Transfer Learning for Punctuation Prediction

Karan Makhija*† Thi-Nga Ho† and Eng-Siong Chng†
* Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani
† School of Computer Science and Engineering, Nanyang Technological University, Singapore
f2014867@pilani.bits-pilani.ac.in, ngaht@ntu.edu.sg, ASESChng@ntu.edu.sg

***Abstract*—The output from most of the Automatic Speech Recognition system is a continuous sequence of words without proper punctuation. This decreases human readability and the performance of downstream natural language processing tasks on ASR text. We treat the punctuation prediction task as a sequence tagging task and propose an architecture that uses pre-trained BERT embeddings. Our model significantly improves the state of art on the IWSLT dataset. We achieve an overall F1 of 81.4% on the joint prediction of period, comma and question mark.**

I. Introduction

In recent years, speech recognition systems have achieved commendable performances, e.g., Word Error Rates (WER) at 5.5% on English conversational telephone speech [1] [2]. However, most of the current systems produce transcripts that are usually not punctuated. Restoring correct punctuation is essential as it can improve the readability of the transcripts [3] and the performance of downstream tasks like machine translation [4], information extraction and name entity recognition [5]. Also, most NLP models are trained on text with the inherent assumption of it being properly punctuated.

In last two decades, a substantial amount of research has been done on punctuation prediction and related tasks, e.g., sentence boundary detection and sentence unit detection. The works and approaches used are usually divided based on the features. There are approaches like [6]–[12] that use both prosody as well as lexical cues, while most use, only lexical features. This is because written data with punctuation is a lot more abundant as compared to audio data with processed prosodic features. Also, prosody cues are inconsistent and may vary from person to person [13]. Due to the above reasons, we too retain from using prosody cues in our model.

In this work, we treat the punctuation prediction task as a sequence labelling task, where label of each word indicates the punctuation present after the word. We proposed a two-layer model. The first layer is a pre-trained language representation model, i.e., BERT - Bidirectional Encoder Representations from Transformers - introduced by Devlin et. al. [14]. The second layer consists of a bidirectional Long Short Term Memory and a linear Conditional Random Field (CRF) Classifier for prediction.

Unlike pre-trained word embeddings which have same values in every situation, a language representation model likes BERT builds its representation vectors by considering the neighbouring context. Thus, each generated embedding captures the correct semantic meaning provided by the given example. Furthermore, BERT has the advantage of capturing deep representation of both left and right context, contrary to previous language representations that are unidirectional or shallow bidirectional. BERT has also shown to give state-of-the-art results when being applied for tasks such as question answering and language inference by just using fine tuning techniques and one additional output layer [14]. These advantages of BERT motivated us to apply it for our punctuation prediction task. The implementation of our model is made publicly available for reproducibility[1].

The rest of this paper can be summarized as follows. Section 2 discusses the related work. In section 3, we introduce our proposed architecture. Experiments and results are presented in section 4. We end with our conclusion and future work in section 5.

II. Related work

A. Punctuation Prediction for ASR Transcripts

Punctuation prediction for ASR transcripts and its related tasks have a long history of development. One of the earliest works on this topic is by Stolcke et al. [6]. The authors create an hidden event language model to estimate the conditional probability of events given words, where the events are sentence boundaries. Along with that they use a prosodic model that utilizes decision trees to model the conditional probability distribution of events given prosodic cues and words recognised from audio features. Various prosody features that are useful for sentence unit detection are extensively discussed in [8].

Liu et al. [7], investigate various approaches to model the task that include Hidden Markov Model (HMM), Maximum Entropy and Conditional Random Fields (CRF) models, they find discriminative models perform better in general. Specifically, CRF models generally outperform previous models because they utilize global sequence information for making the decision. Authors in [15] use a two layer factorial conditional random fields that learns the joint probability of predicting punctuation together with the sentence type.

Punctuation prediction can also be transformed into other similar NLP tasks, e.g., transition based parsing [16], [17] and machine translation [4]. In [4], authors assumes the source language does not contain any punctuation and the target language is the same language with punctuation. Other works that use this approach include [18] and [19].

[1] https://github.com/panda-baba/bert_punct

Majority of recent works in punctuation prediction task are based on deep learning approaches. For example, in [11] authors use word vectors as the only features and experiment with deep neural network (DNN), 1-dimensional convolution neural network (1D-CNN) and 2-dimensional convolution neural network (2D-CNN) layers to predict the punctuation classes. Tilk et al. [20], make use of bidirectional GRU architecture with late fused attention mechanism. They train the network in two stages; the first stage uses only textual features while the second stage takes into account pause durations and is trained on a smaller dataset. In [21], the authors also investigate the use of BiLSTMs for the task of punctuation prediction. They use a multilayer BiLSTM and append that with a CRF to make predictions. Work in [22], tries to use a hybrid architecture of CNN and BiLSTM for the same task. Further, Wang et al. [23] are the first ones to use transformer architecture for punctuation restoration. They treat it as a machine translation task where the decoder is modified to have two outputs. In our proposed models, we use the encoder part of the transformer which is significantly different than the above works.

B. Language Representation Models

Deep learning based language models have been very popular in recent years due to their superiority to traditional statistical n-gram language models. Various kind of deep neural network architectures and tuning methods have been proposed on the topic of language representation models including works presented in [24]–[28] and more recently in [14].

In [24] Dai et al. presented the idea of pre-training the whole neural network along with the hidden states of the LSTM layers using language modeling and fine-tuning it for a supervised task using labeled data. They concluded that using this pretrained weights helped stabilize the learning in LSTM recurrent networks. In [25], the authors used the final hidden state of pretrained stacked LSTM language model as a contextual embedding for a word. They reasoned that this method generated better embeddings as it captured the semantic and syntactic role of words, keeping in mind the context. They augmented the LM embeddings to their sequence tagging architecture and achieved 91.93 % F1 for the CoNLL 2003 Name Entity Recognition task and 96.37 % F1 for the CoNLL 2000 Chunking task. The authors in [26] used a similar approach but instead of language modelling they used machine translation task for pre-training.

Unlike the previous works that used top layers of pretrained language models, Peters et al. [27] use weighted sum of internal states present at every layer in their model. The weights being learnt while fine-tuning the embeddings for a particular task. It enabled the downstream task to give more weight to the hidden representation that captures the semi-supervision signal most useful to it. This constructed richer representation embeddings called the ELMo embeddings, which improved the state of the art across six diverse NLP tasks. Universal Language Model Fine-tuning (ULMFiT) introduced by Ruder et al. [29] describes a fine-tuning method that uses techniques like discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing to enable faster transfer learning of pretrained language models to other NLP tasks.

Further, the transformer architecture introduced in [28] by Vaswani et al., improved the state of the art result for machine translation. They used the attention mechanism that takes into consideration the interaction between time steps and the whole sequence before making the predictions. This way it captures the time dependency better than LSTMs. Researchers started using transformer or self-attention for other NLP tasks and it gave promising results [30], [31].

In [14], the authors combined the ideas of using pre-trained neural network weights [24], context-sensitive features [27], slanted triangular learning rates [29] and transformer architecture with attention mechanism [28], and came up with the language representation model BERT. BERT achieved the state of the art for eleven NLP tasks. It is better than other language representation models [27], [32] because it learns deep bidirectional representations by taking into account both left and right context in every neural layer. It achieves this by changing the language modeling objective. Instead of the normal objective of predicting the next word, it masks some of the words randomly in the input and uses the model conditioned on the left and right context to predict the masked word. The model also jointly trains on the binarized next sentence prediction task, i.e., given two sentences A and B as the input the model trains to predict if sentence B should succeed A or not. The architecture of the model is composed of multi-layer encoder from the transformer [28]. BERT uses the encoder part while the OpenAI GPT [32] uses the decoder part with masked tokens, and hence is conditioned only on the left context. Ablation studies presented in [14] show that the bidirectional aspect of the BERT is what makes it perform better than OpenAI GPT.

III. Model Architecture

In this work, we propose a two layered architecture. The first layer is the pretrained BERT model that produces the BERT embeddings. It is followed by a hybrid biLSTM-CRF layer that finally outputs the predicted labels. The overall architecture is depicted in Figure 1.

A. BERT embedding

BERT model contains multiple encoding layers with pre-trained weights, of the transformer [14] stacked together. The pre-trained weights are learnt by training it on masked word prediction and binarized next sentence prediction, language modelling tasks. Each encoder layer comprises of multi-head self attention and feed forward networks. Attention layer performs the mapping of key-value pair and a query to an output. The output is calculated based on a weighted sum of values where the weights are calculated using alignment score between keys and queries. The attention used in [28] is the

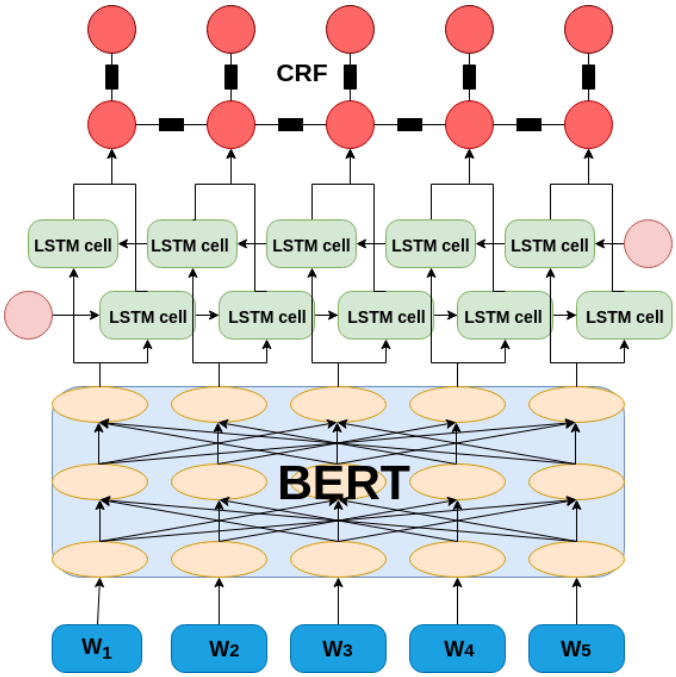

Fig. 1. Proposed Model Architecture

Scaled Dot-Product Attention which is calculated as:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}}{\sqrt{d_k}})\boldsymbol{V} \quad (1)$$

where **Q**, **K** and **V** are the query, key and value matrices, respectively and d_k is the dimension of key vectors. In case of self attention, the query, key and value are equal to the output of the previous encoder layer.

Multi head attention considers multiple representation sub spaces in parallel. This is done by linearly projecting keys, values and queries by different learnt matrices and performing attention on each one of them. The resultant matrices are finally concatenated and again projected. The equations for multi head attention can be summarized as:

$$Multihead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(\boldsymbol{head_1}, ..., \boldsymbol{head_h})\boldsymbol{W^O} \quad (2)$$

$$\boldsymbol{head_i} = Attention(\boldsymbol{Q}\boldsymbol{W_i^Q}, \boldsymbol{K}\boldsymbol{W_i^K}, \boldsymbol{V}\boldsymbol{W_i^V}) \quad (3)$$

where $\boldsymbol{W_i^Q}, \boldsymbol{W_i^K}, \boldsymbol{W_i^V} \in \mathbb{R}^{d_{inp} \times d_{out}}$ and $\boldsymbol{W^O} \in \mathbb{R}^{hd_{out} \times d_{inp}}$, here d_{inp} is the initial dimension of key, value and query, d_{out} is the projected dimension and h is number of heads. After that, the result is passed through a feed forward network to get the final output for the encoder layer. An additional point to note is that the multi-head and feed forward networks have residual connections around them, and are always followed by layer-normalization.

In our task, the weights of the encoder layers are also updated along with the other layers during the back-propagation, this fine-tunes BERT embeddings for our task.

B. BiLSTM layer

Recurrent neural nets that allow cyclic connections in neural nets have been used widely in the NLP domain. These cyclic connections change the state of the RNN cell recursively and helps them to learn features from entire past sequences. However, a simple recurrent unit is not able to capture longer contexts due to the vanishing gradient problem. LSTM introduce in [33] alleviates the problem by having additional gates and cell state to control the flow of information. The basic equations governing the output of a basic LSTM cell are as follows:

$$\boldsymbol{f_t} = \sigma(\boldsymbol{W_f}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_f) \quad (4)$$

$$\boldsymbol{i_t} = \sigma(\boldsymbol{W_i}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_i) \quad (5)$$

$$\boldsymbol{c_t^{'}} = tanh(\boldsymbol{W_c}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_c) \quad (6)$$

$$\boldsymbol{c_t} = \boldsymbol{f_t} * \boldsymbol{c_{t-1}} + \boldsymbol{i_t} * \boldsymbol{c_t^{'}} \quad (7)$$

$$\boldsymbol{o_t} = \sigma(\boldsymbol{W_o}[\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + b_o) \quad (8)$$

$$\boldsymbol{h_t} = \boldsymbol{h_t} * tanh(\boldsymbol{o_t}) \quad (9)$$

In our work, we use a bidirectional LSTM layer comprising of this basic LSTM cell on top of the BERT embedding. It models the sequential dependency further by processing the sequence word by word. The forward LSTM process the sequence from left to right while the backward LSTM processes it in the opposite direction. This makes the prediction more accurate by conditioning it on both past and future contexts.

C. CRF Classifier

CRFs have already been successfully used for sequence labeling tasks in NLP. The hybrid LSTM-CRF model is shown to achieve high performance for name entity recognition, POS tagging and chunking tasks [34] [35].

CRF is a discriminative model, i.e, it tries to learn the conditional probability of output given the input rather than the joint probability. This is helpful because we can ignore the structure and relationships between the input variables, entirely. Along with that, it is composed of feature functions that can take into account overlapping features.

Our model uses linear chain CRF, which is the most commonly used version of CRF. Linear chain CRF has a transition matrix that captures previous and current step dependencies for making the tagging decision unlike the cross entropy classifier that makes independent decision for each label. The loss of CRF layer is just the negative log likelihood of the probability of the correct sequence of labels and we use the Viterbi decoding for finding the correct punctuation labels at the prediction time.

IV. Results and Experiment

A. Dataset

We evaluate our model on the monolingual English punctuation restoration task and use the IWSLT2012[2] dataset for our experiment. It consists of TED talks transcribed in several

[2]http://hltc.cs.ust.hk/iwslt/index.php/evaluation-campaign/ted-task.html

TABLE I
RESULTS ON IWSLT DATASET

Model	Period			Comma			Question mark			Overall		
	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
T-BRNN	72.3	71.5	71.9	64.4	45.2	53.1	67.5	58.7	62.8	68.9	58.1	63.1
T-BRNN-pre	73.3	72.5	72.9	65.5	47.1	54.8	70.7	62.8	66.7	70.0	59.7	64.4
Word CNN	75.8	95.5	84.6	56.1	60.4	58.2	40.8	55.7	47.1	65.8	77.7	71.3
SAPR	**96.7**	**97.3**	**96.8**	57.2	50.8	55.9	70.6	69.2	70.3	78.2	74.4	77.4
Bert-Punct$_{BASE}$	82.6	83.5	83.1	**72.1**	72.4	72.3	77.4	89.1	82.8	77.4	81.7	79.4
Bert-Punct$_{LARGE}$	84.9	83.3	84.1	70.8	**74.3**	**72.5**	**82.7**	**93.5**	**87.8**	**79.5**	**83.7**	**81.4**

languages that are normally used for evaluating Automatic Speech Recognition, Speech Language Translation and Machine Translation tasks.

We take the data from the machine translation track and use the same training, development and test sets, as the previous works. After preprocessing, we get 2.1M and 290k tokens in the train and dev sets, respectively.

B. Preprocessing

To fine-tune the pre-trained BERT language representation model, the input is preprocessed in a certain way. The preprocessing of an input example along with the labels is shown in Figure 2. As can be seen every input example is prefixed with a [CLS] token. The hidden state corresponding to this token learns the encoded representation for the whole input sentence. Tokens are converted to WordPiece embeddings that capture sub-level word information and generalize well to rare tokens. Segment embeddings and [SEP] token is used to identify different parts of an input example. Since our input example has just one part we use zero vector for segment embedding and only append [SEP] at the end of every example.

For punctuation prediction, one input example is typically a paragraph and the sequence length for our model is fixed. So we split our examples into two or more sequences if they are more than this length, ensuring that each input sequence starts with a new sentence i.e., after end sentence punctuation labels. For examples less than the sequence length we pad them with the [PAD] token. We maintain an input mask to ensure that the padded tokens are not accounted in the loss.

Our model predicts only comma, question mark and period. Other end sentence punctuation symbols that exist in text are mapped to period. Words that don't have punctuation following them are given a different class. In addition we have two more classes, one for WordPiece tokens that are part of single word and the other for [PAD] tokens.

The segment, WordPiece and positional embeddings are added together to form the input for the pre-trained BERT model. We don't want our model to learn to predict sentence end punctuation classes using capitalized word cues, hence we use the pre-trained uncased version of BERT.

C. Experimental Setup

For training, we use Adam optimizer with learning rate of 2e-5, beta_1 = 0.9, beta_2 = 0.999 and the weight decay rate of 0.01. Initially we do a learning rate warm-up for 0.1 percent of the training steps and then use exponential decay with decay

```
[CLS] oh    god    i know this    it  '  s louis mac ##ne
 PAD  COMMA PERIOD O  O   PERIOD  O   O  O   O   X    X

##ice    yeah    it  '  s  bag ##pipe music  [SEP]
PERIOD   PERIOD  O   O  O   X     O    PERIOD  PAD
```

Fig. 2. Pre-processed example for BERT model

steps set to 2000 and decay rate set to 0.9. We perform gradient clipping with threshold set to 2. The model is trained for 5 epochs with batch size of each training step equal to 8. The dimension of the BERT embedding is 768 for the base version and 1024 for the large version. The dimension of the hidden state of BiLSTM is set to 200. The number of output classes is 6 and the sequence length is set to 128. We use a dropout of 0.1 for all the layers.

D. Baseline Models

We compare our model with the previous works that achieve good performance on the task of punctuation restoration.

- *Word CNN* is a character based model that uses convolutions over character embeddings to extract features and finally labels them using a CRF classifier [36].
- *T-BRNN* uses a bidirectional LSTM with late-fused attention mechansim.
- *T-BRNN-pre* is the same model using pre-trained word embeddings [20].
- *SAPR* treats the task as a machine translation task. It uses the transformer architecture with the decoder having two output labels [23].

E. Results

We refer to our model as *Bert-Punct*. We have two versions of the model, base and large corresponding to the two pre-trained BERT embeddings. Similar to the other models, we predict the three most common types of punctuation symbols: period, comma and question mark. In addition to that our model is trained on examples consisting of multiple sentences, hence it learns to predict period and other end punctuation classes solely based on syntactic and semantic cues.

For each of the model, we report the overall as well as class-wise precision, recall and F1-measure metrics.

As can be seen from Table I, our both models significantly outperform all the previous models on the IWSLT2012 dataset. For the base version, we achieve an overall improvement of 2% in F1 as compared to the previous best model, SAPR. While for the large, we improve further, 4% over SAPR. This

is due to the high recall capacity of our model. The recall values being 7.3% and 9.3% above the previous best model, for the base and large version, respectively. The recall values for the minority classes are also high, this suggests that our architecture is also able to handle the imbalance of classes.

While looking at individual classes, we can see that the SAPR performs better on predicting period than *BERT-Punct*. However, both of our models significantly outperform SAPR on other classes.

The high overall F1 can be mainly attributed to our model's ability to distinguish and recall commas better than other models, while not sacrificing the performance for other classes. There is an absolute improvement of 14.9%, 21.6% and 16.4% in precision, recall and F1 values for comma, in our base version with respect to SAPR. For the large version, it is 13.6%, 23.5% and 16.6%. Also, our model's ability to predict question mark is far better than SAPR. Further, *BERT-Punct* large version, has higher values for periods and question marks but has a lesser value of precision for the comma punctuation class than the base version.

Notice that the SAPR and Bert-Punct both use the transformer architecture with modifications and achieve better results. This suggests that self-attention architectures are better than recurrent architectures for the task of punctuation prediction.

V. Conclusion and Future Work

In this paper, we proposed a sequence labelling architecture that uses the pretrained language representation model BERT. We fine tuned BERT for our task and with small amount of training achieved state of the art results. Experiments performed show that our model is significantly better at predicting the less common punctuation classes than the other models. We added a simple hybrid of LSTM-CRF on top of the pretrained model and achieved an overall F1 of 81.4% (absolute improvement of 4%).

For the future work, we will modify the model to include prosody features. Along with that, we will train it on other languages and multilingual text. This would help us check the robustness of our model. We would also like to explore the task as a machine translation task that uses pre-trained transformer model.

Acknowledgement

This research is supported by TL@NTU, Nanyang Technological University, Singapore

References

[1] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim, B. Roomi, and P. Hall, "English conversational telephone speech recognition by humans and machines," in *INTERSPEECH*, 2017.

[2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *CoRR*, vol. abs/1610.05256, 2016.

[3] D. A. Jones, F. Wolf, E. Gibson, E. Williams, E. Fedorenko, D. A. Reynolds, and M. A. Zissman, "Measuring the readability of automatic speech-to-text transcripts," in *INTERSPEECH*, 2003.

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

[5] J. Makhoul, A. Baron, I. Bulyko, L. H. Nguyen, L. A. Ramshaw, D. Stallard, R. M. Schwartz, and B. Xiang, "The effects of speech recognition and punctuation on information extraction performance," in *INTERSPEECH*, 2005.

[6] A. Stolcke, E. Shriberg, R. A. Bates, M. Ostendorf, D. Z. Hakkani-Tür, M. Plauché, G. Tür, and Y. Lu, "Automatic detection of sentence boundaries and disfluencies based on recognized words," in *ICSLP*, 1998.

[7] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. P. Harper, "Enriching speech recognition with automatic detection of sentence boundaries and disfluencies," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1526–1540, 2006.

[8] E. Shriberg, A. Stolcke, D. Z. Hakkani-Tür, and G. Tür, "Prosody-based automatic segmentation of speech into sentences and topics," *Speech Communication*, vol. 32, pp. 127–154, 2000.

[9] J. D. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

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

[11] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *LREC*, 2016.

[12] P. Zelasko, P. Szymanski, J. Mizgajski, A. Szymczak, Y. Carmiel, and N. Dehak, "Punctuation prediction model for conversational speech," in *Interspeech*, 2018.

[13] A. Adami, R. Mihaescu, D. A. Reynolds, and J. J. Godfrey, "Modeling prosodic dynamics for speaker recognition," in *ICASSP*, 2003.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

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

[16] D. Zhang, S. Wu, N. Yang, and M. Li, "Punctuation prediction with transition-based parsing," in *ACL*, 2013.

[17] M. Ballesteros and L. Wanner, "A neural network architecture for multilingual punctuation generation," in *EMNLP*, 2016.

[18] E. Cho, J. Niehues, and A. H. Waibel, "Segmentation and punctuation prediction in speech language translation using a monolingual translation system," in *IWSLT*, 2012.

[19] J. Driesen, A. Birch, S. Grimsey, S. Safarfashandi, J. Gauthier, M. Simpson, and S. Renals, "Automated production of true-cased punctuated subtitles for weather and news broadcasts," in *INTERSPEECH*, 2014.

[20] O. Tilk and T. Alumäe, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration," in *INTERSPEECH*, 2016.

[21] K. Xu, L. Xie, and K. Yao, "Investigating lstm for punctuation prediction," *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 1–5, 2016.

[22] D.-C. Can, T.-N. Ho, and E.-S. Chng, "A hybrid deep learning architecture for sentence unit detection," *2018 International Conference on Asian Language Processing (IALP)*, pp. 129–132, 2018.

[23] F. Wang, W. Chen, Z. Yang, and B. Xu, "Self-attention based network for punctuation restoration," *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2803–2808, 2018.

[24] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *NIPS*, 2015.

[25] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *ACL*, 2017.

[26] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *NIPS*, 2017.

[27] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. S. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[29] S. Ruder and J. Howard, "Universal language model fine-tuning for text classification," in *ACL*, 2018.

[30] N. Kitaev and D. Klein, "Constituency parsing with a self-attentive encoder," in *ACL*, 2018.

[31] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *CoRR*, vol. abs/1801.10198, 2018.
[32] A. Radford, "Improving language understanding by generative pre-training," 2018.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
[34] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015.
[35] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *HLT-NAACL*, 2016.
[36] C. Wang and B. Xu, "Convolutional neural network with word embeddings for chinese word segmentation," in *IJCNLP*, 2017.