSPEECH.

24. Marasek K. (2012), *TED Polish-to-English translation system for the IWSLT 2012*, Proceedings IWSLT 2012.

25. Marasek K., Brocki Ł., Koržinek D., Szklanny K., Gubrynowicz R. (2009), *User-centered design for a voice portal*, [in:] Aspects of Natural Language Processing, pp. 273–293, Springer.

26. Michalewicz Z. (1996), *Genetic algorithms + data structures = evolution programs*, Springer.

27. Miłkowski M. (2012), *The Polish language in the digital age*, Springer.

28. Mori R.D. (1998), *Spoken Dialogue With Computers (Signal Processing and its Applications)*, Academic Press.

29. Povey D., Burget L., Agarwal M., Akyazi P., Feng K., Ghoshal A., Glembek O., Goel N.K., Karafiát M., Rastrow A. *et al.* (2010), *Subspace gaussian mixture models for speech recognition*, [in:] Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, pp. 4330–4333, IEEE.

30. Povey D., Ghoshal A., Boulianne G., Burget L., Glembek O., Goel N., Hannemann M., Motlicek P., Qian Y., Schwarz P. *et al.* (2011), *The Kaldi speech recognition toolkit*, [in:] IEEE 2011 workshop on automatic speech recognition and understanding.

31. Pražák A., Psutka J.V., Hoidekr J., Kanis J., Müller L., Psutka J. (2006), *Automatic online subtitling of the Czech Parliament meetings*, [in:] Text, Speech and Dialogue, pp. 501–508, Springer.

32. Przepiórkowski A., Bańko M., Górski R., Lewandowska-Tomaszczyk B. (2012), *Narodowy Korpus Języka Polskiego*, Wydawnictwo Naukowe PWN, Warszawa.

33. Psutka J.V. (2007), *Benefit of maximum likelihood linear transform (mllt) used at different levels of covariance matrices clustering in asr systems*, [in:] Text, Speech and Dialogue, pp. 431–438, Springer.

34. Rabiner L.R. (1989), *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, **77**, 2, 257–286.

35. Robinson T., Hochberg M., Renals S. (1996), *The use of recurrent neural networks in continuous speech recognition*, [in:] Automatic speech and speaker recognition, pp. 233–258, Springer.

36. Romero-Fresco P. (2011), *Subtitling through speech recognition: Respeaking*, St. Jerome Publishing.

37. Stolcke A. *et al.* (2002), *Srilm-an extensible language modeling toolkit*, [in:] INTERSPEECH.

38. Veselỳ K., Ghoshal A., Burget L., Povey D. (2013), *Sequence-discriminative training of deep neural networks.*

39. Wells J.C. (2013), *Polish sampa*,
http://www.phon.ucl.ac.uk/home/sampa/polish.htm.

40. Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D. *et al.* (2002), *The HTK book*, Cambridge University Engineering Department, 3.