LSTM for Punctuation Restoration in Speech Transcripts

Ottokar Tilk, Tanel Alumäe

Institute of Cybernetics
Tallinn University of Technology, Estonia
ottokar.tilk@phon.ioc.ee, tanel.alumae@phon.ioc.ee

Abstract

The output of automatic speech recognition systems is generally an unpunctuated stream of words which is hard to process for both humans and machines. We present a two-stage recurrent neural network based model using long short-term memory units to restore punctuation in speech transcripts. In the first stage, textual features are learned on a large text corpus. The second stage combines textual features with pause durations and adapts the model to speech domain. Our approach reduces the number of punctuation errors by up to 16.9% when compared to a decision tree that combines hidden-event language model posteriors with inter-word pause information, having largest improvements in period restoration.

Index Terms: neural network, punctuation restoration

1. Introduction

The output of most automatic speech recognition (ASR) systems consists of raw word sequences, without any punctuation symbols. While this is sufficient for some tasks, such as indexing and retrieval as well as dictation where punctuation symbols might be entered explicitly by voice, most speech transcription applications benefit from automatically inserted punctuation symbols. This serves two purposes. First, it makes the ASR-based transcripts easier to read and understand for humans. Second, in many cases, it also makes downstream machine processing of the generated texts more accurate, as many natural language processing tools, such as sentiment analyzers, syntactic parsers, information extraction and machine translation systems, are trained on written texts that include punctuation, and thus expect them to be present also in the input texts.

There have been many previous studies on automatic punctuation restoration in speech transcripts. Probably the most widely used approach is based on the so-called hidden event language model (LM) which uses a traditional N-gram LM trained on texts that include punctuation tokens [1]. During decoding, the LM is used to recover the most probable sequence of words and hidden punctuation symbols. Punctuation restoration can also be treated as a sequence labelling task and solved using conditional random fields (CRFs) [2]. This model allows to combine various textual features, such as LM scores, token n-grams, sentence length and syntactic features which is found to give large improvements over purely lexical features [3]. Many approaches combine textual information with acoustic/prosodic features, such as pause length between words, phoneme lengths, pitch and energy, using a classifier such as decision tree [4] or maximum entropy model [5]. Often the purely lexical model trained on large text corpora is combined with a model containing acoustic features, trained on a smaller speech corpus. This is usually done using either (log-)linear interpolation of model predictions [4] or using the posteriors of the lexical model as additional features to the overall model [6].

This paper describes a punctuation restoration system for automatically transcribed Estonian broadcast speech that uses long short-term memory (LSTM)[7]. LSTM, a type of recurrent neural network (RNN), has been used for a variety of supervised sequence labelling tasks, including phoneme classification [8] and spoken language understanding [9]. We are not aware of any prior work using RNNs for punctuation restoration.

Neural networks provide a flexible architecture for constructing and synthesizing complex models. We take advantage of this flexibility by training a punctuation restoration model in two phases. First, a large text corpus is used to train a model that uses only words as features. Then a new model is trained on a smaller pause annotated corpus, using pause durations and first phase models uppermost hidden layer outputs as features. The resulting punctuation system is currently used in our publicly available Estonian speech transcription system[1]. Its performance on automatically transcribed data can be viewed on our constantly updated archive of Estonian broadcast speech [2][10]. The source code of the model is also publicly available [3].

The following section describes how we use LSTM for punctuation restoration. Section 3 presents evaluation data, evaluation metrics and experimental results. Section 4 concludes the paper.

2. Method

We focus on restoring two most important and frequent types of punctuation — commas and periods. Question marks, exclamation marks, semicolons and colons are mapped to periods and all other punctuation symbols are removed from the corpora. The model we use for this task is a LSTM RNN with forget gates [11] and peephole connections [12] in LSTM layers.

2.1. LSTM vs N-gram

The LSTM RNN has several advantages over the widespread hidden event N-gram LM.

First, one of the problems with N-gram models is the data sparsity issue. A better model should to be able to generalize to contexts that were not seen during training. As it is well known from language modeling, neural networks are much better at generalizing to unseen sequences, by learning distributed representations of words [13] or entire contexts as it is the case with RNNs [14]. This suggests that LSTM RNNs should be able to learn similar representations for contexts around similar punctuations and make accurate predictions even in unseen contexts.

[1] http://bark.phon.ioc.ee/webtrans/
[2] http://bark.phon.ioc.ee/tsab
[3] https://github.com/ottokart/punctuator

Another weakness of N-gram models is that their context size is limited to a fixed number of tokens. Although it has been shown that increasing the context size does not help as much as getting more data [15], it seems unjustified to expect that the relevant context size is the same for both types of punctuation and remains constant across the entire text. Therefore, one of our requirements is that the model should be able to dynamically decide on how long context is relevant. RNNs fit this requirement and are able to utilize arbitrary length sequences. Although in practice non-LSTM RNNs have difficulties in remembering long range dependencies due to vanishing gradients [16], this problem can be alleviated by using LSTM units.

Neural networks are very simple to augment with additional features, such as those derived from speech prosody, which gives them another advantage over N-gram models.

The main disadvantage of neural networks is their training speed, although it is not as huge problem as in language modeling as the output layer is small.

2.2. Input features

Our approach to punctuation restoration relies on both textual information and pause durations between words. Contrary to many previous works, we don't use any other prosodic features, such as F_0 contours, phoneme durations and energy values. This has two reasons: first, previous research [17, 4] has shown that pause duration is by far the most useful and robust prosodic feature for predicting punctuation symbols; second, it is easy to extract pause durations from word-aligned recognition output that can be generated using any decoder, without the need to reanalyze the audio signal. We also opted against using other linguistic features, such as part-of-speech tags, because we wanted our approach to be as portable to other languages as possible.

The textual component of our model decides the suitable punctuation for a slot based on the word after the slot and the history of all previously seen words. The word after the slot is given as one-hot encoded input vector and the information about preceding words is stored in the LSTM memory units. The next word is important for predicting both commas and periods correctly. In Estonian language there are many words that are almost always preceded by a comma and thus it is essential to know the following word. Also, in item listings it is indicative whether the following word is from the same category. For periods it helps to detect thematic changes and to recognize words which typically start a new sentence. Since there are many easy cases for commas, we can expect the textual model to be better at predicting commas than periods.

Pauses between words are most informative for predicting periods. This became apparent when we trained a simple model using a small window of pauses only as input features. The model achieved an F-score of 0.53 for periods, but was unable to predict commas. Although, in certain contexts, a small pause can be good indicator for a comma. Therefore, in conjunction with word features, the comma annotation performance should also improve, but we expect periods to benefit the most.

Combining textual and pause duration information should provide a model with balanced performance across different punctuations.

2.3. Two-stage model

Since the amount of pause annotated data is relatively small, we use a two-stage training procedure in which a purely textual model (T-LSTM) is trained first. The forward pass of the T-LSTM model is described in the following equations:

$$\begin{aligned} y_0(t) &= \tanh\big(W_0 x_0(t)\big) \\ y_1(t) &= LSTM\big(y_0(t)\big) \\ y_2(t) &= Softmax\big(W_2 y_1(t)\big) \end{aligned}$$

where x_0 is the one-hot encoded vector representing the input word following the punctuation slot, y and W are the layer activation vectors and weight matrices respectively where the subscript matches the layer index. The $LSTM$ is defined as in [18], except biases are omitted because we found they brought no noticeable improvement.

After obtaining the textual model, the final model utilizing text and pause duration (TA-LSTM-p) is trained in the second stage. As proposed in [19], we treat the last hidden layer outputs of the T-LSTM as high level features learned by the network, representing everything the model knows about the textual input. TA-LSTM-p then utilizes both pause durations and these high level features of text to make decisions about punctuations. To construct a TA-LSTM-p the output layer softmax classifier of T-LSTM is discarded and its last hidden layer features with the current slot pause duration are used as inputs to the TA-LSTM-p. The architecture of the TA-LSTM-p is identical to the T-LSTM with the exception of inputs and the omitted first hidden layer. During the second phase of training the model learns to use the now available pause duration information in conjunction with textual features. It also enables the model to adapt to the style of speech by learning a more suitable classifier for the target domain. The remaining T-LSTM layers are still used in the forward pass of the TA-LSTM-p, but are fixed during training. Stacking a new classifier on top of the T-LSTM features has two advantages when compared to e.g. adapting the existing T-LSTM parameters. First, the size of the model trained on the smaller pause annotated corpus can be easily adjusted to be optimal for the smaller corpus size — which as a side-effect can be also faster to train. Second, according to our experiments, stacking a new classifier performs better than adapting the existing parameters. While in the T-LSTM, the function of the LSTM layer is to remember the textual context, the LSTM layer in TA-LSTM-p serves to have information about previous pauses (e.g. whether the current pause is long enough to indicate punctuation or just characteristic to current context) and previous T-LSTM features. The forward pass of the TA-LSTM-p model takes the following form:

$$\begin{aligned} x_1(t) &= [y_1(t), p(t)] \\ y_3(t) &= LSTM\big(x_1(t)\big) \\ y_4(t) &= Softmax\big(W_4 y_3(t)\big) \end{aligned}$$

where x_1 is the input to the TA-LSTM-p model, which is obtained by concatenating the second hidden layer activations y_1 of T-LSTM and pause duration p of current slot. The output layer y_4 represents the probability distribution over possible punctuations — comma, period or no punctuation. During punctuation restoration the slot t is filled with the most probable punctuation according to y_4. Both T-LSTM and TA-LSTM-p are described in Figure 1.

3. Experiments

The LSTM model is trained similarly in both stages. Gradients are computed with back-propagation through time [20] over 5 time steps and the weights are updated using AdaGrad [21].

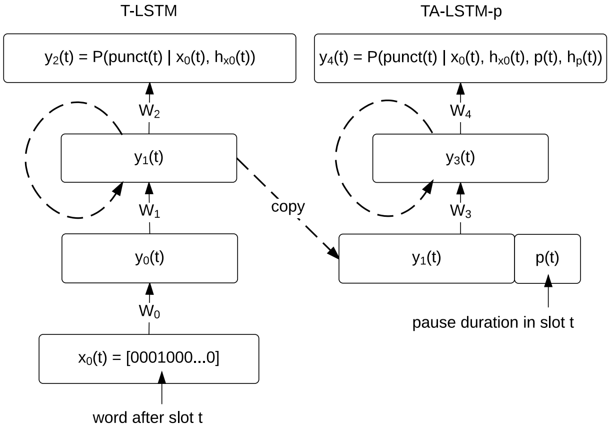

Figure 1: Description of the T-LSTM and TA-LSTM-p model. W_1 and W_3 represent the external input weights (for cell inputs and all gates) of the corresponding LSTM layers. The output of the T-LSTM model is a posterior probability distribution over punctuations $punct(t) = \{comma, period, \varnothing\}$ for slot t given word after the slot $x_0(t)$ and preceding word history $h_{x0}(t)$. TA-LSTM-p has a similar output, but is additionally conditioned on slot pause duration $p(t)$ and pause history $h_p(t)$.

Learning rate starts from 0.1 and when there is no sufficient improvement on the validation set we start to divide the learning rate by 2 at each epoch (adopted from [22]). Training is stopped when the shrinking learning rate no longer yields enough improvements or the maximum number of 20 epochs has been reached. Weights are initialized to random uniform values in a range of ± 0.005 and all hidden states to zeros. To speed up the training, the dataset is split into 100 sequences and these sequences are fed to the network in parallel as mini-batches.

The first and second hidden layer of the T-LSTM model consist of 100 tanh units and 100 single-cell LSTM blocks respectively. Input vocabulary consists of the 100K most frequent words in the training corpus plus two special symbols for unknown words and an end of input.

The TA-LSTM-p input size is 101 (T-LSTM features plus current slot pause duration). Hidden layer has 100 LSTM units.

We use two baselines: a hidden event LM and a decision tree that combines textual and inter-word pause information. The 4-gram hidden event LM uses a vocabulary of the same 100K words as the LSTM models plus punctuation symbols. The model is built by interpolating the models compiled from the individual text sources using coefficients optimized on the development set. The model is smoothed using Kneser-Ney discounting and n-gram probabilities accounting for less than 10^{-7} training set perplexity improvement are pruned. The second baseline is inspired from the models proposed in [4, 6]: a decision tree (4-gram+DT-p) is trained on the pause annotated corpus, using pause durations and the posterior probabilities of the punctuation symbols assigned by the hidden n-gram LM as features. This allows the decision tree to directly benefit from both inter-word pause information as well as a large text corpus which has no corresponding speech data available, similarly to the TA-LSTM-p model.

For a comparison with the 4-gram model, we also train a TA-LSTM-p model without pause duration inputs (i.e. just adapt it on the target domain data). This also helps us to assess the effectiveness of the TA-LSTM-p in utilizing pause duration information and compare it to the baselines DT-p approach. We refer to this adapted, purely textual model as TA-LSTM. We could have used the T-LSTM model for comparison, but we had two reasons to not do that — first, the N-gram model is adapted for target domain, thus for fairness the LSTM model should also be adapted; second, the T-LSTM model did not perform well on the speech data as the training data consisted mostly of written text (this might imply that local contexts don't vary between styles as much as longer ones).

Table 1: Number of tokens of textual data and the amount of hours of audio data used for training the punctuation model.

Source	*#Tokens*	*#Hours*
Newspapers	203M	
Web	74M	
Fiction	35M	
Magazines	29M	
Parliament	15M	
Social media	28M	
Lecture speech	283K	39.3
Broadcast news	127K	32.1
Broadcast conversations	505K	74.0
Total	*386M*	*145.4*

To demonstrate the importance of textual features trained on a large corpus, we train an augmented T-LSTM model with additional pause duration input on pause annotated text only. This model is referred to as T-LSTM-p.

All models are evaluated on force-aligned manual transcriptions and the transcripts generated by the ASR system. In order to insert reference punctuation marks into automatic transcripts, the manual transcripts were aligned with the ASR output, using minimum cost edit distance, and the punctuation marks in the reference texts were propagated to the hypothesized texts.

3.1. Data

Textual and audio data used for training the punctuation model is summarized in Table 1. The audio data is force-aligned using the speech recognition system described below. Inter-word pause durations, used for training the 4-gram+DT-p, TA-LSTM-p and T-LSTM-p model, are then captured from the alignments.

As development data, we use two hours of broadcast news and 4.4 hours of broadcast conversations (radio talkshows and telephone interviews). The test set contains two hours of broadcast news and 5.6 hours of broadcast conversations.

3.2. Speech recognition system

The ASR system that is used for aligning punctuation model training data and producing the ASR hypotheses for the evaluation data is described in [23], although some details have been improved since then. Speech recognition is implemented using the Kaldi toolkit [24]. The acoustic models are trained from around 178 hours of speech from various sources. We use multisplice DNN-based acoustic models that take an i-vector of the current speaker as additional input to the DNNs for unsupervised speaker adaptation.

The ASR LM is compiled from the same data that is used for training the text-based punctuation model (Table 1). A 4-gram LM is built by interpolating models trained on the individual sources, using a vocabulary of 200K compound-split words. The final LM is heavily pruned to be usable for decoding. One-

Table 2: Results on manually transcribed reference and ASR output test set.

	Reference text							*ASR output*						
Model	*P(C)*	*R(C)*	*F(C)*	*P(P)*	*R(P)*	*F(P)*	*Err*	*P(C)*	*R(C)*	*F(C)*	*P(P)*	*R(P)*	*F(P)*	*Err*
4-gram	0.78	0.60	0.68	0.47	0.26	0.33	9.67	0.70	0.54	0.61	0.38	0.19	0.25	11.32
4-gram+DT-p	0.76	0.70	0.73	0.64	0.60	0.62	8.22	0.66	0.62	0.64	0.52	0.48	0.50	10.97
T-LSTM-p	0.79	0.63	0.70	**0.69**	0.60	0.64	8.24	0.70	0.57	0.63	**0.57**	0.49	0.53	10.46
TA-LSTM	0.74	**0.72**	0.73	0.63	0.43	0.51	8.20	0.65	**0.65**	0.65	0.49	0.32	0.39	11.02
TA-LSTM-p	**0.82**	0.70	**0.76**	0.68	**0.77**	**0.72**	**6.83**	**0.71**	0.62	**0.66**	0.55	**0.61**	**0.58**	**10.01**

pass recognition is used, followed by rescoring of the lattices with a larger LM.

The word error rate (WER) of the system is around 17%.

3.3. Metrics

All models are evaluated in terms of four different metrics. First we give an overall classification error rate Err defined as incorrectly punctuated slots divided by the total number of slots. For a more detailed overview we also report precision, recall and F-score of commas and periods.

3.4. Results on reference text

Left side of Table 2 shows the results on manually transcribed reference text. The 4-gram model has decent performance in restoring commas but fails miserably when it comes to periods where especially the poor recall of 0.26 stands out. This might indicate that periods depend on longer context than the 4-gram model is able to utilize and require better generalization.

The LSTM model trained on purely textual features (TA-LSTM) achieves a 15.2% relative decrease in error rate over 4-gram model on reference text. While there are improvements in comma restoration, most of the gains come from better period restoration performance. This confirms that longer context and better generalization of LSTMs is indeed helpful when it comes to punctuation restoration, particularly for periods. TA-LSTM performance may be also partially attributed to the superiority of the adaptation scheme used (the 4-gram model uses linear interpolation). Just as with the 4-gram model, the TA-LSTM model is still worse at period restoration than comma restoration, although the difference is noticeably smaller.

Adding pause duration features yields noticeable reductions in error rate — TA-LSTM-p reduces error by 16.7% relative over TA-LSTM and 4-gram+DT-p improves by 15.0% relative over 4-gram model. As expected, the biggest improvement in TA-LSTM-p performance comes from the large jump in period restoration recall, followed by a noticeable rise in comma restoration precision. The TA-LSTM-p model has a relatively balanced performance over punctuation marks where both commas and periods have similar F-scores while the precision is larger for commas but recall is higher for periods. 4-gram+DT-p compared to 4-gram also mainly improves in period restoration, but shows a smaller improvement in comma restoration.

When comparing T-LSTM-p, a model trained on the small pause annotated corpus only, to the TA-LSTM-p model, it becomes clear that using textual features trained on a large corpus helps a lot. Biggest improvement is in period recall, again hinting that good representations of long contexts are important for period restoration.

The TA-LSTM-p model beats the 4-gram+DT-p baseline by 16.9% relative. It has much higher period recall and higher precision for both punctuation marks.

3.5. Results on ASR output

Experiments on ASR output (Table 2, right) show similar trends as the reference text, although the differences between models are much smaller. TA-LSTM reduces error rate by 2.7% relative compared to 4-gram. 4-gram+DT-p improves over 4-gram by 3.1% and TA-LSTM-p over TA-LSTM by 9.2%. Comparing TA-LSTM-p to the 4-gram+DT-p baseline shows relative error rate reduction of 8.8%.

It is clear that all models suffer from the errors made by the ASR system. It is also evident that LSTM models suffer more and period restoration performance declines more than for commas. This might be another indicator that restoring periods requires larger context (which LSTM models are able to use) and as context size grows the more likely it is to encounter errors made by the ASR system. As LSTM models suffer more from ASR WER, they also have a higher potential for improvements when the WER of the ASR systems improves.

Another thing to note is that alignment method used to propagate reference punctuations to ASR output is not error free either. Manual inspection of punctuations restored by TA-LSTM-p model revealed that many decisions made by the model that were counted as mistakes were actually better than the expected punctuations. While flawed, our current ASR output test set should still give a rough performance estimate.

4. Conclusions

This paper presented a novel LSTM RNN model for restoring periods and commas in speech transcripts. The model was trained in two stages where in the first stage textual features were learned and in the second stage pause durations and textual features were combined and the model was adapted to the target domain. Based on experiments with Estonian broadcast speech, the model showed a balanced performance for both punctuation types and reduced the number of restoration errors by 16.9% on reference text and by 8.8% on ASR output when compared to a decision tree that combines the output of a hidden event LM with pause durations. Most of the gains came from largely improved period restoration.

Future research includes other languages, restoring additional types of punctuation and using larger amount of prosodic features. Also, current LSTM model only peeked one word after the punctuation slot. It would be interesting to find out whether looking further into the future context improves the performance. Bidirectional models [25] can also be considered.

5. Acknowledgements

The work was supported by the European Union through the European Regional Development Fund, project 3.2.1201.13-0010 and by the Estonian Ministry of Education and Research target-financed research theme No. 0140007s12.

6. References

[1] A. Stolcke, E. Shriberg, R. A. Bates, M. Ostendorf, D. Hakkani, M. Plauch, G. Tr, and Y. Lu, "Automatic detection of sentence boundaries and disfluencies based on recognized words," in *ICSLP 1998*, Sydney, Australia, 1998.

[2] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *EMNLP 2010*, Cambridge, MA, USA, 2010.

[3] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text," in *Interspeech 2013*, Lyon, France, 2013.

[4] J. Kolář, J. Švec, and J. Psutka, "Automatic punctuation annotation in Czech broadcast news speech," in *SPECOM 2004*, Saint Petersburg, Russia, 2004.

[5] J. Huang and G. Zweig, "Maximum entropy model for punctuation annotation from speech," in *ICSLP 2002*, Denver, CO, USA, 2002.

[6] J. Kolár and L. Lamel, "Development and evaluation of automatic punctuation for French and English speech-to-text," in *Interspeech 2012*, Portland, OR, USA, 2012.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[8] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP 2013*, Vancouver, BC, Canada, 2013, pp. 6645–6649.

[9] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *SLT 2014*, South Lake Tahoe, NV, USA, 2014.

[10] T. Alumäe and A. Kitsik, "TSAB – web interface for transcribed speech collections," in *Interspeech 2011*, Florence, Italy, 2011, pp. 3335–3336.

[11] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[12] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[13] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[14] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *ICASSP 2011*, 2011, pp. 5528–5531.

[15] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *ICASSP 2009*, 2009, pp. 4741–4744.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] H. Christensen, Y. Gotoh, and S. Renals, "Punctuation annotation using statistical prosody models," in *ISCA Tutorial and Research Workshop (ITRW) on Prosody in Speech Recognition and Understanding*, 2001.

[18] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech 2014*, 2014.

[19] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *ASRU 2011*, 2011, pp. 24–29.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 9, 1986.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[22] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "RNNLM – recurrent neural network language modeling toolkit," *ASRU 2011*, pp. 196–201, 2011.

[23] T. Alumäe, "Recent improvements in Estonian LVCSR," in *SLTU 2014*, Saint Petersburg, Russia, 2014.

[24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. e. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *ASRU 2011*, Dec. 2011.

[25] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.