08

实战演练之文本摘要

（1）文本摘要介绍

（2）基于Transformers的解决方案

（3）代码实战演练，基于T5模型

（4）代码实战演练，基于GLM模型

文本摘要介绍

文本摘要简介

- **什么是文本摘要任务**
  - 文本摘要（Text Summarization）任务的输入是长的文本文档，任务的目标是将较长的文本转换成简短的摘要，一般来说，生成的简短摘要必须要信息量充足、能够覆盖原文的主要内容。
  - 根据输入文档的数量划分，可以将摘要任务划分为单文档摘要和多文档摘要
  - 根据输入和输出的语言划分，可以将摘要任务划分为单语言摘要、跨语言摘要、多语言摘要
  - 本次课程的关注内容为单文档单语言摘要。

文本摘要介绍

文本摘要简介

- **什么是文本摘要任务**
  - 文档
    - 某卖出自家的马自达后,又使用备用车钥匙将车盗走。因涉嫌盗窃,今天上午,肖某在大兴法院受审。肖某是本市丰台人,大学毕业,无业。检方指控,2013年5月,肖某在大兴区某公司院内,使用其保留的车钥匙将此前卖给被害人的一辆马自达轿车(经鉴定价值11.5元),还有车内价值800元的加油卡、现金600元及台球杆3根及1副太阳镜一并盗走。上午9时,肖某被带进法庭。庭审中,肖某对指控没有异议。肖某的辩护人向法庭提交了一份被告人母亲写的家庭情况说明、被告人母亲的医疗费票据及诊断证明,说明其家庭生活困难,且与被害人达成和解,请求法院轻判。公诉人表示,因被告人与被害人已和解,基于被告人母亲的情况,同意法庭综合考虑量刑。最后陈述阶段,肖某说,“我因为法律意识淡薄,给社会和被害人带来了危害,经过这段时间的改造和反省,我认识到了这一错误。我会尽力补偿被害人的损失,希望法庭给我一次机会,让我用实际行动弥补对家庭的创伤。”该案没有当庭判决。
  - 摘要（标题）
    - 北京大学生卖掉自家私家车,又用备用钥匙偷回来,一并顺走车内现金600元;因家庭困难,法院同意综合量刑。

文本摘要简介

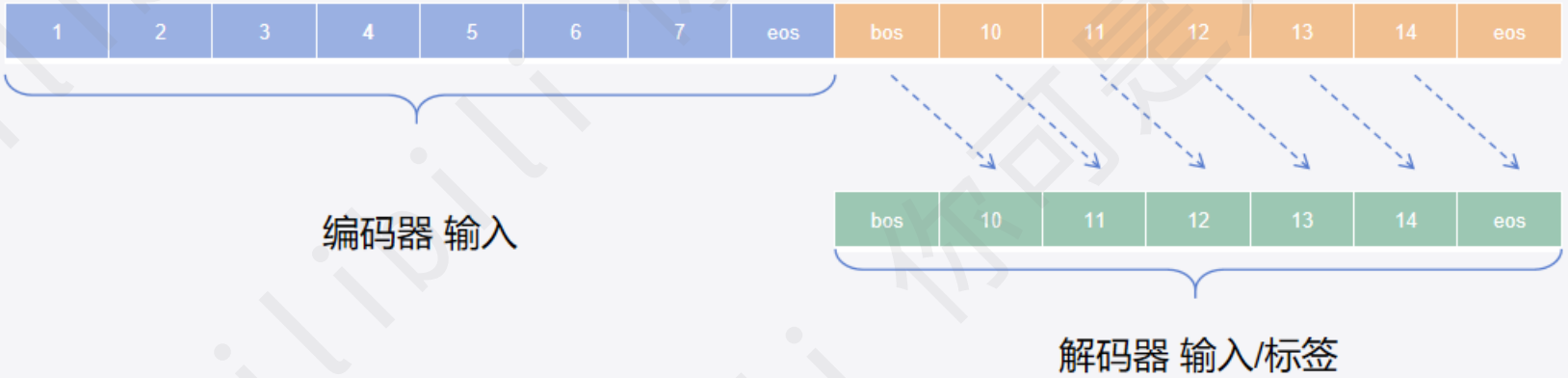
- **评价指标**
  - Rouge
    - Rouge-1、Rouge-2、Rouge-L
    - 分别基于1-gram、2-gram、LCS
  - 示例

原始文本	1-gram	2-gram
今天不错	今 天 不 错	今天 天不 不错
今天太阳不错	今 天 太 阳 不 错	今天 天太 太阳 阳不 不错

    - Rouge-1 P = 4/4， R = 4/6，F = 2*P*R / (P + R)
    - Rouge-2 P = 2/3， R = 2/5，F = 2*P*R / (P + R)
    - Rouge-L P = 4/4， R = 4/6，F = 2*P*R / (P + R)

序列到序列模型

- **数据处理与模型原理**
  - 数据处理
    - input和labels分开处理，labels的最后一定是eos_token
    - labels不仅是标签，还是解码器的输入



基于Transformers的解决方案介绍

序列到序列模型

- **数据处理与模型原理**
  - XXForConditionalGeneration

```python
def __init__(self, config: T5Config):
    super().__init__(config)
    self.model_dim = config.d_model

    self.shared = nn.Embedding(config.vocab_size, config.d_model)

    encoder_config = copy.deepcopy(config)
    encoder_config.is_decoder = False
    encoder_config.use_cache = False
    encoder_config.is_encoder_decoder = False
    self.encoder = T5Stack(encoder_config, self.shared)

    decoder_config = copy.deepcopy(config)
    decoder_config.is_decoder = True
    decoder_config.is_encoder_decoder = False
    decoder_config.num_layers = config.num_decoder_layers
    self.decoder = T5Stack(decoder_config, self.shared)

    self.lm_head = nn.Linear(config.d_model, config.vocab_size, bias=False)

    # Initialize weights and apply final processing
    self.post_init()

    # Model parallel
    self.model_parallel = False
    self.device_map = None
```

基于Transformers的解决方案介绍

序列到序列模型

- **数据处理与模型原理**
  - XXForConditionalGeneration

```
# Encode if needed (training, first prediction pass)
if encoder_outputs is None:
    # Convert encoder inputs in embeddings if needed
    encoder_outputs = self.encoder(
        input_ids=input_ids,
        attention_mask=attention_mask,
        inputs_embeds=inputs_embeds,
        head_mask=head_mask,
        output_attentions=output_attentions,
        output_hidden_states=output_hidden_states,
        return_dict=return_dict,
    )
elif return_dict and not isinstance(encoder_outputs, BaseModelOutput):
    encoder_outputs = BaseModelOutput(
        last_hidden_state=encoder_outputs[0],
        hidden_states=encoder_outputs[1] if len(encoder_outputs) > 1 else None,
        attentions=encoder_outputs[2] if len(encoder_outputs) > 2 else None,
    )

hidden_states = encoder_outputs[0]
```

基于Transformers的解决方案介绍

序列到序列模型

- **数据处理与模型原理**
  - XXForConditionalGeneration

```
# Decode
decoder_outputs = self.decoder(
    input_ids=decoder_input_ids,
    attention_mask=decoder_attention_mask,
    inputs_embeds=decoder_inputs_embeds,
    past_key_values=past_key_values,
    encoder_hidden_states=hidden_states,
    encoder_attention_mask=attention_mask,
    head_mask=decoder_head_mask,
    cross_attn_head_mask=cross_attn_head_mask,
    use_cache=use_cache,
    output_attentions=output_attentions,
    output_hidden_states=output_hidden_states,
    return_dict=return_dict,
)

sequence_output = decoder_outputs[0]
```

基于Transformers的解决方案介绍

序列到序列模型

- **数据处理与模型原理**
  - XXForConditionalGeneration

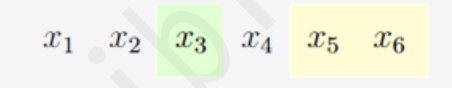
```
lm_logits = self.lm_head(sequence_output)

loss = None
if labels is not None:
    loss_fct = CrossEntropyLoss(ignore_index=-100)
    # move labels to correct device to enable PP
    labels = labels.to(lm_logits.device)
    loss = loss_fct(lm_logits.view(-1, lm_logits.size(-1)), labels.view(-1))
```

代码实战演练（基于T5模型）

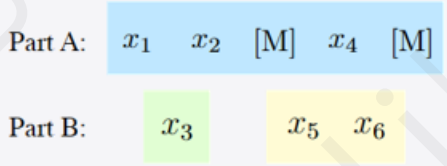
- **数据集**
  - https://huggingface.co/datasets/supremezxc/nlpcc_2017
  - 新闻标题生成
- **预训练模型**
  - Langboat/mengzi-t5-base
- **依赖库**
  - pip install rouge-chinese

前缀语言模型

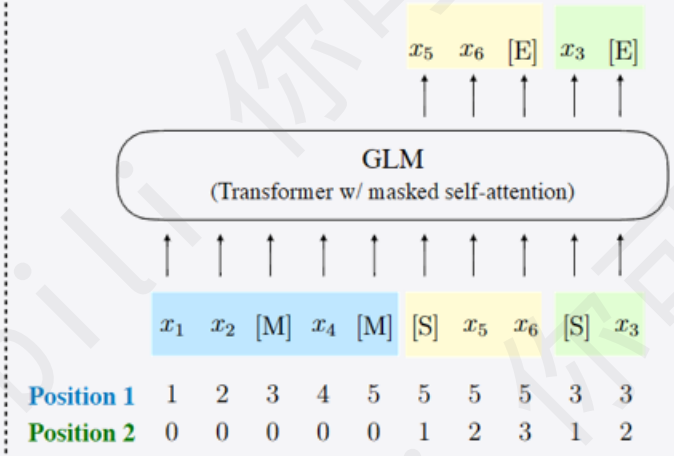
- **数据处理与模型原理**
  - 模型原理
    - 只使用编码器，借助注意力掩码实现编码器解码器的效果，只计算目标部分损失



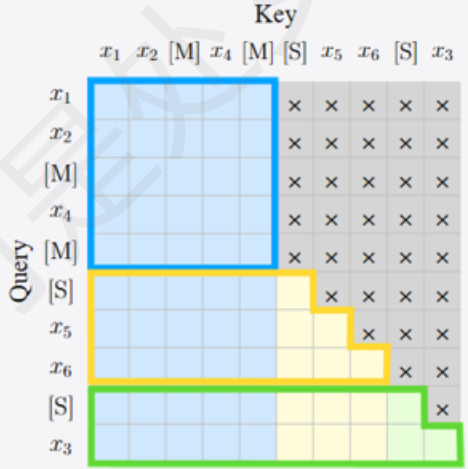
(a) Sample spans from the input text



(b) Divide the input into Part A / Part B



(c) Generate the Part B spans autoregressively

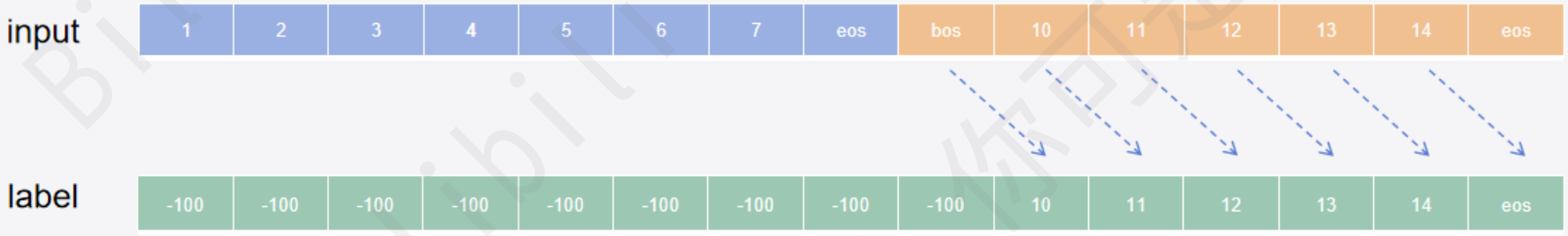


(d) Self-attention mask

基于Transformers的解决方案介绍

前缀语言模型

- **数据处理与模型原理**
  - 模型原理
    - 只使用解码器，借助注意力掩码实现编码器解码器的效果，只计算目标部分损失
  - 数据处理
    - input和labels合并在一起处理，labels的最后一定是eos_token



代码实战演练（基于GLM模型）

- **数据集**
  - https://huggingface.co/datasets/supremezxc/nlpcc_2017
  - 新闻标题生成
- **预训练模型**
  - THUDM/glm-large-chinese
- **依赖库**
  - pip install rouge-chinese