INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration

Ottokar Tilk[1], *Tanel Alumäe*[2]

[1]Institute of Cybernetics, Tallinn University of Technology, Estonia
[2]Raytheon BBN Technologies, Cambridge, MA, USA
ottokar.tilk@phon.ioc.ee, tanel.alumae@phon.ioc.ee

Abstract

Automatic speech recognition systems generally produce unpunctuated text which is difficult to read for humans and degrades the performance of many downstream machine processing tasks. This paper introduces a bidirectional recurrent neural network model with attention mechanism for punctuation restoration in unsegmented text. The model can utilize long contexts in both directions and direct attention where necessary enabling it to outperform previous state-of-the-art on English (IWSLT2011) and Estonian datasets by a large margin.

Index Terms: neural network, punctuation restoration

1. Introduction

Most automatic speech recognition (ASR) systems output an unpunctuated sequence of words. Restoring the punctuation greatly improves the readability of transcripts and increases the effectiveness of subsequent processing, like machine translation, summarization, question answering, sentiment analysis, syntactic parsing and information extraction.

Punctuation restoration and a related task of segmentation or sentence boundary detection have been extensively studied.

Some previous approaches have used textual features only, enabling applications where audio is not available. Various methods have been used, like n-gram models [1], conditional random fields (CRFs) [2, 3], transition-based dependency parsing [4], deep and convolutional neural networks [5]. Some have treated the punctuation restoration as a machine translation task, translating from unpunctuated text to punctuated text [6, 7].

On the other hand, there are methods that rely entirely on prosodic or audio based features, such as pause durations between words, pitch and intensity [8, 9]. For example, a combination of two neural networks has been used, where the first network classifies input as speech or punctuation and the second one predicts the punctuation type [9].

Both approaches have benefits — text based approach does not require audio and has generally shown better results on reference transcripts, while prosody based models are more robust to ASR system errors — but the combination of the two brings further improvements [10, 11]. Pause durations between words have been shown to be particularly helpful when combined with textual features [8, 12]. Approaches for combining textual and prosodic features can be roughly divided into two categories — a single model that utilizes both types of features, and separate models that are combined in various ways.

Single model approach has been used, for example, with maximum entropy model [13, 14, 15, 16], statistical finite state model [8], boosting-based classifier [10] and long short-term memory (LSTM) recurrent neural network [17].

A common way to combine models is to pass the outputs of the textual model along with prosodic features to the main model that makes the final punctuation decision. For example, language model posteriors can be treated as features by a decision tree [18], CRFs [19] or adaptive boosting algorithm [11]. Another option is to use prosodic posteriors as features for a model that combines them with textual features, like in [20] where deep neural network based prosodic model posteriors were used as additional features in a text based CRFs classifier. Prosodic and textual model posteriors can also be interpolated [18, 12, 21, 10] or passed to a third model as features [22].

Combination of a separate textual and prosodic component makes it straightforward to achieve a greater quality textual model, as it is not limited to the availability of corresponding audio and can be separately trained on a much larger amount of text [22]. Single model methods can achieve the same goal through adaptation or 2-stage training, where the model is initially trained on a large text corpus using textual features alone, and then adapted on a smaller corpus where both textual and prosodic features are available [17].

In [23], a multi-pass approach additionally refined a prosody and text based CRFs result, by taking into account the distance from the closest sentence boundary in both directions.

In this work we use two approaches. On the English dataset we use text only, as prosodic features were unavailable to us and the previous best result that we compare with. On the Estonian dataset we use textual features in combination with a prosodic feature. Similarly to the previous best method [17], the only prosodic feature we use is the pause duration between words, but other features can also be easily incorporated into this model. We use a single model that is trained in two stages to maximally utilize both text and prosody. The two-stage approach is similar to the one used in [17] where in the first stage a large written text corpus is utilized for training textual features, and then these features are combined with the pause duration feature in the second stage when the model is trained on a smaller pause annotated corpus. Although some of the previous work (e.g. [6]) reported results on already segmented text, our results are achieved on unsegmented text.

The novelty of our approach is that this is, to the best of our knowledge, the first use of bidirectional recurrent neural networks (BRNN) [24] in combination with an attention mechanism [25] for punctuation restoration in unsegmented text. The source code of the model is publicly available [1].

The next section describes our approach in detail. Section 3 describes training strategies, models, data, metrics and results. Section 4 concludes the paper.

[1]https://github.com/ottokart/punctuator2

2. Method

Our model is a bidirectional recurrent neural network (BRNN) [24] which enables it to make use of unfixed length contexts before and after the current position in text.

In the recurrent layers we use gated recurrent units (GRU) [26] that are well suited for capturing long range dependencies on multiple time scales. These units have similar benefits as LSTM [27] units while being simpler.

We incorporated an attention mechanism [25] into our model to further increase its capacity of finding relevant parts of the context for punctuation decisions. For example the model might focus on words that indicate a question, but may be relatively far from the current word, to nudge the model towards ending the sentence with a question mark instead of a period.

To fuse together the model state at current input word and the output from the attention mechanism we use a late fusion approach [28] adapted from LSTM to GRU. This allows the attention model output to directly interact with the recurrent layer state while not interfering with its memory.

Next we describe in detail how our model processes the inputs to produce the outputs. At time step t the model outputs probabilities for punctuations $\boldsymbol{y}_t$ to be placed between the previous word $\boldsymbol{x}_{t-1}$ and current input word $\boldsymbol{x}_t$. As there is no punctuation before the first word $\boldsymbol{x}_1$, the model predicts punctuations only for words $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$, where $\boldsymbol{x}_T$ is a special *end-of-sequence* token.

The sequence of one-hot encoded input words $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ is first processed by a bidirectional layer consisting of two recurrent layers with GRU units, where one recurrent layer processes the sequence in forward direction and the other in reverse direction. Both recurrent layers are preceded by a shared embedding layer with weights $\boldsymbol{W}_e$. The state $\overrightarrow{\boldsymbol{h}}_t$ at time step t of the forward recurrent layer is

$$\overrightarrow{\boldsymbol{h}}_t = GRU(\boldsymbol{x}_t \boldsymbol{W}_e, \overrightarrow{\boldsymbol{h}}_{t-1}) \quad (1)$$

where GRU is the gated recurrent unit activation function as described in [26] with the exception of added biases. We use $tanh$ as the new hidden state nonlinearity ϕ. The state $\overleftarrow{\boldsymbol{h}}_t$ of the reverse recurrent layer is computed similarly except the input word sequence $\boldsymbol{X}$ is processed in reverse order. The bidirectional state $\boldsymbol{h}_t$ is then constructed by concatenating the states of the forward and backward layers at time t:

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t] \quad (2)$$

So this layer learns representations for each input word x_t that depend on both the preceding and following context, hopefully helping the model to better identify question indicating words as this often depends on the context (e. g. "This is *what* I do." vs. "*What* do you do?"). Also, this gives the model more information to determine whether the current word starts a new sentence or not.

The bidirectional layer is followed by a unidirectional GRU layer with an attention mechanism. This layer processes the bidirectional states sequentially and keeps track of the current position in text, while the attention mechanism can focus on relevant bidirectional context aware word representations before and after the current position. The state $\boldsymbol{s}_t$ of the layer

$$\boldsymbol{s}_t = GRU(\boldsymbol{h}_t, \boldsymbol{s}_{t-1}) \quad (3)$$

is late fused with the attention model output $\boldsymbol{a}_t$ which is computed based on the previous state $\boldsymbol{s}_{t-1}$ and bidirectional layer states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$ as described in [25]. The late fused state $\boldsymbol{f}_t$

$$\boldsymbol{f}_t = \boldsymbol{a}_t \boldsymbol{W}_{fa} \circ \sigma(\boldsymbol{a}_t \boldsymbol{W}_{fa} \boldsymbol{W}_{ff} + \boldsymbol{h}_t \boldsymbol{W}_{fh} + \boldsymbol{b}_f) + \boldsymbol{h}_t \quad (4)$$

is fed to the output layer producing the punctuation probabilities $\boldsymbol{y}_t$ at time step t

$$\boldsymbol{y}_t = Softmax(\boldsymbol{f}_t \boldsymbol{W}_y + \boldsymbol{b}_y) \quad (5)$$

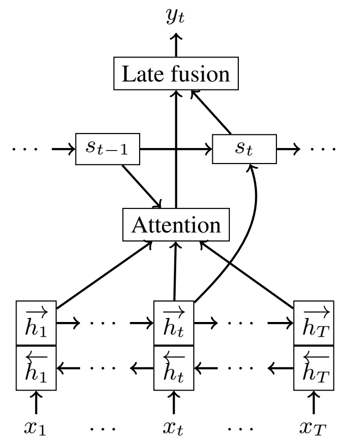

Figure 1: *Description of the model predicting punctuation $\boldsymbol{y}_t$ at time step t for the slot before the current input word $\boldsymbol{x}_t$.*

The model described above is used for single stage training and as the first stage in two-stage training. Graphical description of the model can be seen in Figure 1.

For two-stage training, to incorporate pause duration and adapt to target domain, the second stage discards the first stage output layer and replaces it with a new recurrent GRU layer

$$\boldsymbol{z}_t = GRU([\boldsymbol{f}_t, p_t], \boldsymbol{z}_{t-1}) \quad (6)$$

which takes the concatenation of the late fusion state $\boldsymbol{f}_t$ and the pause duration p_t before word $\boldsymbol{x}_t$ as input. Vector $\boldsymbol{z}_t$ is passed to a newly initialized output layer, similar to 5. Only the newly added parameters are trained during second stage training while the first stage parameters are kept fixed. This worked better than adapting all the parameters. The reason for that might be that as the second stage training corpus is smaller, it does not contain all the words that are in the model vocabulary. Therefore some word embeddings are not updated while the rest of the model changes, causing these embeddings to become less compatible with the model.

3. Experiments

3.1. Training details

During training the weights are updated using AdaGrad [29] with a learning rate of 0.02. The L_2-norm of the gradient is kept within a threshold of 2 by normalizing it every time this threshold is exceeded [30].

Negative log-likelihood of the punctuation sequence is minimized during training. During testing the punctuation with highest probability according to model output is chosen. We also experimented with giving the previously predicted punctuations as an input to the model and using beam search to find the best sequence of predictions, but this caused the model to accumulate mistakes and performed worse.

The first stage of two-stage training is finished when the validation perplexity gets worse for the first time. The second stage of two-stage training and single stage training is completed when the validation perplexity has not improved in the last 5 epochs. Weights are initialized according to the normalized initialization from [31] and biases are initialized to zeros. All hidden layers consist of 256 units.

The models are implemented using Theano [32, 33] and trained on GPUs. The input sequence is partitioned into 200 word long slices. Each slice always begins with the first word of a sentence. If a slice ends with an unfinished sentence, then the unfinished sentence is copied to begin the next slice. The output sequence is one element shorter as no punctuation is placed before the first word. Slices are also used during testing, but unlike during training, the sentence boundaries predicted by the model are used. To reduce training time, the slices are shuffled before each epoch and arranged into mini-batches of 128 slices.

The Estonian model has an input word vocabulary of 100K most frequent words in the training corpus, plus the *end-of-sequence* and *out-of-vocabulary* token. The vocabulary of the English model is constructed by taking all words that occur at least twice in the training corpus, resulting in a vocabulary of 27 244 words and the 2 special tokens.

The output vocabulary consists of the predicted punctuations (comma, period and question mark) and a *no punctuation* token. Other punctuation symbols are either mapped to one of the punctuations in our output vocabulary or removed from corpora. For Estonian dataset, exclamation marks, semicolons and colons are mapped to periods and all other punctuation symbols are removed. In the English dataset exclamation marks and semicolons are mapped to periods, while colons and dashes are mapped to commas.

3.2. Models

On the English dataset we use the model described in Figure 1 (T-BRNN). Since the models in [5] used pre-trained word vectors, we also train one T-BRNN model with embeddings initialized to the same pre-trained word vectors [2] (T-BRNN-pre) for comparison.

The Estonian dataset has out-of-domain and pause annotated data available. Therefore we train our model using the two-stage approach — first training on the large out-of-domain corpus and then adapting on the pause annotated corpus. We train the two-stage Estonian model both with (TA-BRNN-p) and without (TA-BRNN) utilizing pause durations. Analogous models (TA-LSTM-p with pauses and TA-LSTM without pauses) were also trained in [17].

The model that holds the previous best result on Estonian has publicly available source code, so we train the first stage part of it on English for comparison (T-LSTM). We used the same hyperparameters for T-LSTM that were used in [17]. Two-stage training requires out-of-domain data which was not used by [5] and would give our models an unfair advantage.

3.3. Datasets

3.3.1. Estonian

The Estonian dataset we use consists of two parts — a 334M word out-of-domain written text (e. g. newspapers and WWW) corpus and a 1M word in-domain pause annotated speech transcripts (broadcast news and conversations, lectures) corpus. The development and test set consist of 27K and 30K words respectively. The best result so far on this dataset was obtained by [17] and the details of the dataset can be found there.

3.3.2. English

Experiments on English are performed on the IWSLT dataset which consists of TED Talks transcripts. The current best result on this dataset was achieved by [5]. We use the same training, development and test set to train and test our models. The training and development set consist of 2.1M and 296K words respectively and come from the IWSLT2012 machine translation track training data. IWSLT2011 reference and ASR test set are used for testing and contain about 13K words each. More detailed description of the dataset can be found in [5].

3.4. Metrics and results

All models are evaluated in terms of per punctuation and overall precision, recall and F_1-score. We also report the overall slot error rate (SER), as F_1-score has been shown to have some undesirable properties [34]. All comparisons in this section are in terms of absolute differences.

On the Estonian test sets (Table 1), it is clear that our newly proposed BRNN models outperform the previous best LSTM based models despite having to deal with an additional type of punctuation. Our best model (TA-BRNN-p) achieves an overall F_1-score improvement by 2.5% on reference text and 1.8% on ASR output, when compared to the the previous best (TA-LSTM-p). SER is reduced by 4.4% and 2.6% on reference and ASR text respectively. The improvements are even larger and the comparison more fair when we map all question marks to periods (Q=P). Detailed metrics show that the TA-BRNN-p model is better than TA-LSTM-p in all aspects except comma restoration precision, but the difference is small. The gap between the text-only models (TA-BRNN and TA-LSTM), that did not use the pause duration information during second stage training, is even bigger — F_1-score improves by $3.8-4.5\%$ and SER by $3.4 - 6.4\%$. The improvements with the text only TA-BRNN model seem to mostly come from its much higher recall for periods (by $18.1 - 21.2\%$ higher than TA-LSTM) without sacrificing precision. As the previous best model (TA-LSTM-p) used only the next word and the preceding context and many commas in Estonian depend on a very local context (the next word), we conclude that the improved period restoration is the benefit of our model's ability to flexibly utilize the entire context in both directions.

The results on English test sets (Table 2) show even larger differences. The overall F_1-score improves by 8.9% on reference text and by 10.5% on ASR output when comparing our T-BRNN model to the best baseline (DNN-A). The best baseline in terms of SER is the DNN model from [5] and the T-BRNN model reduces it by 11.6% on reference text and by 15.5% on ASR output. The T-BRNN model shows improvements in all metrics for all punctuation types. The biggest difference is in the question mark restoration performance, as the models from [5] were unable to restore any question marks thanks to a limited fixed size context (3 words before and 2 words after the slot) that rarely included the question indicating words that often are in the beginning of the sentence. Our newly proposed T-BRNN model and the T-LSTM model from [17] were both able to restore question marks, as their preceding context length is not limited to a fixed size. The T-LSTM

[2]`http://nlp.stanford.edu/projects/glove/`

Table 1: *Results on Estonian reference transcripts and ASR output test set. T-LSTM-p, TA-LSTM and TA-LSTM-p are the best models from [17], TA-BRNN and TA-BRNN-p are our models, and (Q=P) indicates that question marks have been mapped to periods.*

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		Pr.	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*SER*
Ref.	T-LSTM-p [17]	78.5	63.3	70.1	68.9	59.8	64.0	-	-	-	75.6	62.3	68.3	52.8
	TA-LSTM [17]	74.5	72.2	73.3	62.8	42.9	51.0	-	-	-	71.9	63.9	67.7	52.5
	TA-LSTM-p [17]	**82.3**	69.9	75.6	67.7	76.8	72.0	-	-	-	77.3	71.9	74.5	43.7
	TA-BRNN	75.1	**75.5**	75.3	65.6	64.1	64.8	**63.6**	43.8	51.9	72.5	71.9	72.2	46.1
	TA-BRNN-p	81.6	75.4	**78.4**	72.5	77.0	74.7	59.1	**48.7**	**53.4**	78.6	75.4	77.0	39.3
	TA-BRNN (Q=P)	75.1	**75.5**	75.3	67.4	64.7	66.0	-	-	-	73.0	72.4	72.7	45.6
	TA-BRNN-p (Q=P)	81.6	75.4	**78.4**	**73.8**	**77.3**	**75.5**	-	-	-	**79.2**	**76.0**	**77.6**	**38.7**
ASR	T-LSTM-p [17]	69.9	57.3	63.0	57.3	49.0	52.8	-	-	-	66.2	55.0	60.1	68.5
	TA-LSTM [17]	64.5	64.6	64.6	48.8	32.2	38.8	-	-	-	61.3	55.5	58.2	72.1
	TA-LSTM-p [17]	**71.1**	62.5	66.5	54.9	61.2	57.8	-	-	-	65.7	62.1	63.8	65.5
	TA-BRNN	63.9	**67.9**	65.8	54.0	50.3	52.1	48.8	29.9	37.0	61.3	62.6	62.0	68.7
	TA-BRNN-p	69.1	66.8	**68.0**	59.7	**61.5**	60.6	**51.2**	**31.3**	**38.9**	66.3	64.9	65.6	62.9
	TA-BRNN (Q=P)	63.9	**67.9**	65.8	54.9	50.2	52.4	-	-	-	61.6	62.9	62.2	68.4
	TA-BRNN-p (Q=P)	69.1	66.8	**68.0**	**60.6**	61.1	**60.8**	-	-	-	**66.6**	**65.2**	**65.9**	**62.6**

Table 2: *Results on English reference transcripts and ASR output test set. DNN, DNN-A and CNN-2A are the best models from [5], T-LSTM is first stage model from [17] that we trained on the English dataset, and T-BRNN and T-BRNN-pre are our models.*

	Model	*COMMA*			*PERIOD*			*QUESTION*			*OVERALL*			
		Pr.	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*Pr.*	*Re.*	F_1	*SER*
Ref.	DNN [5]	58.2	35.7	44.2	61.6	64.8	63.2	0	0	-	60.3	48.6	53.8	62.9
	DNN-A [5]	48.6	42.4	45.3	59.7	68.3	63.7	0	0	-	54.8	53.6	54.2	66.9
	CNN-2A [5]	48.1	44.5	46.2	57.6	69.0	62.8	0	0	-	53.4	55.0	54.2	68.0
	T-LSTM [17]	49.6	41.4	45.1	60.2	53.4	56.6	57.1	43.5	49.4	55.0	47.2	50.8	74.0
	T-BRNN	64.4	45.2	53.1	72.3	71.5	71.9	67.5	58.7	62.8	68.9	58.1	63.1	51.3
	T-BRNN-pre	**65.5**	**47.1**	**54.8**	**73.3**	**72.5**	**72.9**	**70.7**	**63.0**	**66.7**	**70.0**	**59.7**	**64.4**	**49.7**
ASR	DNN [5]	47.2	32.0	38.1	59.0	60.9	60.0	0	0	-	54.4	45.6	49.6	73.3
	DNN-A [5]	41.0	40.9	40.9	56.2	64.5	60.1	0	0	-	49.2	51.6	50.4	79.2
	CNN-2A [5]	37.3	40.5	38.8	54.6	65.5	59.6	0	0	-	46.4	51.9	49.1	83.6
	T-LSTM [17]	41.8	37.8	39.7	56.4	49.3	52.6	55.6	42.9	48.4	49.1	43.6	46.2	83.7
	T-BRNN	**60.0**	**45.1**	**51.5**	69.7	69.2	69.4	**61.5**	45.7	52.5	65.5	57.0	60.9	57.8
	T-BRNN-pre	59.6	42.9	49.9	**70.7**	**72.0**	**71.4**	60.7	**48.6**	**54.0**	**66.0**	**57.3**	**61.4**	**57.0**

model showed the lowest period restoration scores, indicating that looking further than one word into the following context is important for sentence boundary detection. The T-BRNN model showed improvements despite the fact that DNN, DNN-A and CNN-2A used an external source of information in the form of pre-trained word vectors (trained on 6B tokens). When we use the same word vectors as an initialization for our word embeddings, then we get further improvements and achieve our best result (T-BRNN-pre).

The comparison of the Estonian and English results reveals that comma restoration is a much more difficult task in English than it is in Estonian. This does not come as a surprise, as many commas in Estonian can be restored by following relatively simple rules based on the next word. Although there is a big difference in question mark restoration performance as well, it is hard to make conclusions as they are too rare in both test sets.

To better understand the individual contributions of bidirectionality and attention, we trained additional models on English with either of the components removed. Bidirectionality turned out to be the biggest factor, as removing the forward context caused the performance of all punctuation marks (especially periods and question marks) to drop. Removing attention had much smaller effect, hurting mostly question mark restoration.

4. Conclusions

This paper presented a bidirectional recurrent neural network with attention mechanism for restoring commas, periods and question marks in unsegmented transcribed speech. Both a purely textual approach and an approach combining textual features with prosodic information were used. Experiments on Estonian and English showed improvements for all punctuation types compared to the state-of-the-art. The overall F_1-score was improved by $1.8 - 10.5\%$ absolute and slot error rate was reduced by $2.6 - 15.5\%$. The biggest improvements were achieved when comparing text-only models.

Future research includes the use of a richer set of prosodic features, training an English model on a larger dataset, and exploring joint punctuation and capitalization models.

5. Acknowledgements

This study was supported by the National Program for Estonian Language Technology funded by the Estonian Ministry of Education and Research, and by the European Regional Development Fund (Centre of Excellence in Estonian Studies). The authors would also like to thank Xiaoyin Che and the other authors of [5] for the support with the IWSLT dataset.

6. References

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

[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] D. Zhang, S. Wu, N. Yang, and M. Li, "Punctuation prediction with transition-based parsing." in *ACL (1)*, 2013, pp. 752–760.

[5] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *The 10th International Conference on Language Resources and Evaluation (LREC)*, 2016.

[6] S. Peitz, M. Freitag, A. Mauser, and H. Ney, "Modeling punctuation prediction as machine translation." in *IWSLT*, 2011, pp. 238–245.

[7] E. Cho, J. Niehues, K. Kilgour, and A. Waibel, "Punctuation insertion for real-time spoken language translation," *Proceedings of the Eleventh International Workshop on Spoken Language Translation*, 2015.

[8] 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.

[9] T. Levy, V. Silber-Varod, and A. Moyal, "The effect of pitch, intensity and pause duration in punctuation detection," in *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of.* IEEE, 2012, pp. 1–4.

[10] J. Kolář, E. Shriberg, and Y. Liu, "Using prosody for automatic sentence segmentation of multi-party meetings," in *Text, Speech and Dialogue*. Springer, 2006, pp. 629–636.

[11] 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.

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

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

[14] F. Batista, D. Caseiro, N. Mamede, and I. Trancoso, "Recovering punctuation marks for automatic speech recognition," in *Interspeech 2007*, Antwerp, Belgium, 2007.

[15] ——, "Recovering capitalization and punctuation marks for automatic speech recognition: Case study for portuguese broadcast news," *Speech Communication*, vol. 50, no. 10, pp. 847–862, 2008.

[16] F. Batista, H. Moniz, I. Trancoso, and N. Mamede, "Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 2, pp. 474–485, 2012.

[17] O. Tilk and T. Alumäe, "LSTM for punctuation restoration in speech transcripts," in *Interspeech 2015*, Dresden, Germany, 2015.

[18] 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.

[19] X. Wang, H. T. Ng, and K. C. Sim, "Dynamic conditional random fields for joint sentence boundary and punctuation prediction." in *INTERSPEECH*, 2012, pp. 1384–1387.

[20] C. Xu, L. Xie, G. Huang, X. Xiao, E. Chng, and H. Li, "A deep neural network approach for sentence boundary detection in broadcast news." in *INTERSPEECH*, 2014, pp. 2887–2891.

[21] E. Matusov, A. Mauser, and H. Ney, "Automatic sentence segmentation and punctuation prediction for spoken language translation." in *IWSLT*. Citeseer, 2006, pp. 158–165.

[22] O. Khomitsevich, P. Chistikov, T. Krivosheeva, N. Epimakhova, and I. Chernykh, "Combining prosodic and lexical classifiers for two-pass punctuation detection in a russian asr system," in *Speech and Computer*. Springer, 2015, pp. 161–169.

[23] M. Hasan, R. Doddipatla, and T. Hain, "Multi-pass sentence-end detection of lecture speech." in *INTERSPEECH*, 2014, pp. 2902–2906.

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

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR2015, arXiv:1409.0473*, 2015.

[26] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

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

[28] T. Wang and K. Cho, "Larger-context language modelling," *arXiv preprint arXiv:1511.03729*, 2015.

[29] 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.

[30] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[32] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation.

[33] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[34] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel *et al.*, "Performance measures for information extraction," in *Proceedings of DARPA broadcast news workshop*, 1999, pp. 249–252.