Chinese Text Summarization

第十組:
何存益 110550165
周冠辰 110550089
文玠敦 110550032
范均宏 110550083

Introduction

Our Problem

In this fast-paced world, people may have limited time for reading. Using AI to summarize articles can save a great deal of time, therefore we choose this topic.

Our problem is using two different methods, extractive method and abstractive method, to summarize Chinese articles. We will compare the result of these two methods and discuss for their performance.

Introduction

Extractive Method

Selects keywords and key sentences from the original text to create the summaries by copying and rearranging passages from the original text.

focuses on selecting and preserving existing information

Abstractive Method

Aims to generate concise and coherent summaries by understanding the content of the text and then generate new sentences.

emphasizes the generation of novel and creative summaries

Introduction

Old school extractive method

- Low computational cost
- Ensuring readability
- Imprecise summary

Cutting edge abstractive method

- Using machine learning method to generate the summary
- Time consuming for training process
- If readability is ensuring, the generated summary will be quite similar to what human would write

Introduction

extractive method	abstractive method
sentence scoring method	Fine-tuning the pre-trained mt5 model

Related work

Text summarization is an important and extensively researched task in the field of Natural Language Processing (NLP)

Assessing sentence scoring techniques for extractive text summarization

A Survey on Extractive Text Summarization

Attention Is All You Need

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Related work

Challenges

Previous studies on automatic text summarization have primarily focused on English text.

We would like to know if these methods can perform similarly in Chinese and compare them.

Dataset

News

- The news in https://www.ettoday.net
- Using Python with BeautifulSoup4 to do webcrawling
- About 38000 news from 2020 to 2022

  (80% for training, 10% for evaluate and test respectively)
- Attribute: id, title, url, article

Dataset

{"id": 1,
"title": "才剛結婚！黑鷹上尉副駕駛殉職　妻英國出差聽死訊崩潰",
"url": "https://www.ettoday.net/news/20200102/1616296.htm",
"article": "綜合報導空軍一架黑鷹直升機2日在新北市烏來山區迫降，造成8死5傷的意外，震驚台灣社會，正副駕駛名單曝光後，更有人指出，上尉副駕駛劉鎮富有著500多小時的飛行經驗，2018年才參加空軍聯合婚禮，與太太永結連理，沒想到竟然出了意外，讓家人相當悲痛。劉姓副駕駛的太太得知消息，也悲痛在臉書發文，「請大家給我一點時間，謝謝。」劉鎮富家住台南市永康區，父親較早離世，平時與母親相依，有空就會回家陪媽媽，或是跟著太太到安南區的岳父家中。劉鎮富死亡的消息傳出後，讓劉母與岳父母感到相當不可置信，趕忙搭車北上，處理兒子的後事。據了解，劉鎮富與專攻音樂的妻子2015年開始交往，兩人2018年參加空軍聯合婚禮，在眾人的祝福下永結連理，當時劉鎮富還興奮指出，太太當時特地到場看他穿上飛行服，讓他印象相當深刻，「老婆是我人生最重要的責任與牽手，未來希望彼此包容、相互扶持，共創美好。」事件發生的第一時間，劉鎮富的妻子與工作的樂團正在英國倫敦出差，聽到消息後感到相當崩潰，開始聯絡航空公司，爭取在第一時間回來，陪丈夫走最後一段路。她也在臉書上悲痛發文，「請大家給我一點時間，謝謝」，讓不少人紛紛湧入打氣。台南市長黃偉哲得知消息，立刻指派區長前往探視家屬，同時也要求持續給予關心、協助。他提到，國家痛失英才，敬上最大哀悼，市府將盡力協助家屬，處理後續事宜。►滑新聞還能亮起來，偷偷分享皮膚發光秘密！"}

How we evaluate results

ROUGE

ROUGE is a evaluation metric commonly used in NLP and text summarization tasks. We will employ several metrics:

- ROUGE-1
- ROUGE-2
- ROUGE-L

$$Rouge - N = \frac{n - gram}{len(x)}$$

$$Rouge - L = \frac{(1 + \beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}}$$

How we evaluate results

$$Rouge - N = \frac{n - gram}{len(x)}$$

- X is the generated summarize
- n-gram represents the number of consecutive n words that match the reference summary

$$Rouge - L = \frac{(1 + \beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}}$$

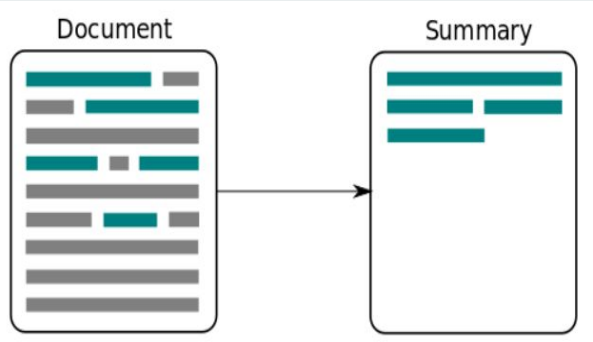
$$R_{LCS} = \frac{LCS(C,S)}{len(S)} \qquad P_{LCS} = \frac{LCS(C,S)}{len(C)}$$

- S is the reference summarize
- C is the generated summarize
- Generally, beta is set to be a large value, so Rouge-L is primarily focuses on RLCS

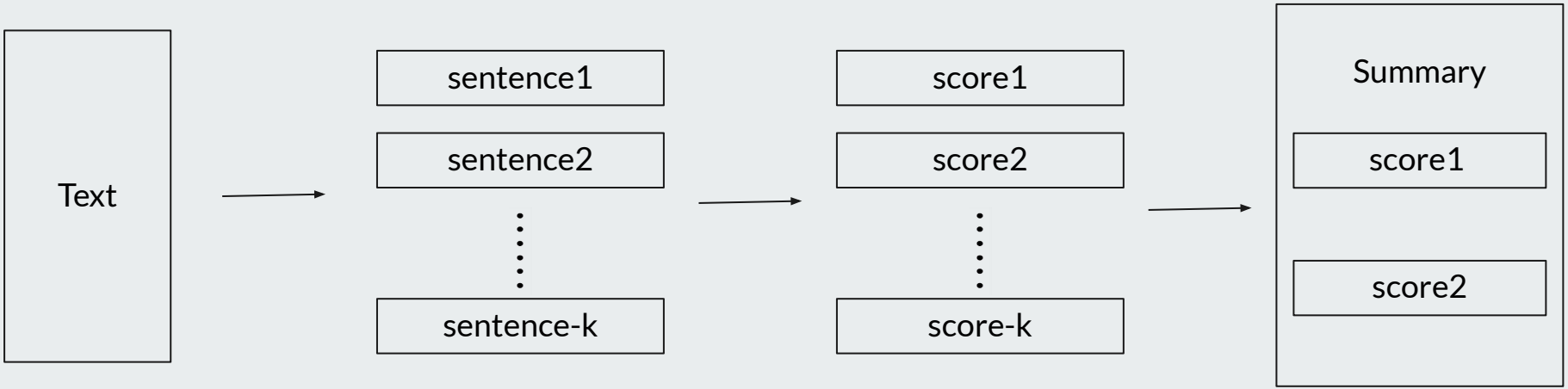
Baseline - Extractive Method

Extractive Method - Sentence Scoring Method

- Split the text into sentences
- Tokenize sentences with CKIPTagger
- Extract high-frequency words
- Compute scores for each sentence
- Select the highest-scoring sentences as the summary



Baseline - Extractive Method



Baseline - Extractive Method

```
"""
The function calculates scores for each sentence based on the top-n words.
"""
def score_sentences(sentences, topn_words):
    scores = []
    sentence_idx = -1
    slist = []

    for s in sentences:     # Tokenizes sentences.
        temp = ws(s.split(" "))
        slist.append(temp[0])

    for s in slist:
        sentence_idx += 1
        word_idx = []
        for w in topn_words:    # Store the indices of the top-n words within the sentence.
            try:
                word_idx.append(s.index(w))
            except ValueError:      # w is not found
                pass
        word_idx.sort()
```

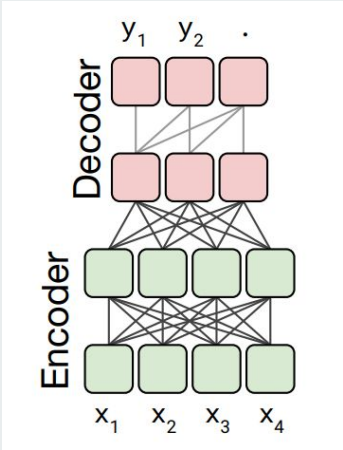
Baseline - Extractive Method

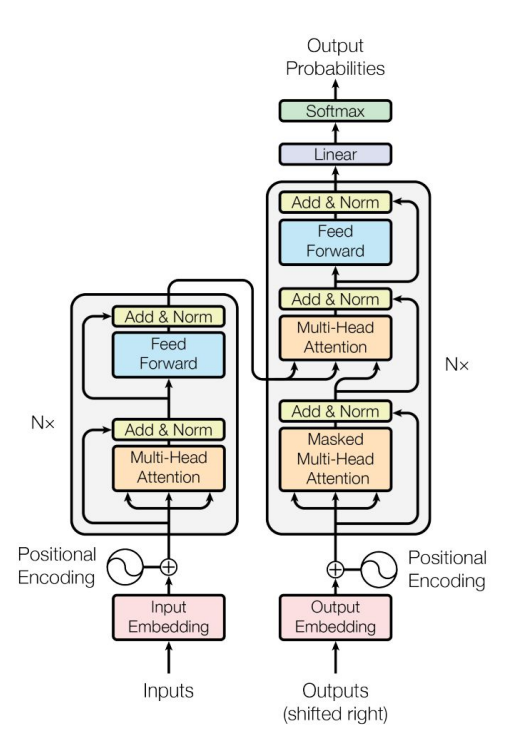
```
    clusters = []
    cluster = [word_idx[0]]
    i = 1
    # The code iterates over the word_idx and forms clusters.
    # If the distance between two word_idx is less than CLUSTER_THRESHOLD, view them as the same cluster.
    while i < len(word_idx):
        CLUSTER_THRESHOLD = 4
        if word_idx[i] - word_idx[i - 1] < CLUSTER_THRESHOLD:
            cluster.append(word_idx[i])
        else:
            clusters.append(cluster[:])
            cluster = [word_idx[i]]
        i += 1
    clusters.append(cluster)
    # For each cluster, the code calculates a score based on the number of significant words and the total number of words.
    # The score is calculated from formula: 1.0 * significant_words_in_cluster^2 / total_words_in_cluster.
    max_cluster_score = 0
    for c in clusters:
        significant_words_in_cluster = len(c)
        total_words_in_cluster = c[-1] - c[0] + 1   # distance between the first and the last top-n word
        score = 1.0 * significant_words_in_cluster * significant_words_in_cluster / total_words_in_cluster
        if score > max_cluster_score:
            max_cluster_score = score
    scores.append((sentence_idx, max_cluster_score))
return scores
```

Main Approach - Abstractive Method

Transformer Model

A neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence.





Main Approach - Abstractive Method

Fine-tuning the pre-trained transformer Model

Why using pre-trained model

- Time and resource efficiency
- Good performance on text generalization
- GPT-3, BERT, T5 etc.

How to Fine-tuning

- Just as how normal neural network training
- Supervised learning with our dataset
- Computing loss through the predict summary and the original summary
- Gradient descent

Main Approach - Abstractive Method

mt5 model

- Standard encoder-decoder transformer - the same architecture as T5 model
- Pre-training on multilingual dataset: mc4 (101 different languages, 27 TB dataset)
- Pre-training process excluding any supervised learning
- We use mt5-small

Main Approach - Abstractive Method

```
for epoch in tqdm(range(epochs)):
    accum_loss = 0
    for step, data in tqdm(enumerate(train_dataloader, 1)):
        model.train()
        data = {k: v for k, v in data.items() if k != "indices"}
        outputs = model(**data)
        ml_loss = outputs.loss
        #
        loss = ml_loss
        if len(train_dataloader) % 16 != 0 \
                and len(train_dataloader) - step < 16:
            loss = loss / (len(train_dataloader) % 16)
        else:
            loss = loss / 16
        accelerator.backward(loss)
        if step % 16 == 0 or step == len(train_dataloader):
            clip_grad_norm_(model.parameters(), max_norm=5)
            optimizer.step()
            lr_scheduler.step()
            optimizer.zero_grad()
```

```
loss = None
if labels is not None:
    loss_fct = CrossEntropyLoss(ignore_index=-100)
    loss = loss_fct(lm_logits.view(-1, lm_logits.size(-1)), labels.view(-1))
```

Hyperparameters

- batch size: 4
- accumulate steps: 16
- epoch: 5
- learning rate: 5e-5
- optimizer: Adafactor
  with weight decay = 1e-2
- scheduler: Linear decay
  with warm up steps = 90

Main Approach - Abstractive Method

Furthur approach -
Applying reinforcement learning on fine-tuning process

- Using ROUGE scores as reward
- Using sample strategy to generate predict topic
- Using greedy strategy to generate baseline topic
- Refered to A DEEP REINFORCED MODEL FOR ABSTRACTIVE SUMMARIZATION

Main Approach - Abstractive Method

self-critical policy gradient

$$L_{rl} = (r(y^b) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \dots, y_{t-1}^s, x)$$

$$L_{rl} = (r(y^b) - r(y^s)) \sum_{t=1}^{n'} -score(y_t^s)$$

Mixed trained objective function

$$L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{ml}$$

Main Approach - Abstractive Method

```python
def compute_rl_loss(data, logits, model, tokenizer, accelerator, device, compute_metrice):
    # Using model with greedy strategy to generate the summary
    baseline_tokens = accelerator.unwrap_model(model).generate(
        data["input_ids"],
        attention_mask=data["attention_mask"],
        max_length=64,
        num_beams=1
    )

    # Using model with sampling strategy to generate the summary
    sampled_tokens = accelerator.unwrap_model(model).generate(
        data["input_ids"],
        attention_mask=data["attention_mask"],
        max_length=64,
        do_sample=True,
        num_beams=1,
        top_k=0,
        top_p=0.2,
        temperature=0.5,
        output_scores=True,
        return_dict_in_generate=True,
    )
    transition_scores = model.compute_transition_scores(
        sampled_tokens.sequences, sampled_tokens.scores, normalize_logits=False
    )
    scores = []
    for i in transition_scores:
        sum = 0.0
        for j in i:
            if j > -10000000:
                sum += j
        scores.append(sum/10)
    scores = torch.tensor(scores).to(device)
    sampled_tokens = sampled_tokens.sequences
```

```python
    # compute the score of baseline summary
    baseline_scores = []
    for pred, ref in zip(baseline_tokens, data['labels']):
        pred = torch.unsqueeze(pred, 0)
        ref = torch.unsqueeze(ref, 0)
        baseline_scores.append(compute_metrice([pred, ref]))
    baseline_rewards = torch.FloatTensor([scores["rougeL"] \
                        for scores in baseline_scores]).to(device)
    # compute the score of sample summary
    sampled_scores = []
    for pred, ref in zip(sampled_tokens, data['labels']):
        pred = torch.unsqueeze(pred, 0)
        ref = torch.unsqueeze(ref, 0)
        sampled_scores.append(compute_metrice([pred, ref]))
    sampled_rewards = torch.FloatTensor([scores["rougeL"]  \
                        for scores in sampled_scores]).to(device)
    diff_rewards = (torch.Tensor(baseline_rewards) - torch.Tensor(sampled_rewards)).to(device)
    rl_loss = -(diff_rewards * scores).mean()
    return rl_loss
```

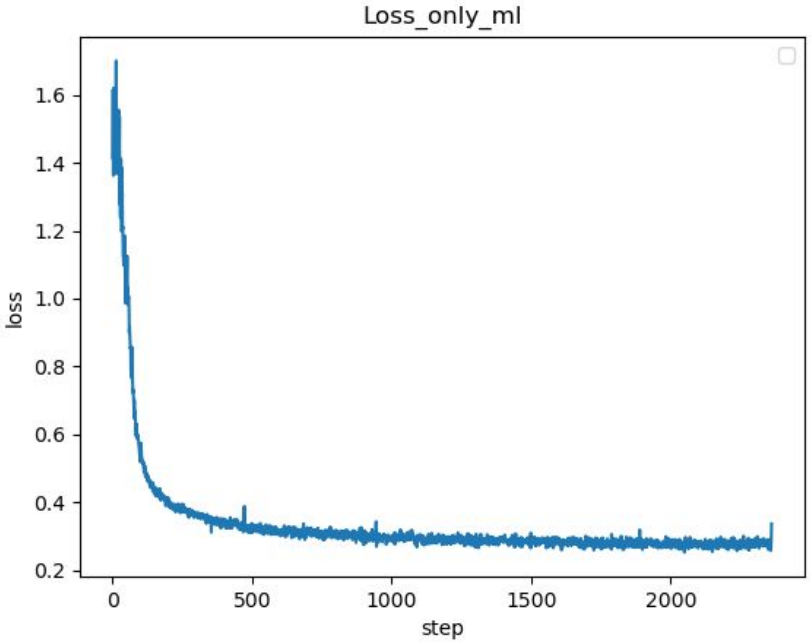
```python
outputs = model(**data)
rl_ratio = 0.1*epoch
if rl_ratio > 0:
    rl_loss = compute_rl_loss(data, outputs.logits, model,
                              mt5_tokenizer, accelerator, device, compute_metrices)
else:
    rl_loss = torch.tensor(0)
ml_loss = outputs.loss
loss = rl_loss * rl_ratio + ml_loss * (1 - rl_ratio)
```

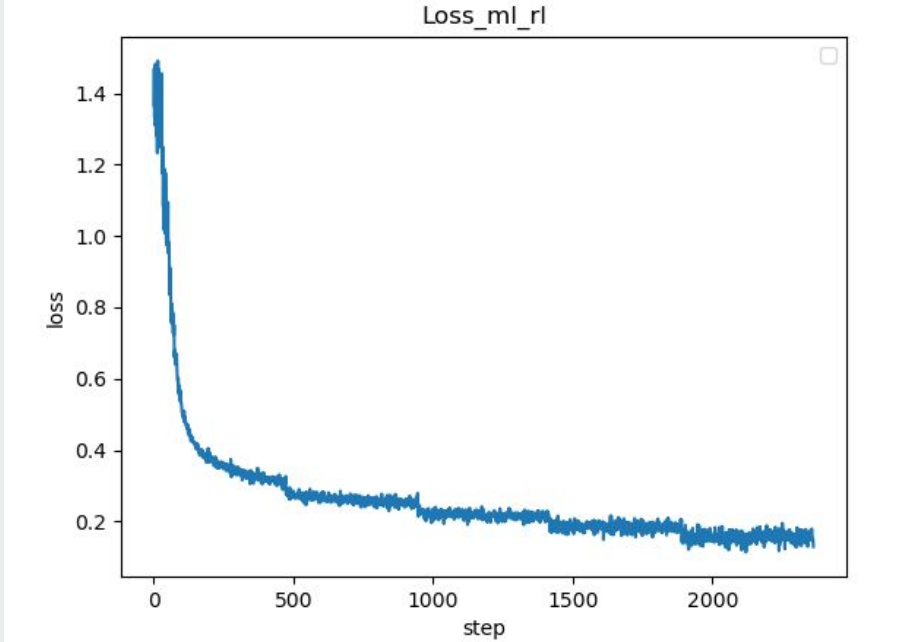
Evaluation Metric - Rouge

Average ROUGE scores of test dataset

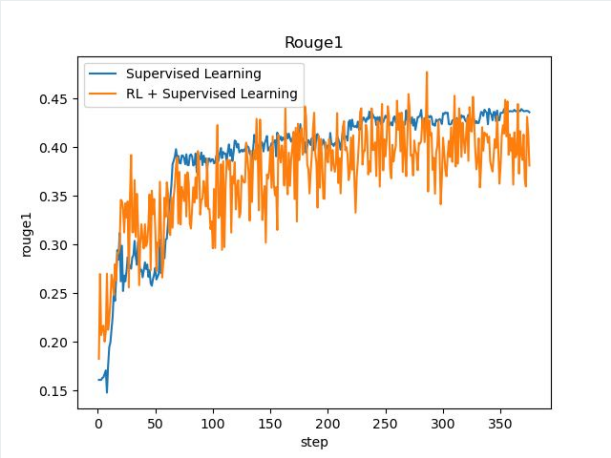
	ROUGE-1	ROUGE-2	ROUGE-L
Baseline - sentence scoring method	0.246016	0.105189	0.214126
mt5 model without fine-tuning	0.140646	0.064830	0.138213
Supervised Learning	0.389000	0.194737	0.343403
Reinforcement Learning + Supervised Learning	0.400387	0.198889	0.345834

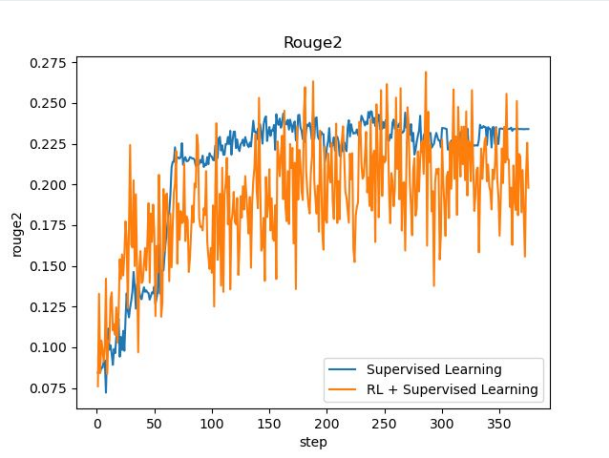
Result & Analysis - Loss Curve

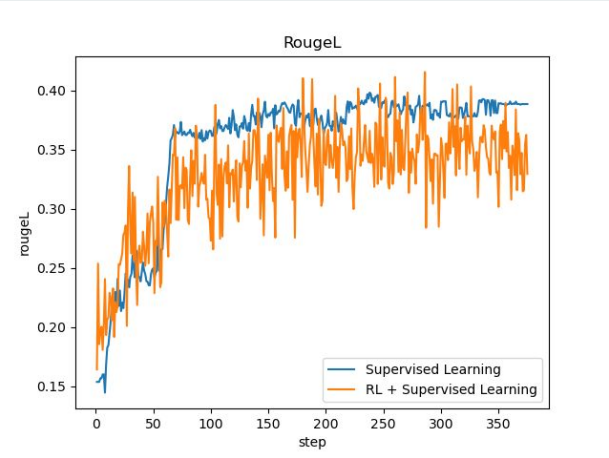




Result & Analysis - Rouge Curve







Result & Analysis - Test Data

Original Title	碧昂絲拒喝金球獎高級酒！自備破萬香檳「真實原因曝」網 :有生意頭腦
Extractive Method	綜合報導「第77屆金球獎頒獎典禮」台灣時間6日上午舉辦，西洋流行天后碧昂絲（Beyonce）和饒舌歌手老公傑斯（Jay-Z）在頒獎典禮進行一半時現身
ML Method	碧昂絲「自備高級香檳」:有錢就是任性
ML + RL	碧昂絲還自備高級香檳!網傻眼:有錢就是任性

Result & Analysis - Test Data

Original Title	免簽入境馬來西亞需備足現金 !外交部稱已實施4年:非針對台灣
Extractive Method	綜合報導中華民國護照現階段享有 168國免簽，並已於2015年9月15日公告實施
ML Method	國人「免簽方式入境吉隆坡機場」遭海關攜帶現金不足為由遣返
ML + RL	國人「免簽方式入境吉隆坡機場」遭海關以攜帶現金不足為由遣返

Result & Analysis - Test Data

Original Title	倒「特製高湯」進關東煮!便利商店工讀生引眾怒 網截圖檢舉...GG了
Extractive Method	韓國的魚糕是國民美食之一，網友看完後紛紛留言怒罵。
ML Method	快訊/便利商店「特製高湯」倒進鍋 內
ML + RL	快訊/便利商店「特製高湯」倒進鍋 內

Result & Analysis

Analysis

- Abstractive methods always generate fluent sentences, but the generated titles sometimes do not correspond to the original articles.
- Extractive methods with our fine-tuned model can find the important information in the article and construct the sentence, but they are often poorly structured.
- Applying RL during the fine-tuning process can slightly improve the performance of the generated titles; however, the learning curve of using the RL method is quite fluctuated. This indicates that more time is needed for training the model, or adjustments to the loss function and hyperparameters are required.

Result & Analysis

Limitation

- We use news headlines to train our model, so if the original titles have poor quality, the resulting outcomes will not be ideal.
- Applying reinforcement learning to a fine-tuning model is quite time-consuming because the evaluation of reward progress cannot be done by the GPU. Training for five epochs requires a whole day.

Result & Analysis

Practical application

- Help people quickly browse through the content of an article within a shorter period of time
- Enhancing news headlines for accurate representation of article content, avoiding reader misinterpretation

Information

Github link:

https://github.com/kevinchou0518/2023-NYCU-AI-Final-Project-Team-10

Youtube link:

https://youtu.be/UFe0m9H5UbE

Reference:

Assessing sentence scoring techniques for extractive text summarization

A Survey on Extractive Text Summarization

Attention Is All You Need

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

A DEEP REINFORCED MODEL FOR ABSTRACTIVE SUMMARIZATION

Information

Reference:

https://www.ettoday.net

https://github.com/ckiplab/ckiptagger

https://huggingface.co/docs/transformers/model_doc/mt5

https://github.com/huggingface/transformers/tree/main/examples%2Fpytorch%2Fsummarization

https://towardsdatascience.com/introduction-to-text-summarization-with-rouge-scores-84140c64b471

Information

Contribution of each member:

何存益 110550165:slide,extrative method,video	**20%**
周冠辰 110550089:slide,abstractive method,web crawling,video	**40%**
文玠敦 110550032:slide,extrative method,video	**20%**
范均宏 110550083:slide,extrative method,video	**20%**