LIAT Team's Extractive Summarizer at NTCIR-15 QALab PoliInfo-2

Kouta Nakayama
RIKEN AIP, Japan
kouta.nakayama@riken.jp

Satoshi Sekine
RIKEN AIP, Japan
satoshi.sekine@riken.jp

ABSTRACT

This is the report of the summarization system that our team LIAT submitted to the dialog summarization task in NTCIR-15 QALab PoliInfo-2. We designed an extractive summarizer by dividing the task into three parts and training a model for each. Analysis from the scores showed that the line level extractive summarizer that we created did not suit the task.

TEAM NAME

LIAT

SUBTASKS

QALab PoliInfo-2 (Dialog summarization task)

1 INTRODUCTION

NTCIR-15 QALab PoliInfo-2[2] is a shared task that deals with political documents in Japan. The dialog summarization task we participated in is a task to parse and summarize the dialogue structure of local councils. For more information on the task, see the PoliInfo-2 organizer's paper[2]. Note that the formal submission was found to contain a bug, so this paper describes the content of the late submission. Therefore, there are no manual evaluation results.

2 SYSTEM DESCRIPTION

We created three models, border detector, topic matcher and summarizer, to solve the task. We use BERT[1] for those models. We explain the role of those models below.

2.1 Border Detector

The dialogue summarization task needs to detect a range of question or answer sentences at first. The border detector estimates the range of question or answer sentences by determining their boundaries. A boundary line is predicted by a binary classification of whether the boundary line is between two sentences. The training data for border detectors is generated from segmented training data. The segment boundaries are treated as positive and the line boundaries inside the segment as negative. We sampled up to two boundaries from inside each segment to avoid bias towards negative examples. Sentence 1 and sentence 2 are combined with a special token as follows and passed to BERT.

[CLS] Sentence 1 [SEP] Sentence 2 [SEP]

BERT then predicts whether the sentence boundary is the beginning or end of the question or answer from the output corresponding to the [CLS] token. In more detail, we have linear layers of $f_{beginning\_question}$, $f_{end\_question}$, $f_{beginning\_answer}$, and f_{end_answer}, each of which is connected to the output of BERT for binary classification.

2.2 Topic Matcher

The topic matcher matches segments and subtopics. We use segmented training data for training, which is positive if the segment is about a subtopic and negative otherwise. BERT will be passed the subtopic and the beginning and end of the segment as following.

[CLS] Sub topic [SEP] Beginning of the segment [SEP]
End of the segment [SEP]

If the input length is exceeded, we cut out the beginning and end of the segment appropriately. BERT predicts from the output corresponding to [CLS] whether a segment is valid for a given subtopic or not. BERT predicts whether a segment corresponds to a given subtopic. Specifically, we connect a linear layer f_{bc_tp} to the output of BERT for binary classification. Let us denote the output of BERT for [CLS] token as x_{cls}. We get the final output using the softmax function f_{soft} as follows.

$$Pr(y|x_{cls}) = f_{soft}(f_{bc\_tp}(x_{cls}))$$

Since more than one segment may be assigned for a subtopic, we select one where $Pr(y = 1|x_{cls})$ is higher than the threshold t. If no segment exceeds the threshold, then the segment with the largest $Pr(y = 1|x_{cls})$ is applied.

2.3 Summarizer

We create an extraction summarizer at the line level. Therefore, it is necessary to identify the training data lines in advance that could be the answer. We took the line that shares the most nouns with the gold summary as the correct answer. Also, we used the results of the topic matcher on the unsegmented training data as training data. The characteristics of a sentence that could be a summary may be very different for a question and an answer. Therefore, we trained a different summarizer with questions and answers. We define the input to BERT as follows. Herein, the number of lines n to be passed simultaneously is determined by the maximum input length of BERT.

[CLS] Sub topic [SEP] [Mask] Line 1 [MASK] Line 2
... [MASK] Line n [SEP]

We get prediction results for [MASK] tokens placed in front of each line. Specifically, we connect the common linear layer f_{bc_s} for binary classification to the outputs of BERT, which correspond to each [MASK] token. Let x_{mask_i} be the prediction of BERT for

Hyperparameter	
Epoch	10
Batch size	32
Gradient accumulation steps	1
Sequence length	512
Weight decay	0.01
Max grad norm	1.0
Hidden dropout	0.3
Attention dropout	0.1
Optimizer	Adam
Learning rate	5e-5
Adam beta 1	0.9
Adam beta 2	0.999
Adam eps	1e-6

Table 1: Common hyperparameters.

the [MASK] token connected to line i. Using the softmax function f_{soft}, the final output is obtained as follows.

$$Pr(y_i|x_{mask_i}) = f_{soft}(f_{bc\_s}(x_{mask_i}))$$

We use the threshold as well as the topic matcher to select the outputs, as we allow multiple lines to be a summary.

3 EXPERIMENTS

3.1 Data Preprocessing

We use MeCab[3] to tokenize the corpus. And, we use Juman[4] as a dictionary for MeCab. During tokenization, we also collect the nouns that are used to create training data for summarizer.

3.2 Experimental settings

We use the pre-trained parameters[1] for BERT. We used the hyperparameters in Table 2 for training unless mentioned otherwise and train the models with mixed precision floating point arithmetic [5]. We used the minutes of two meetings with recent dates in the segmented data as development data.

Topic Matcher. We set the batch size to $b = 128$ and the threshold for output selection to $t = 0.9$.

Summarizer. We set the batch size to $b = 8$ and the threshold for output selection to $t = 0.5$.

3.3 Results

We present our results with the development data in Table1. We see that the border detector and the topic matcher score are relatively high. In other words, our system seems to be able to handle data that doesn't have segments. However, the summarizer scores are quite low. Because the summary lines are identified by noun matching with each gold summary, they may contain a lot of noise. Given that segmentation is working well, perhaps we should create a generative summarizer.

The scores on the leader board are shown in Table 3. ROUGE-1-R is a macro average of the content words. JRIRD is the best result on the leaderboard and seems to be a generative summary. Our summarizer is greatly inferior. Line level extractions are likely to contain irrelevant parts and are likely to have lower scores.

[1]We selected a model that does not use BPE. https://alaginrc.nict.go.jp/nict-bert/index.html

Model	Precision	Recall	F1
Border detector	0.928	0.967	0.947
Topic matcher	0.915	0.874	0.894
Summarizer (question)	0.594	0.632	0.612
Summarizer (answer)	0.607	0.591	0.599

Table 2: Scores on development data.

System	ROUGE-1-R
JRIRD (formal submission)	0.321
Ours (late submission)	0.095

Table 3: Scores on leader board.

4 CONCLUSIONS

This paper describes the system submitted to PoliInfo-2. Analysis from the scores shows that line level extractive summarizers do not seem to match the task very well. Future research will include detailed analysis and the creation of a generative summarizer.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and 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, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[2] Yasutomo Kimura, Hideyuki Shibuki, Hokuto Ototake, Yuzu Uchida, Keiichi Takamaru, Madoka Ishioroshi, Teruko Mitamura, Masaharu Yoshioka, Tomoyoshi Akiba, Yasuhiro Ogawa, Minoru Sasaki, Kenichi Yokote, Tatsunori Mori, Kenji Araki, Satoshi Sekine, and Noriko Kando. 2020. Overview of the NTCIR-15 QA Lab-PoliInfo-2 Task. *Proceedings of The 15th NTCIR Conference* (12 2020).

[3] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Barcelona, Spain, 230–237. https://www.aclweb.org/anthology/W04-3230

[4] S. KUROHASHI. 1994. Improvements of Japanese Morphological Analyzer JUMAN. *Proceedings of The International Workshop on Sharable Natural Language, 1994* (1994), 22–28. https://ci.nii.ac.jp/naid/10027016015/

[5] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed Precision Training. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1gs9JgRZ