Results of CSIT6000R NLP Group Project

Group No.7

1 Topic

Our group project aims to implement the framework proposed in the paper: **AQE: Argument Quadruplet Extraction via a Quad-Tagging Augmented Generative Approach** [Guo et al., ACL'23]. You can get our source code from here.

2 Dataset

The paper introduces a new dataset called *Quadruplet Argument Mining* (QAM) based on the existing IAM dataset. Specifically, the evidence type attribute is annotated to support the AQE task. You can get the QAM dataset from here.

For your information, Table 1 (Table 2 in the original paper) shows the data statistics for the QAM dataset. The whole dataset has been split into training, development and testing and the essential part of the dataset is the so-called quadruplet and it contains four kinds of information: *claim*, *evidence*, *evidence type* and *stance*. The below example means that the first sentence is a claim, the seconde sentence is an evidence for the first sentence, of which the type is explanation, and the stance of the first sentence to the topic is "support".

$$[1, 2, \text{“1”}, \text{“}\mathit{Explanation}\text{”}]$$

3 Loss Function

Two main contributions of this paper are the *argument quadruplet extraction* (AQE) task and the *quad-tagging* (QuadTAG) augmented generative approach. Figure 1 (Figure 3 in the original paper) shows the overview of the QuadTAG model. Specificall, it combines two parts: the quad-tagging

Statistics	**Train**	**Dev**	**Test**
# topics	96	52	53
# documents	639	80	82
# paragraphs	2569	326	342
# claims	2674	358	375
# pieces of evidence	6563	808	948
# quadruplets	7502	938	1098

Table 1: Data statistics for the QAM dataset.

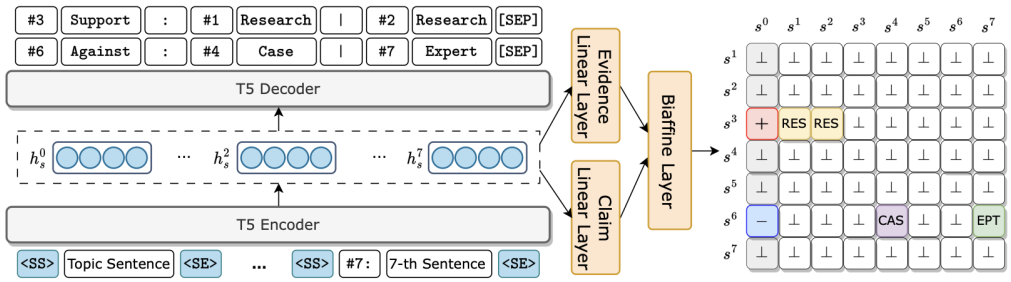

Figure 1: The overview of the QuadTAG model.

augmented part and the autoregressive part. Based on the quad-tagging augmented approach, the whole model provides a comprehensive pipeline to solve the complex task in a generative way.

Equation (1) and (2) shows the loss function for the two parts. The first loss function handles the quad-tagging part by cross entropy loss and it corresponds to the rightmost part of Figure 1, while the second loss function is a generic autoregressive loss function and it handles the encoder-decoder part of the model. Finally, the final loss function is defined as equation (3) and it provides a pipeline approach to train the model.

$$L_a = -\sum_{(i,j)\in\{N\cup P\}}\sum_{k=1}^{r} y_{ij}^k log P_\phi(\hat{y}_{ij}^k) \tag{1}$$

$$L_g = -\sum_{t=1}^{T} log P_\theta(y_t|\mathbf{H}_{enc}, y<t) \tag{2}$$

$$L = L_g + L_a \tag{3}$$

4 Training

Same as the original paper, we finetune the pre-trained T5-base model based on the QAM dataset and the final loss function. You can get the finetuned model and test results from here.

For the hyper-parameters, we directly followed the settings in the original paper, except that we reduced the **max_seq_length** parameter to half, from 2496 to 1248, because of the limitation of memory. Table 2 shows the hyper-parameters we used for training.

Besides, in order to take advantage of the validation set, we recorded the precision, recall and F1 score of the model on the validation set during training, and chose the model with largest F1 score. Figure 2 shows the training loss and validation loss during training and Figure 3 shows the change of the F1 score. Finally, we choose the model trained after 10 epochs.

According to the change of loss and F1 score, there are two "uncommon" results: **i)** At the beginning of the training process, the validation loss is lower than the training loss and maybe this is because the training loss is calculated "during" training while the validation loss is calculated "after" training; **ii)** The validation loss is overall increasing while the F1 score is overall increasing

Name	Value
pre-trained model	T5-base
batch_size	1
max_seq_length	1248
learning_rate	1e-4
num_train_epochs	10
max_generate_len	512
max_sent_num	32
max_sent_len	400
embed_dim	768
hidden_embed_dim	256

Table 2: Hyper-parameters for training.

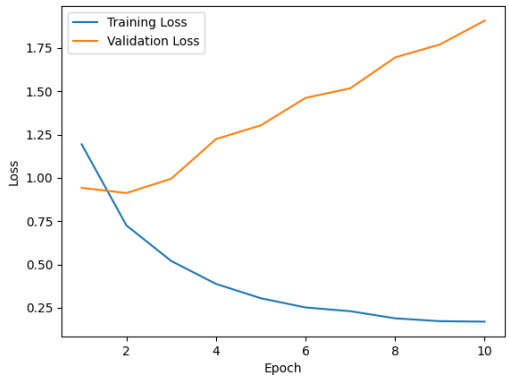

Figure 2: Training loss and validation loss.

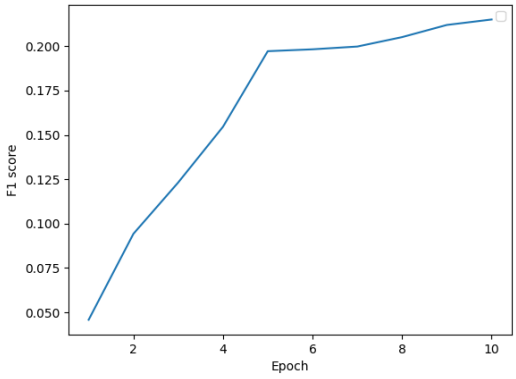

Figure 3: F1 score on the validation set.

Model	Precision	Recall	F1
QuadTAG (original paper)	24.47 ± 3.01	19.01 ± 1.53	21.39 ± 2.11
QuadTAG (our trained model)	24.49	21.22	22.74
ChatGPT with one-shot	11.28	10.92	11.10

Table 3: Experimental results for the AQE task.

too and maybe this is because the loss is calculated based on the probability while F1 score is calculated based precision and recall, which are deterministic in our setting.

5 Evaluation

Based on the original paper, we also calculate the **precision**, **recall** and **F1 score** on the test data to evaluate the model performance. Besides, we also consider applying models with different backbone architectures on the AQE task. Specifically, for the encoder-only model, such as BERT, we can try to apply the quad-tagging part but there is no generative part. For the decoder-only model, we try to prompt ChatGPT with one-shot and hopefully it can mine some relationship between sentences and topics. Below is what we use for prompting:

i) Task description: There are three components of the input: topic, sents and labels. For labels, e.g., [1, 2, "1", "Explanation"] means the 1st sentence is a "claim", and the 2nd sentence is an evidence of the 1st sentence, and the 3rd element "1" means the stance is "support" (if it's "-1", it means the stance is "against" and there are just two cases for this output), and "Explanation" means the evidence type is "Explanation" (note that the evidence type should be in [Expert, Research, Case, Explanation, Others]).

ii) Example:

"topic": "Will artificial intelligence replace humans",

"sents": ["sentence1", "sentence2", ...],

"labels": [[1, 2, "1", "Explanation"], [12, 9, "1", "Explanation"], [12, 10, "1", "Explanation"], [12, 11, "1", "Explanation"], [12, 12, "1", "Explanation"], [15, 15, "1", "Explanation"]].

iii) Prompt:

"topic": "Should we promote the use of wind energy",

"sents": ["sentence1", "sentence2", ...],

"labels": .

Table 3 shows the metric scores of these three cases. You can get the result from here.

As the result shows, we have finetuned a model which performs as well as the model from the original paper. Besides, the performance of the one-shot case is not good and maybe this is because the prompting is not good enough as it's not a generative form.

6 Conclusion

As the metric scores show, we successfully implement the AQE task and the QuadTAG approach, which implies that combining a classification-formed augmentation part, i.e., the quad-tagging, can work well with the generative part to get a better result. We can look forward to leverage this kind of augmentation in other tasks.

However, the proposed model basically rely on the encoder-decoder architecture. The future work that we could consider is to just apply the quad-tagging baseline to the encoder-only model to avoid the accumulated loss from the generative baseline. Although the encoder-only model can't provide the generative ability, we can expect a more efficient procedure in this specific task.