Explainable Information Tagging for Natural Language Understanding

NTU IR Final Project

Jia-Wei Liao, Jung-Mei Chu, and Chia-Chi Huang

{D11922016, D09944017, P10922001}@csie.ntu.edu.tw

February 4, 2024

Abstract

In this competition, we demonstrate our skills by constructing an explainable deep learning model for a Natural Language Processing (NLP) task. We cleverly convert this complex task into a summary task, allowing us to effectively tackle the problem at hand. Initially, we use a LSTM model to do the summary task, but unfortunately, the results are disappointing. Not ones to give up, we switch to using the impressive T5 model with pretrained weights. This model is applied to two different datasets with two unique pre-processing methods, and after a thorough experiment, we are able to surpass the baseline of the original data. Our hard work pays off, and we achieve a fantastic public score of 0.801772 and an impressive private score of 0.85190. Our code is available at https://github.com/jwliao1209/Explainable-NLP.

Keywords: Natural Language Processing (NLP), Explainable AI, Text Summarization, Text-To-Text Transfer Transformer (T5).

1 Introduction

A typical argument mining task involves using a classification model to determine the argumentative structure and evaluate the agreement or disagreement between a pair of premises q and claims r as depicted in Figure 1. The deep learning model currently being utilized has proven to be effective in tackling this type of task.

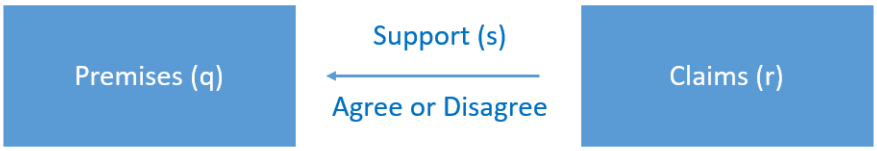


Figure 1: Typical argument mining task.

This project represents an advanced level of argument mining, with a focus on understanding how decision-making models operate through the use of explainable argument mining. The objective of this task is to identify the key statements within the premises q' and claims r' that support the overall argumentative result as shown in Figure 2.

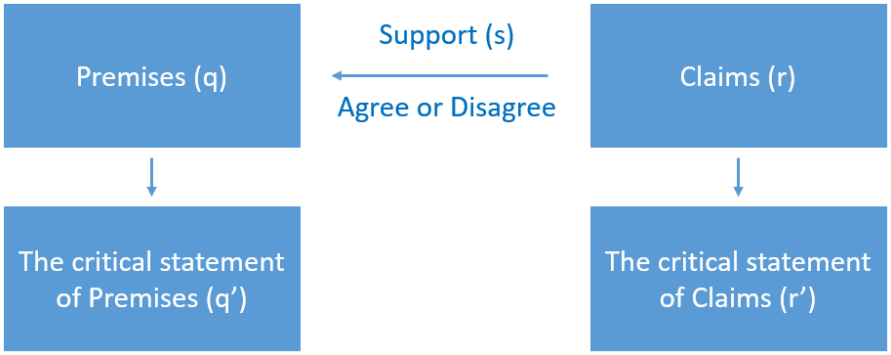


Figure 2: Explainable argument mining task.

2 Method

In this task, extracting the subsequence strings (q', r') from the original strings (q, r) can be thought of as a form of summarization. As such, we decided to use the pre-trained transformer model T5 (Text-To-Text Transfer Transformer) [1] and fine-tune its downstream summarization module. We also experimented with different beam search algorithms and various beam widths to determine their impact on performance.

2.1 Data Preprocessing

We divided the input and output into the following two forms and transformed the problem into a summary task. In addition, we used the [SEP] token to connect q and r.

- Input (Text)
  - Separate: q and r.
  - All: Conjuncting q, [SEP], and r as new input sequences.
- Output (Summery)

- Separate: q' and r'.
- All: Conjuncting q', [SEP], and r' as new output sequences.

Text Type	Text	Summary Type	Summary
all	q + [SEP] + r	all	q' + [SEP] + r'
separate	q r	separate	q' r'
all	q + [SEP] + r	separate	q' r'

Table 1: Data format.

2.2 T5 Model

The T5 model is a pre-trained transformer model that has been trained using a shared text-to-text framework and is capable of taking in text as input and producing modified text as output as illustrated in Figure 3. It has demonstrated state-of-the-art performance on the GLUE benchmark and achieved a score of 88.9 on the SuperGLUE language evaluation.

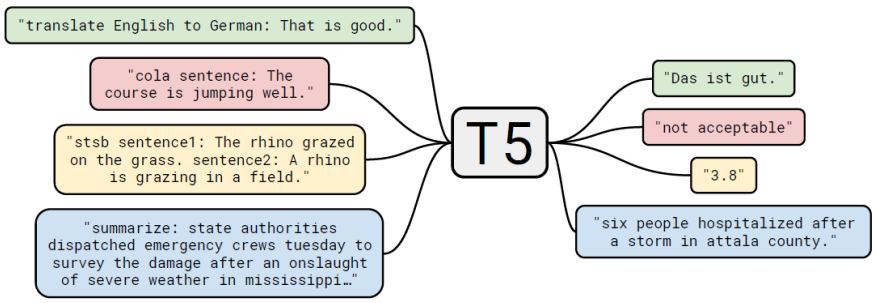


Figure 3: A diagram of T5 text-to-text framework [1].

The research team tested three different architectures: an encoder-decoder model, a language model (LM), and a Prefix LM. After conducting experiments as shown in Figure 4, they selected the encoder-decoder structure as the architecture for T5. The encoder part of this model uses a fully-visible mask, while the decoder uses a causal mask as depicted in Figure 5, left and middle. This architecture is designed to learn from the input data through the encoder and then use that learned model to produce output through the decoder.

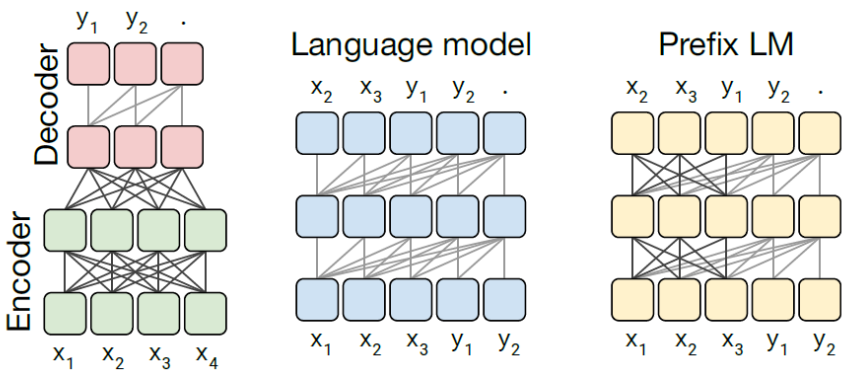


Figure 4: Schematics of the Transformer architecture variants [1].

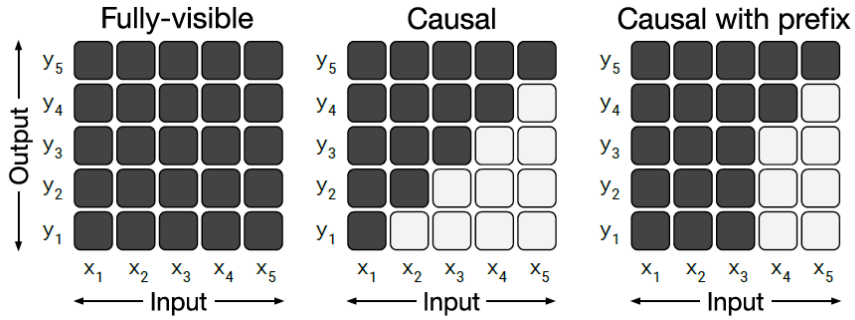


Figure 5: Matrices representing different attention mask patterns [1].

The model architecture serves as the foundation for transfer learning, and the pre-training dataset is a crucial source of knowledge and information. The T5 model was pre-trained on a new open-source dataset called the Colossal Clean Crawled Corpus (C4), which is a cleaned version of the Common Crawl dataset. The C4 dataset has a capacity of up to 750GB, which is nearly 19 times larger than the pre-training dataset used for GPT2 [2].

Utilizing the text-to-text framework, encoder-decoder architecture, and the unlabeled C4 dataset, the next step is to provide the T5 with methods for learning tasks from the data in the C4 dataset as depicted in Figure 6.

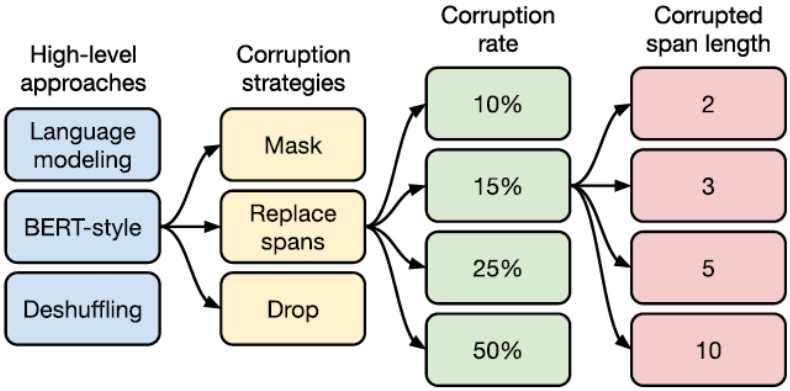


Figure 6: A flow chart of our exploration of unsupervised objectives [1].

- **High-level approaches**: The BERT-style approach, which includes the Masked Language Model and Next Sentence Prediction, performed the best.
- **Corruption strategies**: The Replace Span method, which is also used by the Span-BERT model, had the best results.
- **Corruption rate**: A corruption rate of 15% performed the best, similar to the results obtained with BERT.
- **Corrupted span length**: A length of 3 produced the best results.

2.3 Training Approach

Approach 1

We use the single model processing with all input to all output as shown in Figure 7.

Figure 7: Workflow for approach 1.

Approach 2

We use the double model processing with Separate input to Separate output, which is q to q' and r to r' as shown in Figure 8.

Figure 8: Workflow for approach 2.

.

Approach 3

We use the double model processing with all input to q' and all input to r' as shown in Figure 9.

Figure 9: Workflow for approach 3.

.

2.4 Post Processing

We use the beam search as the post processing. Beam search is an algorithm that improves upon greedy search by expanding the search space, though not to the same extent as exhaustive search. It strikes a balance between the two approaches.

Beam search has a hyperparameter called beam width, which is denoted as k. In the first step, the algorithm selects the k words with the highest probability as candidates. At each subsequent step, it then selects the top k probability sequences among all possible combinations based on the previous output sequence. The algorithm always keeps k candidates and ultimately selects the best one from these candidates.

3 Experiment Result and Discussion

At the beginning, we designed the model as illustrated in Figure 10, with the goal of extracting the attention weight from the BERT encoder. However, we were unable to achieve this objective. Unexpectedly, though, the classification accuracy reached 85%. Subsequently, we attempted to use an LSTM for the summary task, but after training for

10 epochs, we discovered that it was unable to train effectively. This may have been due to the lack of pretraining. After that we utilized the Hugging Face T5 model and were able to successfully achieve baseline results.

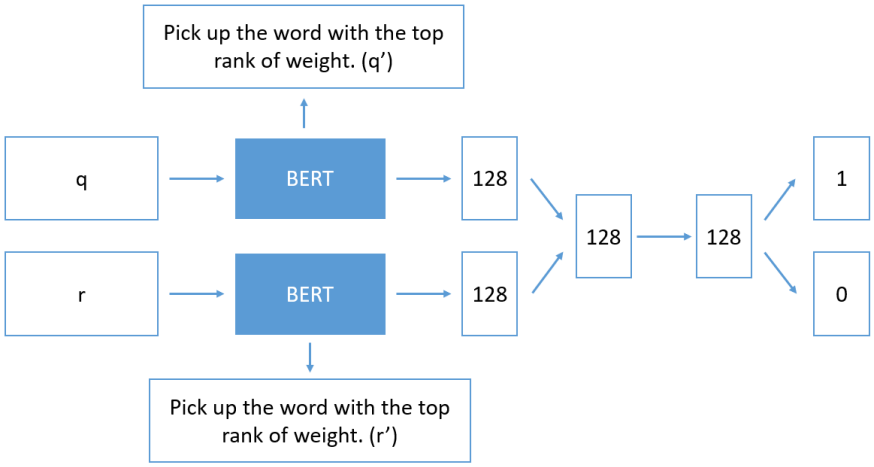


Figure 10: BERT model.

We attempt two type of pre-processing, the different T5 models, beam widths, and batch sizes. Table 2 shows the model parameters of T5-small and T5-base.

Model	Layer	Hidden	Heads	Parameters
T5-small	6	512	8	93M
T5-base	12	768	12	272M

Table 2: T5 model.

The evaluation metric for this competition is the Longest Common Subsequence (LCS), which can be found in (1). This metric calculates the overlap rate between the uploaded answer (q', r') and the ground truth $(\widehat{q'}, \widehat{r'})$. In cases where there are multiple ground truths for a given match, the answer with the highest LCS score will be selected as the final result.

$$\text{Score}((q', r'), (\widehat{q'}, \widehat{r'})) = \frac{1}{2N} \sum_{i}^{N} \max_{j} \left(\frac{|q'_i \cap \widehat{q'}_{i,j}|}{|q'_i \cup \widehat{q'}_{i,j}|} + \frac{|r'_i \cap \widehat{r'}_{i,j}|}{|r'_i \cup \widehat{r'}_{i,j}|} \right), \tag{1}$$

where N is the number of total data.

Table 3 presents the public and private scores of our experimental results with various pre-processing techniques, batch sizes, and beam search strategies. Due to hardware constraints, we employed the accumulated gradient technique to increase our batch size. As can be observed, the inputs are all combined, while the outputs are separate, resulting

in the best performance. For post-processing, a smaller number of beams generally yields better results. Our best results were obtained with a beam number of 1.

Model	Text	Summery	Batch Size	Beam	Public Score	Private Score
baseline	-	-	-	-	0.729603	0.794780
T5-small-0	all	all	8	5	0.740365	0.806056
T5-small-1	all	all	4×4	3	0.733411	0.803812
T5-small-2	all	all	4×4	5	0.741273	0.812843
T5-small-3	all	all	4×4	9	0.727152	0.802321
T5-small-4	all	all	4×4	10	0.734734	0.800842
T5-small-5	separate	separate	4×4	5	0.768011	0.809785
T5-small-6	all	separate	4×4	5	0.768519	0.812684
T5-small-7	all	separate	4×4	1	**0.801772**	**0.851903**
T5-base-1	separate	separate	4×4	5	0.731019	0.801831
T5-base-2	separate	separate	4×4	10	0.727834	0.800213
T5-base-3	all	separate	4×4	1	0.799337	0.851701
T5-base-4	all	separate	8×4	1	0.796900	0.849771

Table 3: The public score and private score of our experiment models.

We also statistics the length of our prediction. The average length of our predictions is about 13, but the average length of q' and r' was nearly 20 words in training set, so we choose longest predicted answer from the top 3 results. The ensemble method improve accuracy of 0.4% as shown in Table 3.

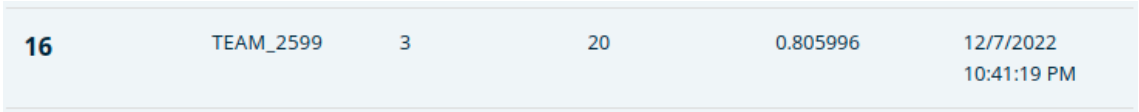
Model	Public Score	Private Score
T5-small-7 T5-base-3	**0.805996**	**0.855363**
T5-small-7 T5-base-3 T5-base-4	0.797934	0.854497

Table 4: The public score and private score of our ensemble result.

4 Conclusion

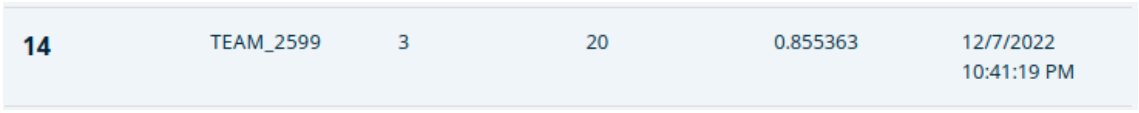
As a preliminary test, we also tested the LSTM [3] and the BERT [4]. However, both models failed to produce satisfactory results for this task. One reason for the failure of the LSTM model was its lack of pre-training, leading to insufficiently powerful embeddings even after training the model for several epochs. We trained the BERT model with separate input and a classification output, in which the larger the weight, the more important the word. However, we were unable to effectively pick out the weight in the model,

leading to this experiment's failure. Ultimately, we chose the T5 model as it is a powerful pre-trained model that only required fine-tuning. After confirming its trainability, we conducted further experiments by trying different T5 models and varying the beam width, the results of which are shown in Table 1. In the train data, we found that the average length of q' and r' was nearly 20 words, but our best answers were usually shorter than 20 words. Therefore, we employed a strategy of selecting the top 3 results from the best T5 model and choosing the longest one as the answer. Finally, we achieved a public scores of 0.805996 and a private score of 0.855363, corresponding to ranking of 16 and 14, respectively.



16	TEAM_2599	3	20	0.805996	12/7/2022 10:41:19 PM

Figure 11: Public leaderboard.



14	TEAM_2599	3	20	0.855363	12/7/2022 10:41:19 PM

Figure 12: Private leaderboard.

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] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[3] Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

A Appendix

A.1 Reproduce the best result

In this section, we demonstrate how to reproduce the best result. We release our code to GitHub repository https://github.com/Jia-Wei-Liao/Crop_Classification.

1. Setting the environment:

```
conda create --name explainable_nlp python=3.7.4
source activate explainable_nlp
pip install -r requirements.txt
```

2. Download the model weights from google drive and move into the assigned place in folder structure.

3. Rub the following commend:

```
bash reproduce.sh
```

A.2 Acknowledgements

We thank the authors of these GitHub repositories, which we use in this competition:

1. Hugging Face: https://github.com/huggingface

2. pytorch-seq2seq: https://github.com/bentrevett/pytorch-seq2seq

A.3 Operating System

We develop the code on Ubuntu 22.04. All trainings are performed on a server with a single NVIDIA 3090 GPU.

A.4 Team Information

Team Name	Public Score	Public Rank	Private Score	Private Rank
TEAM_2599	0.805996	16 / 255	0.855363	14 / 255

A.5 Contribution

Name	Student ID	Work	Contribution
Jia-Wei Liao (廖家緯)	D11922016	coding, report	33.33%
Jung-Mei Chu (朱蓉美)	D09944017	report	33.33%
Chia-Chi Huang (黃佳琪)	P10922001	survey	33.33%