ADL Homework 2: Chinese News Summarization

Department: CSIE Student ID: d11922016 Name: Jia-Wei, Liao

November 8, 2023

1 Model

1.1 Model

Describe the model architecture and how it works on text summarization.

Solution.

We employed the `google/mt5-small` [1] as the model for this task. The mT5 (Multilingual Pre-trained Text-to-Text Transformer) model is an extension and improvement upon the original T5 (Text-to-Text Transfer Transformer) model [2]. The T5 model adopts an encoder-decoder architecture, where the encoder utilizes fully visible masking, and the decoder employs causal masking. mT5 further enhances this architecture by incorporating GeGLU nonlinearities [3] and undergoing pre-training on a vast amount of unlabeled data without the use of dropout. Key architectural details of this model include:

- Layer Configuration: The mT5 model consists of 8 encoder layers and 8 decoder layers.
- Attention Heads: Within each layer, it utilizes 6 attention heads.
- Feed-Forward Layer: The size of the feed-forward layer is set to 1024.
- Projection Size: The key, query, and value projections are configured to be of size 64.

Within the realm of text summarization tasks, at every time step, the decoder receives the tokens generated in the preceding steps and is assigned the responsibility of producing the next token until the "</s>" symbol is reached. Moreover, it has the flexibility to employ a multitude of methods for generating the summary. We will discuss in detail in Q3.

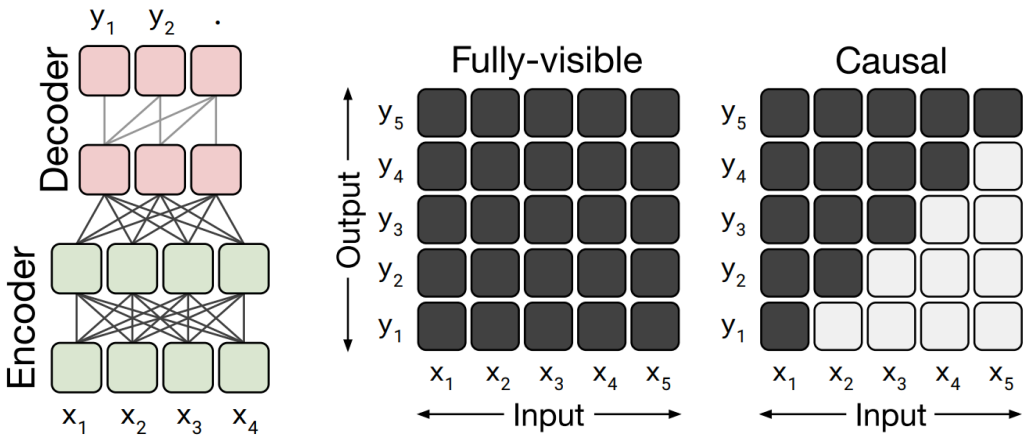

Figure 1: T5 model architecture.

1.2 Preprocessing

Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

Solution.

We utilized a tokenizer to convert both the maintext and title into vectors. In order to conserve memory, the maximum length for maintext was set to 256, while the maximum length for the title was set to 64. If the length exceeded the maximum length, truncation was applied, and for lengths less than the maximum, padding was employed.

2 Training

2.1 Hyperparameter

Describe your hyperparameter you use and how you decide it.

Solution.

The model was trained for 15 epochs with a batch size of 16 and a gradient accumulation step of 4. The cross-entropy loss function was used, and the AdamW optimizer was employed. To accelerate convergence, a larger initial learning rate of 1e-3 was used. Additionally, a linear learning rate schedule was implemented.

2.2 Learning Curves

Plot the learning curves (ROUGE versus training steps)

Solution.

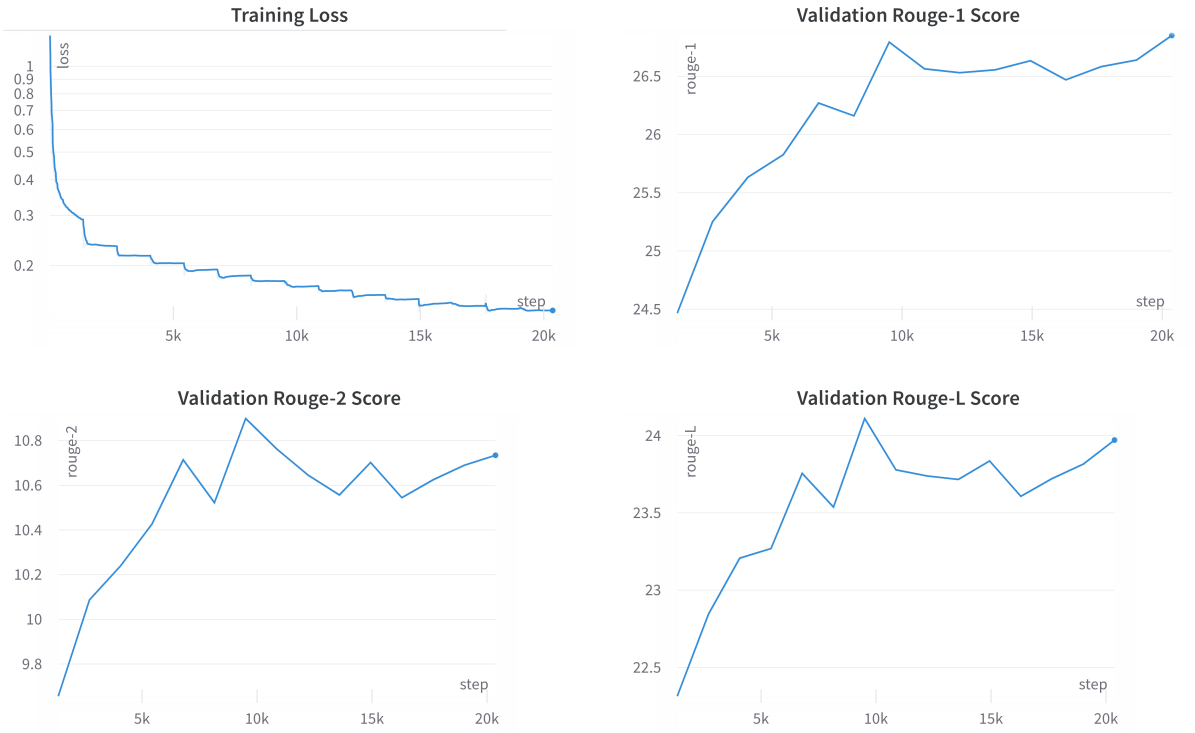

Figure 2: Loss and rouge score for the mT5 model.

3 Generation Strategies

3.1 Stratgies

Describe the detail of the following generation strategies: greedy, beam search, top-k sampling, top-p sampling, and temperature.

Solution.

Greedy Search

Greedy Search is a straightforward approach where, given the preceding words in a sequence, the next word is chosen directly based on the highest conditional probability. This method is known for its simplicity and efficiency; however, it can yield suboptimal results due to its lack of consideration for the entire sentence context. Mathematically, Greedy Search can be defined as follows:

$$w_t = \underset{w}{\operatorname{argmax}}\, P(w|w_{1:t-1}).$$

In this equation, w_t represents the word at position t in the sequence, and it is determined by selecting the word w that maximizes the conditional probability $P(w|w_{1:t-1})$ based on the preceding words in the sequence $w_{1:t-1}$.

Beam Search

Beam search [4] represents a sophisticated decoding strategy that outperforms greedy search by maintaining a beam of the most promising candidate sequences at each step. This proactive approach allows the algorithm to explore a broader portion of the search space, potentially yielding superior results, albeit at the expense of increased computational demands. By preserving the most likely hypothesis at each step, beam search effectively avoids overlooking high-probability word sequences. At each time step, it selects candidates with the highest probabilities and ultimately chooses the sequence with the highest overall probability. This approach empowers beam search to navigate the complex structure of language skillfully, resulting in text that is more fluent and coherent compared to greedy search.

Top-k Sampling

Top-k sampling [5] is a probabilistic decoding strategy that constrains the selection of the next word to the k most probable words according to the model's predictions. We define the top-k vocabulary as $V_k \subseteq V$, where $|V_k| = k$, and it is chosen to maximize the following expression:

$$\max_{V_k} \sum_{w \in V_k} P(w|w_{1:t-1}).$$

Let p^* be the value of this summation, i.e., $p^* = \sum_{w \in V_k} P(w|w_{1:t-1})$. Subsequently, we rescale the probabilities $P(w|w_{1:t-1})$ to form a new distribution denoted as $P'(w|w_{1:t-1})$, defined as follows:

$$P'(w|w_{1:t-1}) = \begin{cases} P(w|w_{1:t-1})/p^* & \text{if } x \in V_k, \\ 0 & \text{otherwise.} \end{cases}$$

Top-p Sampling

Top-p sampling [5] is another probabilistic decoding strategy that constrains the selection of the next word to a subset of the vocabulary containing at least p of the total probability mass. We define the top-p vocabulary, denoted as $V_p \subseteq V$, as the smallest set satisfying the following condition:

$$\sum_{w \in V_p} P(w|w_{1:t-1}) \geq p.$$

Let p^* be the value of this summation, i.e., $p^* = \sum_{w \in V_p} P(w|w_{1:t-1})$. Subsequently, we normalize the probabilities $P(w|w_{1:t-1})$ to create a new distribution, denoted as $P'(w|w_{1:t-1})$, using the following transformation:

$$P'(w|w_{1:t-1}) = \begin{cases} P(w|w_{1:t-1})/p^* & \text{if } x \in V_p, \\ 0 & \text{otherwise.} \end{cases}$$

Temperature

The temperature parameter [5] plays a crucial role in controlling the level of randomness during sampling. A higher temperature setting results in more random sampling outcomes,

whereas a lower temperature setting leads to more deterministic sampling. This parameter provides a means to strike a balance between fluency and creativity in generating text. The probability distribution for generating the next word, denoted as $P(w_t|w_{1:t-1})$, is defined as follows:

$$P(w_t|w_{1:t-1}) = \frac{e^{h_{t,i}/T}}{\sum e^{h_{t,i}/T}},$$

In this equation, $h_{t,i}$ represents the unnormalized log probabilities associated with each word in the vocabulary at time step t, and T is the temperature parameter which is typically constrained within the range $[0, 1)$.

3.2 Hyperparameters

Describe your hyperparameter you use and how you decide it.

- Try at least 2 settings of each strategies and compare the result.
- What is your final generation strategy? (you can combine any of them)

Solution.

Method	Number of beams	rouge-1	rouge-2	rouge-L
Greedy Search	-	25.5048	9.5331	22.6720
Beam Search	2	26.1955	10.1999	23.3702
Beam Search	3	26.5979	10.5315	23.7497
Beam Search	4	26.7983	10.7460	23.9456
Beam Search	5	**26.8510**	10.7346	23.9712
Beam Search	6	26.7812	10.7511	23.9278
Beam Search	7	26.8447	**10.7876**	**23.9821**
Beam Search	8	26.8133	10.7742	23.9337

Table 1: Hyperparameters for generation strategies.

k	rouge-1	rouge-2	rouge-L
10	26.5029	10.3898	23.5967
25	26.5361	10.3808	23.6252
50	26.5389	10.3644	23.6710
100	26.3735	10.3775	23.5287

Table 2: Top-k sampling and beam search with 5 beams.

p	rouge-1	rouge-2	rouge-L
0.4	26.0978	10.0565	23.1992
0.7	26.3562	10.3046	23.5041
0.8	26.4028	10.3806	23.5736
0.95	26.5480	10.4080	23.6213

Table 3: Top-p sampling and beam search with 5 beams.

T	rouge-1	rouge-2	rouge-L
0.7	25.6951	9.9916	22.9742
0.9	24.5000	9.3485	21.9643

Table 4: Sampling with temperature and beam search with 5 beams.

In Tables 1,2 3, and 4, it is evident that the exclusive use of beam search yields the best outcomes, and the performance improves as the number of beams increases. However, this enhancement comes at the cost of increased memory usage and longer inference times. Therefore, we adopt beam search with five beams for our final results to balance performance with computational efficiency during inference.

4 Applied RL on Summarization (Bonus)

4.1 Algorithm

Describe your RL algorithms, reward function, and hyperparameters.

Solution.

We adopt the Policy Gradient Algorithm, and the reward function is defined as follows:

$$R_\theta = \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\tau) r(\tau)$$

where τ is trajectory, $p_\theta(\tau)$ is probability of trajectory, N is number of batch data, and r is reward.

We first trained the model for one epoch using supervised learning, followed by 15 epochs of reinforcement learning, with a batch size of 16 and a gradient accumulation step of 4. We chose the cross-entropy loss function and employed the AdamW optimizer. The initial learning rate was set to 1e-3. and a linear learning rate schedule was implemented.

4.2 Compare to Supervised Learning

Observe the loss, ROUGE score and output texts, what differences can you find?

Solution.

	rouge-1	rouge-2	rouge-L
Baseline	22.0	8.5	20.5
Supervised Learning	26.8510	10.7876	23.9712
Reinforcement Learning	26.5740	10.5980	23.7600

Table 5: Performance evaluation

In Table 5, both Supervised Learning and Reinforcement Learning outperform the Baseline approach across multiple evaluation metrics, specifically, rouge-1, rouge-2, and rouge-L.

Notably, Supervised Learning achieves higher scores in rouge-1 (26.8510), rouge-2 (10.7876), and rouge-L (23.9712) compared to the Baseline method, while Reinforcement Learning also exhibits improved performance with rouge-1 (26.5740), rouge-2 (10.5980), and rouge-L (23.7600) scores.

References

[1] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[2] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[3] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[4] Alexander M Rush, SEAS Harvard, Sumit Chopra, and Jason Weston. A neural attention model for sentence summarization. In *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*, 2017.

[5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Appendix

All training sessions were conducted on a PC running Ubuntu 20.04.2 with a 12th Gen Intel(R) Core(TM) i7-12700 CPU, equipped with a single NVIDIA GeForce RTX 4090 GPU featuring 24 GB of dedicated memory.

Environment

To set the environment, you can run this command:

```
pip install -r requirements.txt
```

Download

To download the datasets and model checkpoint, you can run the commad:

```
bash ./download.sh
```

Reproduce

To reproduce the result of inference, you can run the commad:

```
bash ./run.sh <data file> <output file>
```

Training

To train the summarization model, you can run the commad:

```
python train.py
```