INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Online Punctuation Restoration using ELECTRA Model for streaming ASR Systems

Martin Polacek[1], *Petr Cerva*[1], *Jindrich Zdansky*[1], *Lenka Weingartova*[2]

[1]Technical University of Liberec, Studentska 2, 461 17 Liberec, Czech Republic
[2]NEWTON Technologies, Na Pankraci 1683/127, 140 00 Praha 4, Czech Republic

martin.polacek@tul.cz, petr.cerva@tul.cz

Abstract

In this work, we propose a lightweight online approach to automatic punctuation restoration (APR), which can be utilized in speech transcription systems for, e.g., live captioning TV or radio streams. It uses only text input without prosodic features and a fine-tuned ELECTRA-Small model with a two-layer classification head. It allows for restoring question marks, commas, and periods with a very short inference time and a low latency of just three words. Our APR scheme is first tuned and compared to other architectures on a set of manual TV news transcripts. The resulting system is then compared to another real-time APR module utilizing a recurrent network and a combination of text and acoustic features. The test data we use contains automatic transcripts of radio talks and TV debates; we are also publishing this data. The results show that our APR module performs better than the above-mentioned system and yields on the two test sets an average F1 of 71.2% and 69.4%, respectively.
Index Terms: automatic punctuation restoration, automatic speech recognition, ELECTRA model, streamed data

1. Introduction

In the most recent decade, various automatic speech recognition (ASR) systems have become parts of our everyday lives. Their existing applications include voice dictation, live captioning, virtual assistants, conversational systems, and various services that enable the transcription of individual recordings or even streamed audio data. These developments in ASR are made possible by advanced deep learning methods, namely by deploying the recent end-to-end (E2E) neural network architectures [1], which have significantly increased ASR accuracy.

However, ASR systems are usually not designed to produce transcripts with punctuation marks. This is because the data for training conventional or E2E models are rarely annotated with punctuation marks. Missing punctuation negatively affects user experience, especially in the above-mentioned systems for live captioning [2] or transcription of audio data. In all these applications, users find it difficult to read or correct a long transcript without sentence boundaries [3]. Moreover, the absence of punctuation can degrade the results of other downstream tasks, e.g., translation or context analysis, which use ASR output and usually require sentence boundaries to be defined.

To address this issue, ASR systems are often complemented by an automatic punctuation restoration (APR) module, which adds selected punctuation marks to the stream of recognized words. Existing approaches for APR utilize two main types of features [4]: acoustic (prosodic) [5] and text-based. The methods of the former type require less training data and are more robust to ASR errors as well. However, they require training data annotations on the acoustic level and often yield lower accuracy. In contrast, the latter methods provide more accurate punctuation marks and can be trained using text data only. Their disadvantage is higher computing requirements and higher latency related to a larger input context, which may incorporate future words. Note that textual and acoustic features can also be combined, such as in [6] or [7].

2. Related work

The APR task is usually solved as a sequence labeling problem aimed at restoring only the three most frequent punctuation symbols, i.e., periods (”.”), commas (”,”), and question marks (”?”). For this purpose, various statistical methods have been employed, such as maximum entropy models [4, 8] or conditional random fields [9].

Consequently, advancements in deep learning techniques have shifted the research in this direction with excellent results. For example, convolutional neural networks and pre-trained word vectors have been used in [10] for punctuation prediction in unsegmented transcripts. Other works have utilized recurrent neural networks (RNNs), allowing for sequence-to-sequence modeling. Among others, unidirectional RNN was trained in [11] or in [4], and bidirectional RNN with attention mechanism in [12] or [13]. Transformer-based models have also been adopted for the APR task, such as in [14]. Significant progress has been achieved using large pre-trained transformer language models (LMs), such as BERT [15]. Moreover, excellent results have been reported using the statistical information encoded in these models, e.g., in [16, 17].

To further improve APR results, ELECTRA model [18], a recently proposed improvement on BERT, has also been employed. This architecture has a generator-discriminator structure allowing for automatic injection of errors into the training data and subsequent training with a multi-task learning objective to improve the robustness against ASR errors [19]. A further attempt to improve the robustness has been proposed in [20], where the authors have utilized the ELECTRA model with a disfluency generator and a multi-task discriminator.

This work focuses on the most challenging APR sub-task, i.e., real-time streaming data processing with low latency. Our specific use case is live captioning of various TV and radio (TV/R) broadcasts in the Czech language. The number of existing papers investigating online APR with regard to latency is rather limited. For example, the authors in [6] utilize RNN and rely on a combination of prosodic and textual features. Their approach has a latency of three words. Another recent work [21] utilizes decoding over output from a controllable time-delay transformer model that jointly completes the punctuation prediction and disfluency detection tasks. This method operates with a latency of 10 words.

10.21437/Interspeech.2023-664

In contrast, our approach utilizes text features only to avoid the need for training data annotations at the acoustic level while maintaining as high accuracy as possible. It thus utilizes the recent ELECTRA model, but with a limited size and future input context, to allow for real-time processing with low computation demands and a latency of three words.

In an experimental study performed on a development set, we first investigate individual components of our APR module and compare its performance to other architectures or existing systems, including the OpenAI web service, which all utilize purely textual input. This study is not solely concerned with accuracy as in most existing works but also considers inference time and a streaming scenario. After that, we evaluate our approach with tuned parameters using transcripts created by an ASR system. We also compare the achieved results to another real-time APR system combining textual and prosodic features.

3. Proposed APR approach

3.1. Neural network architecture used

The neural network architecture adopted in this work corresponds to the ELECTRA-Small model [18]. Experimental investigation on the development set in Sec. 4.3 shows that this model achieves excellent results with a small number of parameters and, consequently, low computational requirements.

For the APR task, this model as well as all of the similar compared architectures must be complemented by a classification head to classify each output token into one of the target classes. We distinguish four classes. The first, denoted by None, represents a case where no punctuation mark should complement the input word. The remaining classes represent the three most common punctuation marks: periods, commas, and question marks. The classification head used in this work consists of two linear layers. The first layer with SELU [22] activation function takes a vector with a size of 256 as input and provides a vector with a size of 512 on output. The second layer then produces a vector of size 4, passed to a softmax function to determine the probability score for each class. We have also tried the utilization of additional layers or different activation functions, but without any significant improvement in accuracy.

3.2. Training data pre-processing and tokenization

All data used to train the ELECTRA models have been processed as follows: First, solely alphabetic characters, digits, and punctuation marks are preserved, and all other symbols are removed. Second, all capital letters are converted to lowercase. Finally, the text is split into sentences and tokenized using the SentencePiece tokenizer [23] operating with a dictionary containing 30,522 tokens, as recommended in [18].

3.3. Fine-tuning

The fine-tuning of the general model must be performed together with the newly added classification layers. The data used for this purpose is processed as follows. First, random numbers of sentences (from 1 to 10) are concatenated into long strings corresponding to the size of the input layer of the model. The strings with similar word counts are then organized into batches and the data in each batch is normalized and tokenized as described in Sec. 3.2. After that, a vector containing a number between 0 and 3 is created for each token. Its value represents the class of the punctuation, that follows the given token in the input text. Finally, all tokenized punctuation marks are removed.

3.4. Method of streamed data processing

In an offline processing mode, input text can be divided into blocks, and punctuation restoration can be performed for each block individually. This segmentation can also be performed with an overlay, so that words on block boundaries are processed within an adequate context.

However, this block-processing scheme is not suitable for streamed environments. In the latter, the punctuation marks must be restored for every input word with the lowest possible latency, i.e., with only a limited future context. Therefore, the context of every input word is usually formed by a long sequence of preceding words and only several following words, with the maximum total length corresponding to the number of input tokens. An experimental comparison of this streamed way of processing with the per-block regime is presented in Sec. 4.4.

3.5. Class weighting

The number of tokens in each class in our training data is significantly imbalanced, as the occurrence frequency of the None class is much higher than for periods, commas, or even question marks. To eliminate this problem, it is necessary to determine the weight for each class, which allows us to achieve an optimal balance between precision and recall [**?**]. The resulting weights, determined using the dataset as described in Sec. 4.2, are as follows: 1.0 for the None class, 5.0 for question marks, 2.0 for periods, and 1.5 for commas. They are used to multiply the output probabilities of the model during inference.

4. Investigation on the development set

4.1. Development dataset and evaluation metrics

Our development set consists of manually corrected transcripts of Czech TV/R news. This data contains 258,608 tokens in total, of which the class None comprises 225,443 tokens, the question mark class 595 tokens, the period class 17,103 tokens, and the comma class 15,467 tokens. The metrics chosen to evaluate the results from an accuracy point of view are precision (P), recall (R), and F1. Since high scores are to be expected for class None, these metrics are computed for tokens corresponding to the three punctuation marks only.

4.2. Training data and parameters used for training

The corpus used to train the ELECTRA models contains 23 GB of Czech texts (i.e., 5 billion tokens), including newspaper articles, manually corrected automatic transcripts of various Czech TV/R broadcasts, diploma theses, and legal texts and judgments. To obtain the best possible accuracy in the target domain, the data used for the fine-tuning comprises an additional 1.5 GB of manually corrected transcripts (i.e., 335 million tokens). Another 100 MB of these transcripts is used to find the weights for individual punctuation classes.

We perform the fine-tuning using the AdamW optimizer in three epochs with a learning rate (LR) of 0.00003 for all investigated models and an LR of 0.0001 for the classification head. Starting with epoch two, the LR is decreased to 95% of the previous LR every 10,000 batches. The training of ELECTRA models from scratch follows a recipe presented in [18].

4.3. Comparison of various architectures

In the first experiment, several neural network architectures are compared with respect to the accuracy, model size, and infer-

Table 1: *Comparison of performance of various architectures in the offline block-processing mode.*

architecture	P [%]	R [%]	F1 [%]	# params	inf. time [ms]
LSTM + BiLSTM + linear layers	51.5	34.5	41.3	8.5M	3
ELECTRA-Small trained from scratch	75.3	76.8	**76.0**	13.6M	11
ELECTRA-Base trained from scratch	75.3	75.5	75.4	109M	60
pre-trained ELECTRA-Small from [24]	73.4	**78.7**	**76.0**	13.6M	11
pre-trained BERT from [25]	75.4	74.9	75.2	110M	61
GPT-3-based web service in an edit mode	**80.5**	54.6	65.1	-	-

ence time. This time is calculated using a single thread of one core of an Intel i7 9700K processor and corresponds to the time needed to perform one forward pass using the given architecture. The experiment has been performed in the above-described block-processing mode without any overlays.

The first architecture consists of LSTM, BiLSTM and GRU layers as proposed in[1]. Here, the first layer corresponds to an embedding layer with a size equal to the number of tokens in the dictionary. This layer is followed by the LSTM layer and then by alternating Bi-LSTM and GRU layers, gradually reducing the number of tokens to a final number of four. This model, as well as the following two ELECTRA models (second and third row in Table 1), has been trained from scratch and subsequently fine-tuned. The fourth model is again an ELECTRA-Small one, but this time trained by authors of [24] using a much larger corpus compiled from 253 GB of Czech text data. The next model is the BERT model pre-trained using 36 GB of Czech text data [25]. These last two mentioned models have been complemented with the two-layer classification head and fine-tuned on our data. Note that the class weighting has been applied to all of the above-described architectures as specified in Sec. 3.5.

Finally, the last investigated approach corresponds to the general-purpose web service of OpenAI, which utilizes the GPT-3 model [26]. This service has been evaluated in an edit mode, which allows us to insert content into the existing text rather than just complete the existing text.[2] In our particular use case, every data block passed to this service has been complemented with the following instruction: "Add punctuation to the text. Use periods, commas, and question marks."

The results in Table 1 clearly show that all of the transformers outperform the LSTM/GRU-based model by a large margin. For example, ELECTRA-Small model, trained from scratch, yields an F1 of 76%, while the LSTM/GRU-based model has an F1 of only 41.3%. The value of 76% is the same as that valid for the ELECTRA-Small model pre-trained using 253 GB of text data. The use of this much larger corpus thus does not bring any further improvement in the accuracy of this model.

It is also evident that both ELECTRA-Small models yield slightly higher F1 values than the pre-trained BERT model (F1 of 75.2%), which has a much higher number of parameters (110M versus 13.6M) and thus also a higher inference time (61 ms versus 11 ms). The same also holds for the bigger ELECTRA-Base model. However, its slightly worse performance compared to the small-sized variants may have been caused by the smaller size of our training corpus.

Our results also show that the general GPT-3 based web service, currently of great interest and increasingly used for various tasks, performs significantly worse for APR purposes, i.e., with F1 of just 65.1%, than the specialized models.

[1]https://community.wolfram.com/groups/-/m/t/1379001

[2]https://openai.com/blog/gpt-3-edit-insert/

Table 2: *Detailed results [%] achieved by the ELECTRA-Small model for individual punctuation marks.*

	P [%]	R [%]	F1 [%]
question mark	46.2	38.6	42.0
period	76.9	70.9	73.8
comma	74.6	84.7	79.2
weighted average	75.3	76.8	76.0

It should also be noted that the inference time of 11 ms of the best-performing ELECTRA-Small model is sufficient for real-time processing. The reason is that the average speaking rate for many European languages is reported to be between 100 and 200 words per minute (i.e., 1.7-3.3 words per second), depending on the speaking style and word length. These rates correspond to the maximum possible inference times from 590 ms to 300 ms. Czech, a morphologically rich inflectional language, has an above-average word length, so a word-per-minute rate on the lower boundary is expected.

Finally, Table 2 shows the detailed performance of the ELECTRA-Small model trained from scratch for individual punctuation marks. The highest F1 of 79.2% is achieved for commas, a slightly lower value of 73.8% for periods, and question marks are restored with a low F1 value of 42%. These differences show that restoring a question mark's position in Czech is almost twice as challenging than restoring a comma or a period.

4.4. Performance in the streaming mode

The next performed experiment investigates the performance of the ELECTRA-Small model trained from scratch in the streaming mode of processing as described in Sec. 3.4. In contrast to the previous experiments, the APR task is thus not carried out for individual text blocks. The maximum allowed left input context size is 100 words, and the maximum allowed right context size varies from 1 to 100 words. This gives the total maximum context size of 200 words corresponding to approximately 512 input tokens (i.e., the size of the input layer of the ELECTRA-Small model). Note that the sizes actually used depend on each word's position in the input text; the left context of a word at the beginning, or the right context of a word at the end, can be shorter than the maximum set values.

The achieved results (see Table 4) show that the right context of just one word is insufficient from the accuracy point of view – the corresponding F1 value is just 70.2%. On the contrary, three or more words yield F1 values just by 1% smaller than those achieved in the block-processing mode (the last row of Table 4), and 10 or more words are sufficient without any loss of accuracy.

Table 3: *Detailed comparison of the proposed APR module in online mode to the APR module from [6] on outputs from an ASR system.*

	read radio talks						spontaneous TV debates					
	proposed APR module			APR module from [6]			proposed APR module			APR module from [6]		
	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]	P [%]	R [%]	F1 [%]
question mk	31.7	59.4	**41.4**	40.0	7.5	12.7	48.5	39.9	**43.8**	75.8	22.8	35.0
period	82.6	66.7	**73.8**	69.6	54.0	60.8	67.5	35.7	46.7	68.8	58.4	**63.2**
comma	60.2	82.3	**69.5**	61.7	66.8	64.1	75.8	84.1	**79.7**	83.3	72.7	77.6
average	68.7	73.8	**71.2**	65.2	59.4	62.1	73.2	65.9	69.4	78.2	66.0	**71.6**
one class	80.7	88.3	**84.3**	77.5	69.8	73.4	93.1	85.3	**89.0**	92.2	75.9	83.3

Table 4: *Results [%] of the proposed APR module in the streaming mode for different max. sizes of left and right input context.*

max. left cont.	max. right cont.	P	R	F1
100	1	73.3	67.4	70.2
100	2	75.0	72.5	73.7
100	3	75.3	74.2	74.7
100	4	75.2	75.1	75.1
100	5	75.5	75.5	75.5
100	10	**76.0**	76.0	**76.0**
100	100	75.6	73.7	74.6
block-processing with no overlay		75.3	**76.8**	**76.0**

5. Results for streamed ASR transcripts

In our last experiment, we compare the results of the best performing ELECTRA-Small model trained from scratch to a reference system – a real-time APR module for Czech from [6], which is based on RNN with LSTM and utilizes word embeddings, prosodic (mainly temporal) features, and information about silence extracted from the speech signal. The latency of both approaches is three words.

This comparison is performed on a test set compiled from recordings and corresponding automatic transcripts with manually added punctuation. The set contains 10 radio talks on current world events (read or prepared speech) and 5 TV political debates (spontaneous speech). The former data contains 31,932 words, 53 question marks, 2,113 periods, and 1,954 commas; the latter 39,087 words, 289 question marks, 2,724 periods, and 4,868 commas. We make all these transcripts publicly available,[3] including the corresponding recordings and annotations. We have created them using an ASR system for live captioning TV/R streams. It utilizes a modified RNN-transducer with stateless prediction network [27] trained with the aid of the Icefall toolkit[4] on more than 15k hours of Czech recordings. It has a latency of around one second and yields a word error rate (WER) of 3.9% on radio talks and a WER of 5.4% on TV debates.

All achieved results are summarized in Table 3. We can see that our APR module, operating in streaming mode with the maximum left and right context of 100 words and 3 words, respectively, has F1 values on radio talks higher by a large margin than the values valid for the reference APR module. This holds for all three individual punctuation marks, averaged F1 values, and F1 values calculated by merging all three punctuation classes into one class (the last row in Table 3). We can also see that our F1 value of 84.3% for a one-class scenario is much higher than the averaged F1 value of 71.2% calculated over all three classes. This observation holds for both systems and datasets, and the big difference between these values shows to what degree it is easier to determine where a punctuation mark should be than to decide precisely which one it should be. Note that the one class scenario corresponds to the identification of Sentence-Like Units (SU) as proposed in [3, 28].

The results on the second set show a small decrease in averaged F1 value for our APR module: the achieved value of 69.4% is slightly lower than 71.6% yielded by the reference system. The reason is that spontaneous TV debates contain large portions of long-lasting single-speaker utterances. These are often very complex sentences with many conjunctions, where even manual annotation of commas and periods is ambiguous. It is thus difficult to automatically determine whether a period or a comma should be placed after each sub-sentence without the associated acoustic information. Our approach achieves an F1 value of just 46.7% for periods. However, it outperforms by a large margin the reference system in the one-class scenario. This means that the proposed APR module can, in transcripts of TV debates, detect more correct positions for the addition of a punctuation mark. It also achieves a higher accuracy in restoring question marks and commas in this data type.

6. Conclusions

We have proposed a lightweight approach to the streamed APR, which relies on the ELECTRA-Small model and operates purely over a text input with a latency of just three words. It is therefore suitable for online scenarios, where it performs with almost the same accuracy as in the block-processing regime. Our results suggest that a) low-frequency question marks are much harder to restore than commas and b) the performance for periods is mixed since their detection is accurate in transcripts of read speech but very difficult in spontaneous data, such as TV debates containing many complex long sentences. This scenario also represents the only case where our approach has been slightly outperformed by an APR system combining textual and prosodic features. However, utilizing purely textual data is significantly less demanding from a data preparation point of view.

7. Acknowledgements

The research leading to these results has received funding from the Norway Grants and the Technology Agency of the Czech Republic within the KAPPA Program (project No. TO01000027). Computational resources were provided by the e-INFRA CZ project (ID:90140), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

[3] https://owncloud.cesnet.cz/index.php/s/fHqtWwZ5G9V5pPN

[4] https://github.com/k2-fsa/icefall

8. References

[1] J. Li, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, pp. 1–64, 2022.

[2] M. A. Tundik, G. Szaszak, G. Gosztolya, and A. Beke, "User-centric evaluation of automatic punctuation in ASR closed captioning," in *Interspeech 2018, Hyderabad, India, September 2-6, 2018*. ISCA, 2018, pp. 2628–2632.

[3] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper, "Enriching speech recognition with automatic detection of sentence boundaries and disfluencies," *IEEE Transactions on Audio, Speech, and Language Processing*, no. 5, pp. 1526–1540, 2006.

[4] M. A. Tundik, B. Tarjan, and G. Szaszak, "Low latency maxent- and rnn-based word sequence models for punctuation restoration of closed caption data," in *SLSP 2017, Le Mans, France, October 23-25, 2017*, ser. Lecture Notes in Computer Science, vol. 10583. Springer, 2017, pp. 155–166.

[5] D. Wang and S. S. Narayanan, "A multi-pass linear fold algorithm for sentence boundary detection using prosodic cues," in *ICASSP 2004, Montreal, Quebec, Canada, May 17-21, 2004*. IEEE, 2004, pp. 525–532.

[6] P. Hlubik, M. Spanel, M. Bohac, and L. Weingartova, "Inserting punctuation to ASR output in a real-time production environment," in *TSD 2020, Brno, Czechia, September 8-11, 2020*, ser. Lecture Notes in Computer Science, vol. 12284. Springer, 2020, pp. 418–425.

[7] O. Klejch, P. Bell, and S. Renals, "Sequence-to-sequence models for punctuated transcription combining lexical and acoustic features," in *ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 5700–5704.

[8] J. Huang and G. Zweig, "Maximum entropy model for punctuation annotation from speech," in *ICSLP2002 – Interspeech 2002, Denver, Colorado, USA, September 16-20, 2002*. ISCA, 2002.

[9] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *EMNLP 2010, Massachusetts, USA, October 9-11, 2010*. ACL, 2010, pp. 177–186.

[10] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *LREC 2016, Portoroz, Slovenia, May 23-28, 2016*. ELRA, 2016.

[11] O. Tilk and T. Alumae, "LSTM for punctuation restoration in speech transcripts," in *Interspeech 2015, Dresden, Germany, September 6-10, 2015*. ISCA, 2015, pp. 683–687.

[12] ——, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *Interspeech 2016, San Francisco, CA, USA, September 8-12, 2016*. ISCA, 2016, pp. 3047–3051.

[13] S. Kim, "Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration," in *ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 7280–7284.

[14] B. P. Nguyen, V. B. H. Nguyen, H. Nguyen, P. N. Phuong, T. Nguyen, Q. T. Do, and L. C. Mai, "Fast and accurate capitalization and punctuation for automatic speech recognition using transformer and chunk merging," in *O-COCOSDA 2019, Cebu, Philippines, October 25-27, 2019*. IEEE, 2019, pp. 1–5.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019*. ACL, 2019, pp. 4171–4186.

[16] Y. Cai and D. Wang, "Question mark prediction by bert," in *APSIPA ASC 2019, Lanzhou, China, November 18-21, 2019*. IEEE, 2019, pp. 363–367.

[17] K. Makhija, T. Ho, and E. S. Chng, "Transfer learning for punctuation prediction," in *APSIPA ASC 2019, Lanzhou, China, November 18-21, 2019*. IEEE, 2019, pp. 268–273.

[18] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," in *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[19] M. Hentschel, E. Tsunoo, and T. Okuda, "Making punctuation restoration robust and fast with multi-task learning and knowledge distillation," in *ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 7773–7777.

[20] W. Wang, Y. Liu, W. Jiang, and Y. Ren, "Making punctuation restoration robust with disfluency detection," in *CSCWD 2022, Hangzhou, China, May 4-6, 2022*. IEEE, 2022, pp. 395–399.

[21] Q. Chen, M. Chen, B. Li, and W. Wang, "Controllable time-delay transformer for real-time punctuation prediction and disfluency detection," in *ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 8069–8073.

[22] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *NIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, 2017, pp. 971–980.

[23] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *EMNLP 2018, Brussels, Belgium, October 31 - November 4, 2018*. ACL, 2018, pp. 66–71.

[24] M. Kocian, J. Naplava, D. Stancl, and V. Kadlec, "Siamese BERT-based model for web search relevance ranking evaluated on a new czech dataset," in *AAAI 2022, IAAI 2022, EAAI 2022, Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 12 369–12 377.

[25] J. Sido, O. Prazak, P. Priban, J. Pasek, M. Sejak, and M. Konopik, "Czert - Czech BERT-like model for language representation," in *RANLP 2021, Virtual Event, September 1-3, 2021*. INCOMA Ltd., 2021, pp. 1326–1338.

[26] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *NIPS 2020, Virtual event, December 6-12, 2020*, 2020.

[27] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-transducer with stateless prediction network," in *ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 7049–7053.

[28] M. Ostendorf, B. Favre, R. Grishman, D. Hakkani-Tur, M. Harper, D. Hillard, J. Hirschberg, H. Ji, J. G. Kahn, Y. Liu, S. Maskey, E. Matusov, H. Ney, A. Rosenberg, E. Shriberg, W. Wang, and C. Wooters, "Speech segmentation and spoken document processing," *IEEE Signal Processing Magazine*, no. 3, pp. 59–69, 2008.