End-to-end Speech-to-Punctuated-Text Recognition

Jumon Nozaki\textsuperscript{1}, Tatsuya Kawahara\textsuperscript{1}, Kenkichi Ishizuka\textsuperscript{2}, Taiichi Hashimoto\textsuperscript{2}

\textsuperscript{1} Graduate School of Informatics, Kyoto University, Japan
\textsuperscript{2} RevComm, Inc., Japan

\{nozaki, kawahara\}@sap.ist.i.kyoto-u.ac.jp
\{ishizuka, taiichi.hashimoto\}@revcomm.co.jp

\begin{abstract}
Conventional automatic speech recognition systems do not produce punctuation marks which are important for the readability of the speech recognition results. They are also needed for subsequent natural language processing tasks such as machine translation. There have been a lot of works on punctuation prediction models that insert punctuation marks into speech recognition results as post-processing. However, these studies do not utilize acoustic information for punctuation prediction and are directly affected by speech recognition errors. In this study, we propose an end-to-end model that takes speech as input and outputs punctuated texts. This model is expected to predict punctuation robustly against speech recognition errors while using acoustic information. We also propose to incorporate an auxiliary loss to train the model using the output of the intermediate layer and unpunctuated texts. Through experiments, we compare the performance of the proposed model to that of a cascaded system. The proposed model achieves higher punctuation prediction accuracy than the cascaded system without sacrificing the speech recognition error rate. It is also demonstrated that the multi-task learning using the intermediate output against the unpunctuated text is effective. Moreover, the proposed model has only about 1/7th of the parameters compared to the cascaded system.
\end{abstract}

\textbf{Index Terms:} speech recognition, punctuation prediction, connectionist temporal classification, transformer

\section{1. Introduction}

Automatic speech recognition (ASR) systems have made significant progress with advances in deep neural networks (DNNs) \cite{1, 2, 3}. ASR systems are widely used in many applications, such as conversational robots and speech captioning systems. However, conventional ASR systems do not generate punctuation marks, which affect the readability of the transcripts. Also, unpunctuated texts are not suitable for subsequent applications of natural language processing. For example, some previous work has shown that unpunctuated texts degrade the performance of machine translation and named entity recognition \cite{4, 5, 6}.

To address this problem, research on punctuation prediction models has been conducted. The punctuation prediction model is used to insert punctuation marks into speech recognition results as post-processing. Before DNNs became widely used, n-gram language models \cite{7}, support vector machines \cite{8}, conditional random fields \cite{9, 10} were mainly used to predict punctuation. After that, using DNN models such as LSTM \cite{11} and Transformer \cite{12} were investigated to output punctuated sentences from unpunctuated word sequences as input \cite{13, 14}. More recently, there have been a lot of works on using pre-trained models based on the Transformer architecture such as BERT \cite{15} for punctuation prediction \cite{16, 17}. Thanks to the powerful Transformer architecture pre-trained with a huge amount of text corpora, they achieved state-of-the-art performance on the IWSLT2011 dataset \cite{18}, a well-known benchmark for punctuation prediction. Then, the focus of research has shifted towards using more advanced pre-trained models such as RoBERTa \cite{19} and ELECTRA \cite{20} and more advanced training techniques to further push the performance of punctuation prediction \cite{21, 22}.

These conventional studies implicitly assume a cascaded application of two separate models: an ASR model and a punctuation prediction model. However, there are some disadvantages caused by the nature of the cascaded system. First, these previous studies generally use only lexical features and not acoustic (prosodic) features although acoustic information such as pauses and pitches are considered to be important for punctuation prediction. Ignoring acoustic information for punctuation prediction also makes the system vulnerable to speech recognition errors. Second, recent studies use pre-trained models such as BERT, but they have a large number of parameters and are not suitable for on-device systems. Because much attention has been given to running ASR systems on mobile devices such as smartphones and tablets \cite{23}, it is important to develop a fast and lightweight model that can run on limited computational resources. Lastly, there is accuracy degradation due to segmentation errors. When feeding an ASR output into the punctuation model based on a pre-trained model such as BERT, it needs to be tokenized according to the vocabulary of the pre-trained model, but tokenizing texts without punctuation and with ASR errors is difficult, thus causing some tokenization errors. One might be tempted to use the same vocabulary for the ASR model and the pre-trained model to make tokenization unnecessary, but the vocabulary of the pre-trained model is not suitable for the vocabulary of ASR because it is case-sensitive and contains a lot of unpronounceable symbols such as parentheses. This issue of the segmentation error is more serious for languages without explicit word boundaries (e.g., Japanese and Chinese).

In this study, we propose an end-to-end model for speech-to-punctuated-text recognition. Specifically, we use the stacked Transformer encoder layers and train them with CTC loss \cite{24} using speech as input and punctuated texts as output. This model predicts punctuation robustly against ASR errors and segmentation errors while using acoustic information. We also propose a method to train the model using an auxiliary loss calculated from the output of the intermediate layer and unpunctuated texts, in addition to the original loss in the last layer. The experiments are conducted on English and Japanese datasets to demonstrate the effectiveness of the proposed model compared to the cascaded system.
2. Related Work

2.1. Punctuation prediction using acoustic features

There are studies that use acoustic features to predict punctuation [25, 26]. In these studies, it is assumed that there is a separately trained ASR model that outputs unpunctuated texts. For each token in the ASR output, the corresponding acoustic features are obtained, which are used as input to train a punctuation prediction model. While they still use separate models for ASR and punctuation prediction, we propose an end-to-end model from speech to punctuated text, which is optimized as a single model.

2.2. Auxiliary loss in intermediate layers

Some studies investigated using an auxiliary loss for the output of intermediate layers to train CTC-based ASR [27, 28, 29]. They proposed to use phones [27], a small-sized vocabulary [28], or even the same label as that for the last layer [29] for the output of the intermediate layer. Our method can be seen as an extension of these studies to speech-to-punctuated-text recognition, in that we use unpunctuated texts for the output of the middle layer and punctuated texts for the output of the last layer.

2.3. End-to-end approach

Some previous work proposed to train an end-to-end model that takes speech as input and outputs “formatted” texts. Casco et al. [30] proposed a system that takes speech as input and outputs a case-sensitive text directly in an end-to-end manner. The most similar work to ours is that of Mimura et al. [31]. They proposed an end-to-end model that takes speech as input and outputs a clean text with punctuation and without disfluency for the Japanese parliamentary meetings. While they addressed inserting punctuation, removing fillers, and substituting colloquial expressions at the same time, we focus on punctuation and “speech-to-punctuated-text recognition, in that we use unpunctuated texts for the output of the middle layer and punctuated texts for the output of the last layer.

3. Speech-to-Punctuated-Text Recognition

3.1. Task Definition

In this study, we define speech-to-punctuated-text recognition as the problem of outputting a token sequence containing punctuation marks $y_{punct}$ from an acoustic feature sequence $X$ as input. For punctuation marks in English, we consider a comma (,), a period (.), and a question mark (?).

3.2. Baseline Model

The mainstream method for speech-to-punctuated-text recognition is to train a speech recognition model and a punctuation prediction model separately, and then cascade them together for inference. An overview of the cascaded system is shown on the left-hand side of Figure 1. In this study, we use a Transformer-based ASR model and BERT-based punctuation prediction model as a baseline system. Specifically, the ASR output is first tokenized according to the vocabulary of BERT. Then, BERT is fine-tuned with a task to classify each token according to the type of punctuation marks inserted immediately after the token. In the case of English data, each token is assigned to the following classes: “O”, “COMMA”, “PERIOD”, and “QUESTION”, where “O” indicates there is no punctuation after the token. For example, suppose that the input sequence is “yes it is”, the corresponding label sequence is “COMMA, O, PERIOD” since the properly punctuated version of the input text is “yes, it is.” Using the classification task into these punctuation classes, BERT is fine-tuned with the following cross-entropy loss.

$$L_{CR} = - \sum_{k=1}^{K} t_k \log p_k$$ (1)

where $K$ is the number of classes (in this case $K = 4$), $p_k$ is the predicted probability for label $k$, and $t_k$ is the target probability for label $k$. We use the one-hot label for $t_k$: $t_k = 1$ if $k$ is the corresponding label, else $t_k = 0$.

4. Proposed Method

In this study, we propose a model that directly outputs a token sequence containing punctuation marks $y_{punct}$ from an acoustic feature sequence $X$ in an end-to-end manner. An overview of the proposed model is shown on the right-hand side of Figure 1.

We use stacked Transformer Encoder layers as a model architecture and train it with a CTC loss function. The CTC loss function calculates the sum of the probabilities of the alignments which can be reduced to the output label series $y$, as represented by the following equation.

$$P_{CTC}(y \mid X) = \sum_{a \in \Gamma(y)} P(a \mid X)$$ (2)

Here, $\Gamma^{-1}(a)$ is a function that concatenates consecutive identical tokens and removes special blank tokens. The alignment probability $P(a \mid X)$ is formulated under the conditional independence assumption between tokens:

$$P(a \mid X) = \prod_{s} P(a_s \mid X)$$ (3)

where $a_s$ represents the $s$-th symbol of $a$, and $P(a_s \mid X)$ represents the probability of observing $a_s$ at time $s$.

The proposed model uses an $l$-layer Transformer encoder and is trained to minimize the following CTC loss function for the output of the final $l$-th layer $X_l$. 

---

Figure 1: Overview of the cascaded system (left) and the proposed end-to-end model (right) for speech-to-punctuated-text recognition.
\[ \mathcal{L}_{\text{CTC}} = -\log P_{\text{CTC}}(y_{\text{unpct}} \mid X_l) \] (4)

Although Eq. (4) is sufficient to train the model in an end-to-end manner, we also propose to use an auxiliary CTC loss calculated with the token sequence without punctuation \(y_{\text{unpct}}\) and the output of the intermediate layer to more effectively train the model. The loss for the output of the middle \([l/2]\)-th layer \(X_{[l/2]}\) is calculated as follows.

\[ \mathcal{L}_{\text{inter}} = -\log P_{\text{CTC}}(y_{\text{unpct}} \mid X_{[l/2]}) \] (5)

The final loss function is formulated as a weighted linear sum of the Eq. (4) and Eq. (5).

\[ \mathcal{L}_{\text{total}} = \lambda_{\text{CTC}} \mathcal{L}_{\text{CTC}} + \lambda_{\text{inter}} \mathcal{L}_{\text{inter}} \] (6)

where \(\lambda_{\text{CTC}}\) and \(\lambda_{\text{inter}}\) are the coefficients of the respective loss terms.

During inference, we do not make the predictions in the middle layer, but only the predictions in the final layer, thus there is no overhead in inference time.

5. Experimental Evaluations

5.1. Datasets

We used two datasets of different languages: English and Japanese.

The MuST-C corpus [32] was used as the English dataset. Although MuST-C is a dataset mainly used for research on speech translation, in this study, we extracted the English speech and the English script with punctuation from the English-German speech translation data and used them as paired data. Comma (,), period (.), and a question mark (?) are considered to be punctuation marks. We used the “tst-COMMON” set as a test set. As a preprocessing, the scripts were all converted to lowercase. Note that in the MuST-C corpus, periods and question marks are mostly at the end of an utterance, so the prediction of these marks is relatively easy.

JCALL is an in-house Japanese dataset, which consists of audio recordings of conversations between a salesperson and a customer in inside sales and those between an operator and a customer in call centers. We considered the Japanese comma (、), the Japanese full-stop mark (。), and a question mark (?) as punctuation marks. Utterances were segmented using VAD, and punctuation marks are present in the middle of an utterance as well as at the end of an utterance.

For both datasets, we removed utterances longer than 30 seconds from the training set due to computational resource constraints. We also removed some duplicate utterances from the training set so that the number of utterances with the same reference text was at most 300. The statistics of the two datasets are shown in Table 1.

5.2. Experimental Setup

For the baseline cascaded system, we trained an ASR model and a punctuation prediction model separately. The ASR model consisted of stacked Transformer encoder layers and was trained with the CTC loss. During training, speech and unpunctuated texts were used as paired data. The number of Transformer layers was 12, the dimension of the hidden layer was 256, and the number of heads was 4. For input features, 80-dimensional log-mel spectrum features were used, and SpecAugment [33] was used for data augmentation. For the vocabulary of ASR, we used 2000 tokens created by SentencePiece [34] for MuST-C, and 1,923 characters for JCALL.

For the punctuation prediction model, we used the pre-trained base-sized BERT model taken from Hugging Face’s transformers package [35]. We added a linear layer to the final layer of the BERT model and fine-tuned it with the punctuation classification task. The transcripts of the training set of MuST-C and JCALL were used as the fine-tuning data of BERT. The training was done for 10 epochs on a single GPU using Adam [36] as the optimizer with a learning rate of \(10^{-5}\).

The batch size was set to 32.

The proposed end-to-end model was trained with the same architecture and the training method as the ASR model of the cascaded system, except that punctuated texts instead of unpunctuated texts were used as labels for training.

For the proposed model and the ASR model of the cascaded system, we conducted training with and without using the auxiliary loss represented by Eq. (5). When using the auxiliary loss, \(\lambda_{\text{CTC}}\) and \(\lambda_{\text{inter}}\) in Eq. (6) were equally set to 0.5.

5.3. Evaluation

As a measure of the accuracy of ASR, we used Character Error Rate (CER) for JCALL and Word Error Rate (WER) for MuST-C. All punctuation marks in the output texts were removed before the calculation. As a measure of the accuracy of the punctuation prediction, the F1 score for each type of punctuation mark and its average were calculated. As the output of the model contains ASR errors, it is not possible to simply calculate the F1 score by comparing it to the ground-truth. Therefore, we first aligned the predicted text with the ground-truth text, then calculated the F1 score. In addition, we compared the total number of parameters in each model.

5.4. Results

The results of the experiments are shown in Table 2.

In the experiment on MuST-C, when we did not use the intermediate loss, the WER of the proposed model was largely worse than that of the cascaded system. However, when the intermediate loss was incorporated, the WER was improved to a comparable level to the cascaded system, showing the effectiveness of using the intermediate loss. We conjecture that simply training the mapping from speech to punctuated texts was difficult.

Table 1: The number of utterances and punctuation marks for each split of MuST-C and JCALL datasets. “、” and “。” denote the Japanese comma and the Japanese period, respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Split</th>
<th>#Utterances</th>
<th>#Punctuation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(、)</td>
<td>(。)</td>
</tr>
<tr>
<td>MuST-C</td>
<td>dev</td>
<td>1,423</td>
<td>2,117</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>2,641</td>
<td>2,761</td>
</tr>
<tr>
<td>JCALL</td>
<td>dev</td>
<td>4,000</td>
<td>4,247</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>4,000</td>
<td>4,043</td>
</tr>
</tbody>
</table>

1Publicly available at https://huggingface.co/. We used bert-base-uncased for MuST-C and cl-tohoku/bert-base-japanese-whole-word-masking for JCALL.
Table 2: Evaluation results on MuST-C and JCALL datasets.

<table>
<thead>
<tr>
<th>Model</th>
<th>MuST-C</th>
<th>JCALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WER (%)</td>
<td>Punctuation F1-Score (%)</td>
</tr>
<tr>
<td>Cascaded System</td>
<td></td>
<td>avg</td>
</tr>
<tr>
<td>+ intermediate loss</td>
<td>21.2</td>
<td>63.3</td>
</tr>
<tr>
<td>End-to-End (Proposed)</td>
<td>19.8</td>
<td>61.1</td>
</tr>
</tbody>
</table>

Table 3: Word error rates (WER) and averaged punctuation F1 scores when different labels are used for the last layer and the middle layer on MuST-C. "-" indicates that intermediate loss was not used. The last row shows the result of multitask learning for the output of the last layer. “E2E” stands for end-to-end.

<table>
<thead>
<tr>
<th>Last layer</th>
<th>Middle layer</th>
<th>E2E?</th>
<th>WER (%)</th>
<th>Punctuation F1-Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y_{pnct}</td>
<td>-</td>
<td>✓</td>
<td>27.4</td>
<td>68.0</td>
</tr>
<tr>
<td>Y_{unpnct}</td>
<td>-</td>
<td>✓</td>
<td>21.2</td>
<td>74.1</td>
</tr>
<tr>
<td>Y_{pnct}</td>
<td>Y_{unpnct}</td>
<td>✓</td>
<td>20.0</td>
<td>74.4</td>
</tr>
<tr>
<td>Y_{pnct}</td>
<td>Y_{unpnct}</td>
<td>✓</td>
<td>25.2</td>
<td>72.4</td>
</tr>
<tr>
<td>Y_{pnct}</td>
<td>Y_{unpnct}</td>
<td>✓</td>
<td>19.8</td>
<td>75.9</td>
</tr>
<tr>
<td>Y_{pnct} &amp; Y_{unpnct}</td>
<td>-</td>
<td>✓</td>
<td>20.4</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Table 3 shows the result of the experiments. We can see that using punctuated texts for the last layer and unpunctuated texts for the middle layer achieved the best WER (19.8%) and F1 punctuation score (75.9%). Training the model with punctuated texts for both of the last layer and the middle layer significantly degraded WER (25.2%), which confirms using the unpunctuated texts for the middle layer was essential. We conjecture that this was because gradually making the task difficult as going to the deeper layer made the training stable. When we trained a model using Y_{pnct} and Y_{unpnct} in the last layer, the accuracy of punctuation prediction was drastically degraded because it was not effectively trained.

6. Conclusions

In this study, we proposed an end-to-end model that directly predicts a punctuated text using speech as input. The proposed model can utilize acoustic information for punctuation prediction and can be robust against ASR errors and segmentation errors. The evaluation experiments showed that the proposed model can achieve higher recognition accuracy with much fewer parameters than the conventional cascaded system. In addition, we showed the effectiveness of using an auxiliary loss using unpunctuated texts for the output of the intermediate layer. Our approach also has advantages in terms of inference speed and the simplicity of its architecture. Our study sets out a future research direction of using an end-to-end model for speech-to-punctuated-text recognition. In the future, we plan to study the improvement of the proposed model using different architectures and its application to real-time speech recognition.

7. References


