Automatic Insertion of Commas and Linefeeds into Lecture Transcripts based on Multi-Task Learning

Zhicheng Fang
Graduate School of Informatics
Nagoya University
fang.zhicheng.d4@s.mail.nagoya-u.ac.jp

Masaki Murata
National Institute of Technology, Toyota College
murata@toyota-ct.ac.jp

Shigeki Matsubara
Information Technology Center
Graduate School of Informatics
Nagoya University
matubara@nagoya-u.jp

Abstract

To support audience understanding in lecture halls, it is effective to transcribe speech into text automatically and present it using subtitles. However, lectures tend to have long sentences, and subtitles without clear word boundaries can be difficult to read. Thus, this paper proposes a method to insert commas and linefeeds into lecture text simultaneously to generate easy-to-read subtitles. The proposed method involves using data with inserted commas and linefeeds in lecture text and fine-tuning a pretrained BERT model through multi-task learning for the comma insertion and linefeed insertion tasks. Experiments conducted on Japanese spoken language data demonstrated that this method obtained higher performance compared to sequentially inserting commas and linefeeds, thereby confirming the effectiveness of the proposed method.

1 Introduction

Speech recognition technology can be used to transcribe speech from lectures into text automatically and present it as subtitles to assist the audience in understanding the lecture content. However, to generate readable subtitles, it is necessary to accurately transcribe the speech into text and consider how the transcribed text is presented to the audience. In the lecture context, sentences tend to be long, and when the entire sentence spans multiple lines on the screen, it can become difficult to discern word breaks, which can result in subtitles that are difficult to read (Murata et al., 2010). Thus, it is desirable to insert appropriate commas and linefeeds in lecture subtitles to enhance readability and comprehension.

Previous studies have addressed comma (Murata et al., 2010; Tilk and Alumäe, 2016) and linefeed insertion (Ohno et al., 2009) in text separately, and methods have been proposed to format text by inserting commas and linefeeds sequentially(Murata et al., 2011). However, the appropriate positions for commas and linefeeds may influence each other; thus, it is preferable to insert commas and linefeeds interactively.

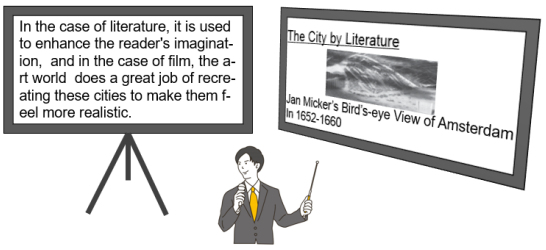

Figure 1: Subtitle presentation environment for lectures

Therefore, in this paper, we propose a method to generate readable subtitles by inserting commas and linefeeds simultaneously into transcribed Japanese lecture speech. This study assumes the installation of a display that shows subtitles on multiple lines as a way to provide subtitle information to audiences in lecture halls. In the proposed method, the comma insertion and linefeed insertion tasks are achieved by multi-task learning. Here, a pretrained BERT (Kenton and Toutanova, 2019) model is utilized with additional layers to determine the comma and linefeed insertion positions using binary classification followed by a fine-tuning process.

An experiment conducted using Japanese spoken language data demonstrated an F-measure of 78.17% for comma insertion and 75.52% for linefeed insertion, which represents performance improvements compared to the baseline methods and confirms the effectiveness of the proposed method.

The remainder of this paper is organized as follows: In Section 2, we provide an overview of the target task, i.e., comma and linefeed insertion in subtitle text, and related research. In Section 3, we explain the method proposed to realize simultaneous comma and linefeed insertion. In Section 4, we report on an experiment conducted using Japanese spoken language data, and in Section 5, we discuss

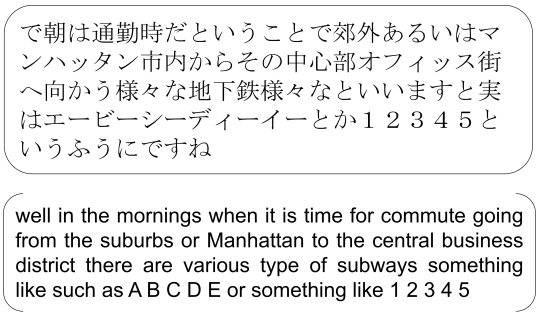

Figure 2: Transcribed text of the speech

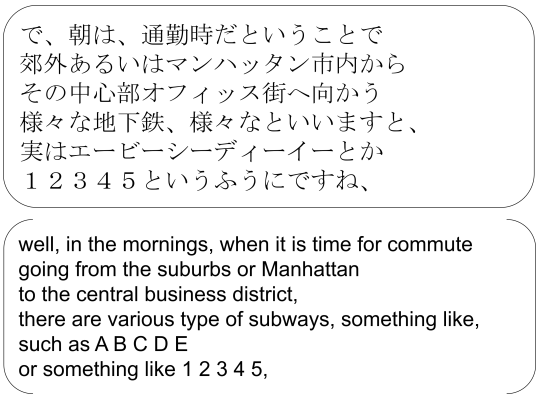

Figure 3: Text with commas and linefeeds inserted at appropriate positions

the experimental results. Finally, the paper is concluded in Section 6, including a brief discussion of future challenges.

2 Insertion of Commas and Linefeeds into Lecture Subtitle Text

2.1 Problem Settings

This study assumes the use of a dedicated display for subtitle text, which is integrated with a screen displaying presentation slides in the subtitle presentation environment in a lecture hall. The text is designed to swap lines and display several lines continuously while scrolling. Figure 1 shows the subtitle presentation environment.

Moreover, when displaying the transcribed text of the speech on the display, if the text is presented without considering word boundaries to fit the display width, it may be difficult to read. An example of this scenario is shown in Figure 2. For subtitle text, it is essential to insert commas and linefeeds in readable positions, as shown in Figure 3, to match the speaker's speech speed and facilitate easy reading.

This paper proposes a method to insert commas and linefeeds at appropriate positions to improve the readability of lecture texts. Note that it is essential to define the assumptions about the input text, and in a subtitle presentation system, there are several ways to convert speech into text. Representative methods include speech recognition systems (Yu and Deng, 2016) and computer-aided transcription (Kurita, 2016). For example, with speech recognition, there are approaches that can be used to directly recognize the speech or recognize the speech based on the speaker's repetition (Yu and Deng, 2016). Furthermore, there is an option to correct recognition errors manually (Errattahi et al., 2018). Similarly, in computer-aided transcription, the transcription style can vary depending on the transcriber or transcription tool.

Due to these differences in transcription methods, the input text may (or may not) contain recognition errors. If errors are present, the extent of these errors may vary. In this study, we assume a system where speech is transcribed automatically using an automatic speech recognition or computer-aided transcription method. Based on the progress of transcription, the lines are presented to the system as comma and linefeed positions are determined.

It is also necessary to set the maximum number of characters per line that can be shown on the display. In this study, the number is set at 20, taking into account the relation between readability and font size on the display.

2.2 Related Work

2.2.1 Comma Insertion

In English, comma insertion is typically considered to be a part of the punctuation restoration task. In the punctuation restoration task, missing punctuation marks in text are restored or inserted. Punctuation marks, e.g., commas, periods, question marks, and exclamation marks, are important in terms of conveying the intended meaning and structure of written language (Sato, 2000). Several studies have investigated inserting commas into English texts, and various comma insertion methods have been proposed, including methods that combine a bidirectional recurrent neural network and an attention mechanism (Tilk and Alumäe, 2016), and methods that utilize lightweight neural networks based solely on self-attention for feature extraction (Wang et al., 2018).

Conversely, in Japanese language, commas are also used to indicate divisions within a sentence.

Although there are some similarities with the commas used in English and other languages, differences also exist in the Japanese context. Regarding studies on inserting commas into Japanese texts, there are statistical methods that utilize features based on the specific use of commas and conditional random field methods that incorporate lexical information, pause information, and syntactic information. These methods are designed to identify appropriate comma insertion positions based on various linguistic features (Murata et al., 2010) and contextual information in Japanese texts (Akita and Kawahara, 2011).

2.2.2 Linefeed Insertion

In English writing, spaces are utilized to separate words. When a sentence is split across multiple lines, it is typically divided at the spaces, or hyphens may be used. Alternatively, in Japanese, spaces are not used to separate words, and various character types, e.g., kanji, hiragana, and katakana, are frequently mixed together. Thus, determining appropriate linefeeds at meaningful boundaries is even more important.

Several studies have investigated the linefeed insertion problem for Japanese texts. For example, some methods determine linefeed positions based on the patterns of morpheme sequences in the closed captions of TV programs (Monma et al., 2003). Statistical approaches incorporate various information, e.g., dependency relationships, clause boundaries, pauses, and line length, analyzed from linefeed positions (Murata et al., 2009), and progressive insertion methods attempt to realize real-time subtitle generation (Ohno et al., 2009). These methods attempt to identify optimal positions for inserting linefeeds in Japanese texts to improve readability and comprehension for the users of subtitle services.

2.3 Relationship between Comma Insertion and Linefeed Insertion

Furthermore, a previous study investigated the sequential insertion of commas and linefeeds to format the text (Murata et al., 2011). In this method, commas are first inserted into text does not contain commas and linefeeds, and then the linefeeds are inserted into text where the commas have been inserted while considering that the length of each line does not exceed the maximum number of characters per line. However, the positions of the comma and linefeeds insertions can affect each other, and with this sequential approach, it may be impossible to adjust the comma insertion based on the position of the linefeed. The following example illustrates this issue.

- 駅まで自転車で向かう自分はやはりどこか仕事に向かうような感覚でした
  (I was riding my bike to the station well I still felt like I was on my way to work.)

First, commas are inserted into the above sentence.

- 駅まで自転車で向かう自分は、やはり、どこか仕事に向かうような感覚でした
  (I was riding my bike to the station, well, I still felt like I was on my way to work.)

Then, linefeeds are inserted to ensure that the maximum number of characters on a line (20 characters in this case) is not exceeded, and the following result is obtained.

- 駅まで自転車で向かう自分は、
  やはり、
  どこか仕事に向かうような感覚でした
  (I was riding my bike to the station,
  well,
  I still felt like I was on my way to work.)

Consequently, a short line "やはり、(well,)" is generated. The mixture of short and long lines can increase eye movements and make the subtitles more difficult to read. Thus, in this case, it is preferable to not insert a comma and linefeed immediately after "やはり(well)" and generate the line "やはりどこか仕事に向かうような感覚でした(well I still felt like I was on my way to work.)" without interruption.

3 Proposed Method for Comma and Linefeed Insertion

In the proposed method, the sentence is the processing unit. The structure of the proposed method is illustrated in Figure 4. Here, the input $X_i (1 \leq i \leq N)$ comprises the token x_i and its preceding and succeeding $2n$ tokens $x_{i-n}, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{i+n}$. x_i transitions from the first token x_1 to the last token x_N of the sentence sequentially. The input includes the [CLS] and [SEP] tokens, which indicate the start and end positions, respectively.

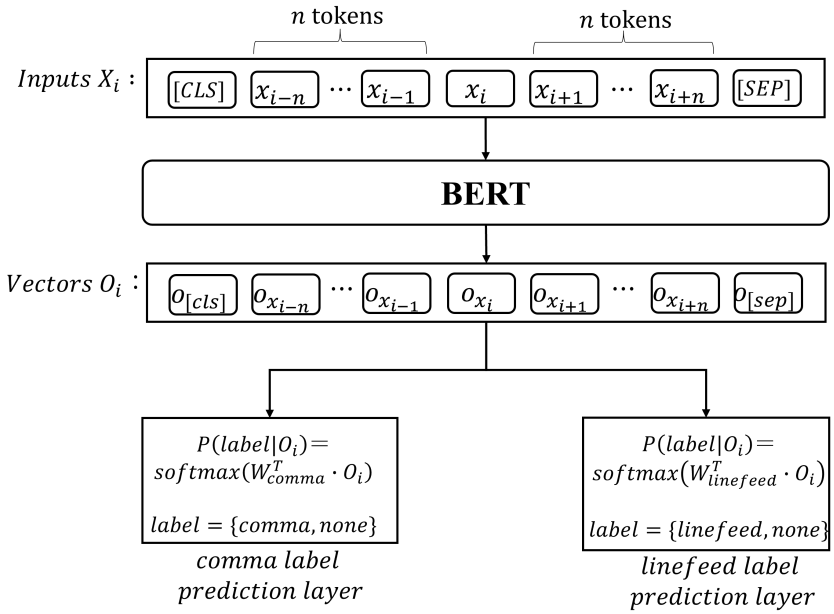

Figure 4: Structure of the model. W_{comma} and $W_{linefeed}$ are independent parameters of the comma label prediction layer and linefeed label prediction layer, respectively.

3.1 Comma and Linefeed Insertion through Multi-Task Learning

In the proposed method, the comma insertion and linefeed insertion tasks are executed as multi-task learning. Typically, individual models must be trained for each of these tasks. However, with multi-task learning, a single model can be utilized to perform multiple tasks, and performance improvement is expected by capturing common factors between the tasks (Zhang and Yang, 2018).

Here, two prediction layers are added to the BERT model to perform the comma insertion and linefeed insertion tasks. Each token in the input X_i is converted to a vector by the BERT model and input to the comma label prediction and linefeed label prediction layers. In the comma label prediction layer, the Softmax function is employed to calculate the probabilities of the labels $comma, none$. Here, $comma$ indicates that a comma is inserted immediately after token x_i when the probability of $comma$ is greater than that of $none$, which means that no comma is inserted. Similarly, in the linefeed label prediction layer, the probabilities of the labels $linefeed, none$ are calculated.

3.2 Line Length Constraint for Linefeed Insertion

When presenting subtitles on the display, it is necessary to insert linefeeds such that the number of characters on each line does not exceed the maximum number of characters allowed by the display. However, the linefeed positions predicted by the model may not satisfy this constraint. In the proposed method, for lines that exceed the maximum width, as shown in the top part of Figure 5, we redetermine the linefeed positions to satisfy the maximum number of characters constraint.

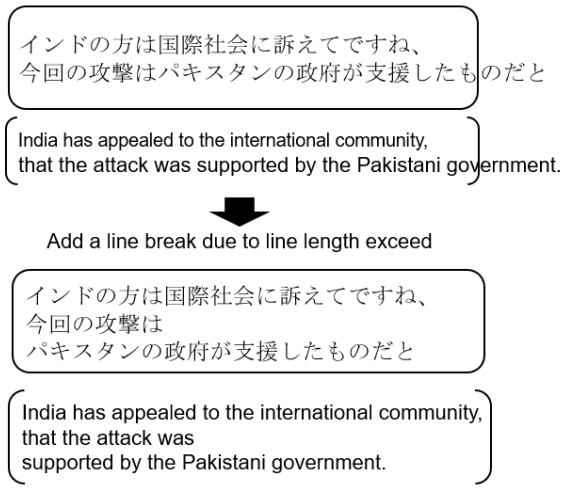

Figure 5: Example of line length exceed

Let $L = (t_1, t_2, \cdots, t_m)$ represent a line containing m tokens, and $Label = (l_1, l_2, \cdots, l_m)$ represent the labels of tokens in L.

For lines that do not satisfy the maximum number of characters constraint, we calculate the following probability $P(Label)$.

Table 1: Size of correct data

Sentences	Tokens	Commas	Linefeeds
1,935	67,032	4,833	5,841

$$P(Label) = P(l_1, l_2, \cdots, l_m = linefeed) \\ = P(l_1) \times P(l_2) \times \cdots \times (P(l_m) = 1) \quad (1)$$

We assume that whether a linefeed is inserted immediately after a certain token is independent of the other tokens. As a result of this assumption, the probability for the token sequence can be calculated by simply multiplying the individual probabilities. Note that a linefeed is always inserted after the last token t_m; thus, $l_m = linefeed$.

By finding the highest probability combination of $Label$ that satisfies the maximum number of characters constraint, we can obtain the updated linefeed insertion results, as shown in the bottom part of Figure 5.

4 Experiment

4.1 Experiment Overview

To evaluate the effectiveness of the proposed method, a comma and linefeed insertion experiment was conducted on Japanese spoken language data.

4.1.1 Experimental Data

The data used in this experiment comprise transcripts from 16 Japanese lecture speeches in the Simultaneous Interpretation Database (Matsubara et al., 2002). The data have been annotated manually with commas and linefeeds. Note that linefeeds were inserted with the constraint of 20 characters per line. The details of the correct data are shown in Table 1.

4.1.2 Experimental Method

In this experiment, we used Tohoku University's bert-base-japanese model[1] as the pretrained Japanese BERT model. Moreover, the Adam optimizer and cross-entropy loss were used as the optimizer and loss function, respectively. For the hyperparameters, the batch size was set to 16, the number of epochs was set to five, and the learning rate was set to 1e-5. The variable n was set to eight, which means that eight tokens before and eight tokens after the token x_i would be used as input for the model.

This experiment involved performing cross-validation using the data for all 16 lectures. Here, in a 14-fold cross-validation, two lectures were used as the validation data, thirteen lectures were used as the training data, and the remaining lecture was used as the testing data.

4.1.3 Compared Methods

The following two methods were considered to compare the performance of the proposed method.

- Method from a previous study (Murata et al., 2011): This method, as described in section 2.3, first identifies the positions for inserting commas, and then performs the linefeed insertion. In this method, positions for inserting commas and linefeeds are determined base on maximum entropy method using morphological information, dependency relations, and clause boundary information.

- Baseline: To assess the effect of multi-task learning, we established the method of inserting commas and linefeeds sequentially using BERT. We trained two BERT models separately, one for comma insertion and the other for linefeed insertion. In the baseline method, positions for comma insertion are identified first, followed by linefeed insertion. This procedure enables us to evaluate the impact of multi-task learning, which involves the simultaneous insertion of both commas and linefeeds.

4.1.4 Evaluation Metrics

The performance of the compared methods was evaluated in terms of the recall, precision, and F-measure metrics. Here, the recall and precision for the comma insertion task were calculated as follows.

$$Recall = \frac{\#\ of\ correctly\ inserted\ commas}{\#\ of\ commas\ in\ the\ correct\ data}$$

$$Precision = \frac{\#\ of\ correctly\ inserted\ commas}{\#\ of\ automatically\ inserted\ commas}$$

Note that the same formulas were used to evaluate the linefeed insertion task.

[1] https://huggingface.co/cl-tohoku/bert-base-japanese

Table 2: Experimental results of comma and linefeed insertion

	comma insertion			linefeed insertion		
	Recall(%)	Precision(%)	F1	Recall(%)	Precision(%)	F1
Our method	**76.69** (3,103/4,046)	79.71 (3,103/3,893)	**78.17**	**82.98** (4,184/5,042)	**69.29** (4,184/6,038)	**75.52**
Murata method (Murata et al., 2011)	71.65 (2,899/4,046)	**81.07** (2,899/3,567)	76.07	76.91 (3,878/5,042)	69.19 (3,878/5,605)	72.85
Baseline	70.17 (2,839/4,046)	76.44 (2,839/4,004)	73.17	67.91 (3,424/5,042)	68.91 (3,424/4,969)	68.41

4.2 Experimental Results

The experimental results are shown in Table 2. As can be seen, the proposed method achieved a recall of 76.69% and precision of 79.71% for the comma insertion tasks. For the linefeed insertion task, the proposed method obtained a recall of 82.98% and precision of 69.29%. The proposed method outperformed the baseline and previous methods, thereby confirming its effectiveness.

4.3 Analysis of Comma Insertion Errors

In this section, we discuss the errors observed in the experimental comma insertion task. Among the positions where commas were inserted in the correct data, the proposed method failed to detect some of these cases correctly. One such example is when commas were intended to separate parallel nouns. The following example is used to facilitate this discussion.

- なぜか掲載のスペースが小さいお菓子類の広告、その他、自動車教習所、アルコール、ホテル、そして、ディナーショー、レストラン、旅行、薬_バーゲン
  (Advertisements for snacks and other items seem to have unusually small spaces for publication, additionally, there are advertisements for driving schools, alcohol, hotels, dinner shows, restaurants, travel and medicine_ bargains.)

In this example, the comma between the parallel nouns "薬(medicine)" and "バーゲン(bargain)" was not detected. As a result, there is a possibility that the audience may misinterpret "薬(medicine)" and "バーゲン(bargain)" as one word "薬バーゲン(medication bargain)", without the intended separation.

Among the 378 commas intended to separate parallel nouns in the test data, the proposed method failed to detect 126 of these cases. Here, the recall rate was only 66.7% (252/378), which is lower than the overall recall rate for the comma insertion task.

To effectively detect commas to separate parallel nouns, it is believed that utilizing part-of-speech (POS) information would be beneficial, which is discussed in Section 5.

4.4 Analysis of Linefeed Insertion Errors

Here, we discuss the errors identified in the experimental linefeed insertion task. As a result of the analysis, it was evident that linefeed insertion errors led to the generation of extremely short lines. The presence of a mixture of short and long lines can cause frequent shifts in the viewer's gaze, which can potentially hinder comprehension of the subtitles (Iwasaki and Kurimoto, 1988). This is demonstrated using the following example.

- 島内では
  魚の豊富な川や湖、小川などにめぐまれ、
  この大地を一層肥沃な大地に
  (On the island
  blessed with rivers, lakes and brooks rich in fish,
  this makes the land even more fertile.)

In this example, as a result of linefeed insertion, the first line has a length of only four characters, and the second line has a length of 19 characters, thereby creating a significant disparity in the line lengths.

Thus, we measured and compared the occurrence frequencies of the line lengths in the correct data and the experimental results. The corresponding results are shown in Figure 6. The average line length in the correct data was 14.7, and that in the experimental results was 14.0. According to a previous study (Murata et al., 2009), the frequency of lines within six characters is relatively small, and linefeeds are more likely to be inserted to ensure that lines are at least seven characters long. In the correct data, there were 106 occurrences

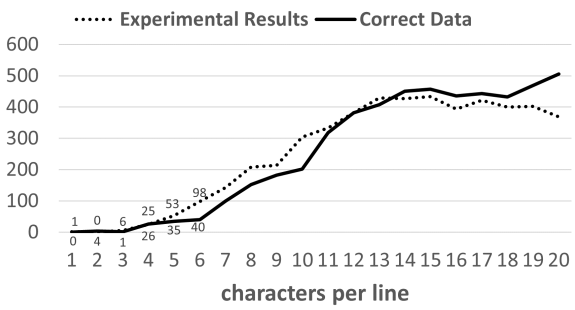

Figure 6: Comparison of characters per line in experimental results and correct data

of lines within six characters, whereas there were 183 occurrences in the experimental results. Thus, the experimental results included more linefeeds to make short lines.

5 Using POS Information in Comma and Linefeed Insertion

In this section, we discuss the use of POS information for comma and linefeed insertion. By explicitly incorporating POS information in the model, we attempt to reduce the frequency of comma insertion errors (Section 4.3). In Section 5.1, we describe the method of utilizing POS information in the proposed approach, and in Section 5.2, we discuss the comma and linefeed insertion experiments utilizing the POS information.

5.1 Using POS Information in Proposed Method

In previous research on automatic comma insertion in Japanese text (Murata et al., 2009), high accuracy in comma insertion has beed achived by a statistical approach using information such as morphemes, including POS information. Therefore, incorporating POS information into the model allow the model to more accurately capture sentence structure and insert commas to separate parallel nouns.

In the proposed method, we used the MeCab (Kudo, 2005) morphological analyzer, which utilizes the IPA dictionary as a reference, to add POS information as tags to tokens. Here, the tokens are smaller units than words. For example, the word "走る(run)" with the POS tag "動詞(verb)" is divided into "走" and "##る". For such subword tokens, we assign the POS tag "動詞(verb)" to "走" and the POS tag "##動詞(##verb)" to "##る".

The structure of the model utilizing POS information is shown in Figure 7. Here, both the input $X_i(1 \leq i \leq N)$ and its corresponding POS tags $P_i(1 \leq i \leq N)$ are converted to embedding vectors $O_i = o_{[cls]}, o_{x_{i-n}}, \ldots, o_{x_i}, \ldots, o_{x_{i+n}}, o_{[sep]}, o_{p_{i-n}}, \ldots, o_{p_i}, \ldots, o_{p_{i+n}}$ by the BERT model and used in the comma insertion and linefeed insertion tasks, respectively.

5.2 Effect of POS Information

To confirm the effectiveness of using POS information in the comma and linefeed insertion tasks, we conducted an experiment using the same settings described in Section 4.1.

The experimental results obtained using POS information in the proposed method are shown in Table 3. For the comma insertion tasks, we confirmed that using the POS information improved the accuracy. However, for the linefeed insertion task, no significant improvement was observed.

We use the examples presented in Section 4.3 to discuss the comma insertion results obtained using the POS information.

- なぜか掲載のスペースが小さいお菓子類の広告、その他、自動車教習所、アルコール、ホテル、そして、ディナーショー、レストラン、旅行、薬、バーゲン
  (Advertisements for snacks and other items seem to have unusually small spaces for publication, additionally, there are advertisements for driving schools, alcohol, hotels, dinner shows, restaurants, travel, medicine, bargains.)

As shown in the example, the comma between the parallel nouns "薬(medicine)" and "バーゲン(bargain)" was detected correctly.

Among the 378 commas used to separate parallel nouns in the correct data, 293 were detected correctly using POS information. Here, the recall rate was 77.5% (293/378), representing an improvement of 10.8 percentage points compared to the results obtained by the proposed method without the POS information.

However, there were still cases where the comma placement to separate nouns was not detected correctly. Here, we consider the following example.

- それと衛星のフライトダイナミックス、航空力学のソフトですね、これを作ろうというのに、同じプロセス、メソッド、組織、ツール、それらを使って両方作ってしまおう

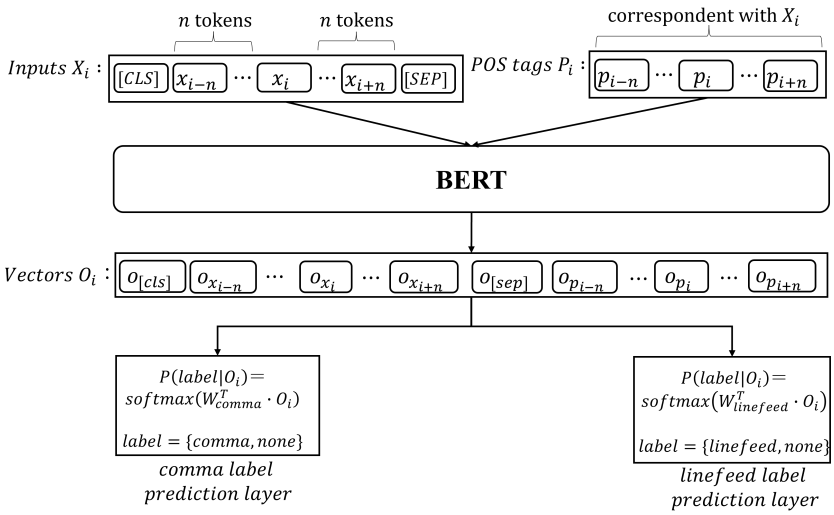

Figure 7: Structure of the model using part-of-speech information.

Table 3: Experimental results of comma and linefeed insertion with part of speech information

	comma insertion			linefeed insertion		
	Recall(%)	Precision(%)	F1	Recall(%)	Precision(%)	F1
Our method+ POS information	**76.99** (3,115/4,046)	80.91 (3,115/3,850)	**78.90**	**83.58** (4,214/5,042)	68.89 (4,214/6,117)	75.52
Our method	76.69 (3,103/4,046)	79.71 (3,103/3,893)	78.17	82.98 (4,184/5,042)	**69.29** (4,184/6,038)	75.52

(And also, the flight dynamics software for satellites, aerodynamics software, and we will use the same process, method, organization_ tool to create both.)

In this example, the comma between the nouns "組織(organization)" and "ツール(tool)" was not detected correctly. The word "ツール(tool)" only appeared once in the training data, and it is believed that the lack of sufficient training data affected the detection. To address this issue, a possible solution is to perform pretraining on a large amount of training data for comma insertion, and then fine-tune the model using data with inserted commas and linefeeds.

6 Conclusion

This paper has proposed a method to provide easy-to-read lecture subtitles for visually impaired individuals and nonnative speakers in lecture halls. The proposed method inserts commas and linefeeds into lecture text simultaneously to achieve this goal. The comma and linefeed insertions are realized using a multi-task learning approach with a pretrained BERT model. The proposed method was evaluated experimentally, and the experimental results obtained on transcribed Japanese spoken language data demonstrated an F-measure of 78.17% for comma insertion and 75.52% for linefeed insertion, confirming the effectiveness of the proposed method.

However, in the proposed method, the decision to insert commas and linefeeds after a token is based on the information from n preceding and succeeding tokens, which may cause a delay between the speech and the display of subtitles. To address the need for real-time subtitle generation, Iwamura et al. (2021) proposed a method that considers the remaining length of the sentence. Generally, there is a relationship between the remaining length and the position of linefeeds, i.e., as the remaining length decreases, the necessity of linefeeds decreases. Therefore, in the future, we plan to implement the utilization of the remaining length into the proposed method to realize real-time subtitle generation.

Limitations

The method proposed in this paper has limitations, which are discussed here. The proposed method

is based on information from a fixed number of preceding and succeeding tokens, which can result in a delay between speech and displaying the corresponding subtitle. While this study focuses on the Japanese language, and the research dataset consists of Japanese language data, it's crucial to consider that comma and linefeed insertion positions may exhibit language-specific characteristics in other languages. Therefore, when extending the study's results to different languages, it is essential to conduct data collection and model adjustments adapted to those languages.

Acknowledgements

This research was partially supported by JSPS Grant-in-Aid for Scientific Research (C) Grant Number JP22K12122.

References

Yuya Akita and Tatsuya Kawahara. 2011. Automatic comma insertion of lecture transcripts based on multiple annotations. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (Interspeech 2011)*, pages 2889–2892.

Rahhal Errattahi, Asmaa El Hannani, and Hassan Ouahmane. 2018. Automatic speech recognition errors detection and correction: A review. *Procedia Computer Science*, 128:32–37.

Yuka Iwamura, Tomohiro Ohno, and Shigeki Matsubara. 2021. Linefeed insertion in speech text considering remaining sentence length. In *Proceedings of the 83rd National Convention of the Information Processing Society of Japan (IPSJ 2021)*, volume 1, pages 545–546. (in Japanese).

Tsutomu Iwasaki and Shinji Kurimoto. 1988. Changes in adjustment time and ocular fatigue associated with eye movement. *Ergonomics*, 24(Supplement):196–197. (in Japanese).

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, volume 1, page 2.

Taku Kudo. 2005. MeCab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. net/*.

Shigeaki Kurita. 2016. Development of CART software "IPtalk" and captioning services via personal computers: Communication support for the hearing-impaired. *Information Processing and Management*, 59(6):366–376. (in Japanese).

Shigeki Matsubara, Akira Takagi, Nobuo Kawaguchi, and Yasuyoshi Inagaki. 2002. Bilingual spoken monologue corpus for simultaneous machine interpretation research. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 153–159.

Takao Monma, Eiji Sawamura, Takahiro Fukushima, Ichiro Maruyama, Terumasa Ehara, and Katsuhiko Shirai. 2003. Automatic closed-caption production system on TV programs for hearing-impaired people. *Systems and Computers in Japan*, 34(13):71–82.

Masaki Murata, Tomohiro Ohno, and Shigeki Matsubara. 2009. Automatic linefeed insertion for improving readability of lecture transcript. *New Directions in Intelligent Interactive Multimedia Systems and Services-2*, pages 499–509.

Masaki Murata, Tomohiro Ohno, and Shigeki Matsubara. 2010. Automatic comma insertion for Japanese text generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 892–901.

Masaki Murata, Tomohiro Ohno, and Shigeki Matsubara. 2011. Automatic text formatting for social media based on linefeed and comma insertion. In *Proceedings of the 4th International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS 2011)*, pages 285–294. Springer.

Tomohiro Ohno, Masaki Murata, and Shigeki Matsubara. 2009. Linefeed insertion into Japanese spoken monologue for captioning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2009)*, pages 531–539.

Masamitsu Sato. 2000. Japanese comma - a reexamination of the rules. *The bulletin of arts and sciences, Meiji University*, 331:1–18. (in Japanese).

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (Interspeech 2016)*, pages 3047–3051.

Feng Wang, Wei Chen, Zhen Yang, and Bo Xu. 2018. Self-attention based network for punctuation restoration. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018)*, pages 2803–2808. IEEE.

Dong Yu and Lin Deng. 2016. *Automatic speech recognition*, volume 1. Springer.

Yu Zhang and Qiang Yang. 2018. An overview of multitask learning. *National Science Review*, 5(1):30–43.