NTU IR Final Project Report

Group 12 (TEAM_2553)

r10946008 李品澤 r11922100 曾奕崴 r09b48004 蕭如秀

1 Introduction

This is the implementation report about the competition of argument mining held by T-Brain `https://tbrain.trendmicro.com.tw/Competitions/Details/26`. In this competition, we aim to find the claim of the provided text and provide the key paragraph that supports the claim at the same time. Providing the key paragraph helps advance research in natural language processing, and reveals more information about the built model to researchers. Our code is available in `https://github.com/aqweteddy/NTU_IR_final`.

2 Method

This competition is similar to the text summarization task. We treated it as a seq2seq problem and used a pre-trained seq2seq model (T5[1] / Long T5[2]) and various tricks to achieve 8th place in the competition. Fig.1 is a simple illustration of our approach.

2.1 Data Preprocessing

2.1.1 Hard Prompt

We design two types of templates to concatenate q, r, and s.

q is <s>with [r]. [q] <q>[r] <r>

q <q>is <s>with [r] <r>

where [q] and [r] are special tokens denote begin of q and r. <q>represents the sentence of q and r. In the decoding phase, [q] and [r] also are regarded as begin of the sentence token.

2.1.2 Same Word Replacement Augmentation

To improve the robustness of our model and prevent overfitting, we use an augmentation strategy in the preprocessing stage.

The competition states that q' must be a substring of q, that is, $q = [w_0, w_1, .., w_n], q' = [w_i, w_{i+1}, ..., w_j]$. We randomly sample a word w_k from q', where $i \leq k \leq j$, then use a pretrained word2vec model[3] to find the top 5 similar words $w_{k0'}...w_{k4'}$ to w_k, and finally, randomly select a word $w_{k'}$ to replace w_k in q and q'. This process is repeated for each word w_k with a probability of 0.1. We also apply the same strategy to r.

Symbols	means
q	source article q
r	source article r
s	relation between q and r
q'	target explanation of q
r'	target explanation of r
q_g	model generated explanation of q
r_g	model generated explanation of r

Table 1: Annotation used in this report.

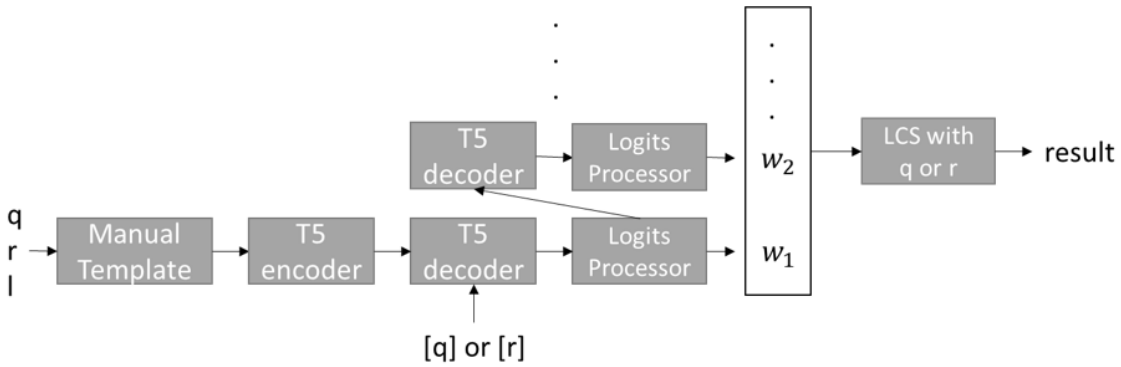

Figure 1: Pipeline of our method. q, r, and l are the input data. The manual hard template concatenates the data as input to T5. The target output is in front. [q] [r] in the decoder are special tokens representing the output of q' or r'. The logits processor modifies the logits of words not in q and r with -inf, achieving the effect of only copying from q and r. The decoding strategy is greedy. Finally, LCS is performed with the input sentence q or r to improve the results.

2.2 Calibration in Inference Stage

2.2.1 Decoder Logits Restirction

To prevent the model from generating words that do not exist in q and r, which goes against the competition premise, we restrict the generated words to those that appear in q and r during decoding.

Formally, let l_i denote the prediction logits when generating the i-th token of q (q_{gi}), where the dimension is the decoder vocabulary size. And the set of words appearing in q is V. Then we apply the following mask:

$$mask = words\ not\ in\ V \tag{1}$$

$$l_i[mask] = -\infty \tag{2}$$

We apply the same strategy when generating r_g.

2.2.2 Align with Source Sentence

We only restrict the generated words to those that appear in q and r, but the scoring method LCS is directional. For example, "This is an apple." and "This apple is an." are the same words, but the LCS length is 2.

According to the official metric formula, the longer sequence pairs that have the same LCS score obtain lower scores. To deal with this problem, we compute LCS between the source sentence (q or r) and generated sentence as the final answer.

3 Training Details

In our experiments, we split the original training set into two parts: a training set and a validation set. We then apply early stopping based on the validation loss.

We conduct the experiment based on the transformers library and Adam as the optimizer with a learning rate of 1e-4 and a warmup scheduler. The maximum input and output lengths are set to 1024 and 250, respectively. 15% of the data is augmented for data augmentation. The batch size is set to 8. We also apply label smoothing 0.15 when calculating cross-entropy loss. Our early stopping strategy is based on validation loss.

4 Experiment & Discussion

The Long T5 [2] model generates our best result. Throughout the whole competition, we've also tested many other models. Below is a table summarizing the models we've experimented with and their results.

Model	Public score
Bart-base [4]	0.788
Bart-base-CNN	0.777
Pegasus [5]	0.791
T5-small	0.798
T5-base [1]	0.809
T5-large	0.819
Long T5-base	**0.834**
Long T5-large	0.803

Table 2: Overall comparison

As the result shows, we found that Long T5-large doesn't outperform Long T5-base. This might be caused by its large architecture requiring more training time, and thus we don't have that much time to find satisfying hyper-parameters.

4.1 Efficient of Data Augmentation

In our preprocessing stage, we apply data augmentation to increase the robustness of our model and prevent overfitting. We show different augmentation probability results in Table3. We observe a 0.03 increase when using the same word replacement augmentation on long T5 base.

Method	public score
Long-T5 Base w/o aug.	0.801
Long-T5 base with aug(prob=0.l5)	**0.834**
Long-T5 base with aug(prob=0.25)	0.827
Long-T5 base with aug(prob=0.5)	0.825

Table 3: Comparison with and without augmentation

4.2 Why Long-T5 Perform Better than T5 ?

There are two reasons we guess:

- Long-T5 adopt pegasus[6] as pretraining task.This self-supervised pretraining method is state-of-the-art in summarization and is more relevant to this competition than T5.
- Long-T5 can deal with longer sentences than T5.

4.3 Sequence Labeling v.s. Conditional Generation

Besides the method mentioned above, we have also tried the sequence labeling method. Below is the best score achieved by either method.

Method	Public score
Sequence Labeling	0.726
Conditional Generation	0.834

Table 4: Comparison between methods

We empirically found that sequence labeling does not outperform our final method when a similar encoding model is used, hence we stop trying this method.

4.4 Use of LCS

When our model generates its prediction, we further calculate the LCS score between this prediction and the original sentence. We then try to find the possible substring of the prediction that maximizes the LCS score. Here is the comparison before and after this method.

Model	Before LCS	After LCS
T5-base	0.795	0.809
Long T5-base	0.807	0.834

Table 5: Comparison before and after LCS

We can easily see that this method indeed boosts the LCS score, which is as intended.

5 Conclusion

In this project, we treated this task as text summarization and used a pre-trained seq2seq method to address it. We also applied logit restriction, data augmentation, and other tricks to improve our results. In the end, we achieved 8th place in this competition.

One limitation of our method is the problem of LCS direction. Although we reduce this disadvantage by aligning with the source sentence, we still cannot perfectly solve this problem. There are some ways to improve performance in the future:

- Pointer Generation T5: edit cross attention to directly copy words from the source sentence (q, r).
- Two-stage filter: train a classifier to obtain a confidence score about (q', r') to explain (q, r). We can rank the result of the beam search to find the most confident result.
- generation-based text augmentation:train a reverse (q',r') to (q, r) generation model M_g, create a large number of new examples using M_g, and then train the model on the augmented data.

6 Contribution

Member	Contribution
李品澤	模型建立、報告撰寫
曾奕崴	模型建立、報告撰寫
蕭如秀	報告撰寫

References

[1] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[2] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019.

[5] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2019.

[6] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2019.