04 基础组件之Model

（1）Model 简介

（2）Model Head

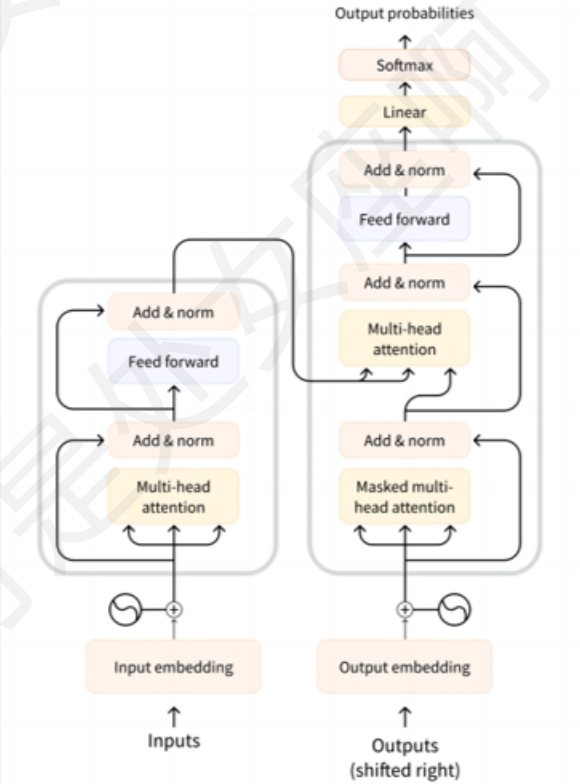
（3）Model 基本使用方法

（4）模型微调代码实例

基础组件之Model

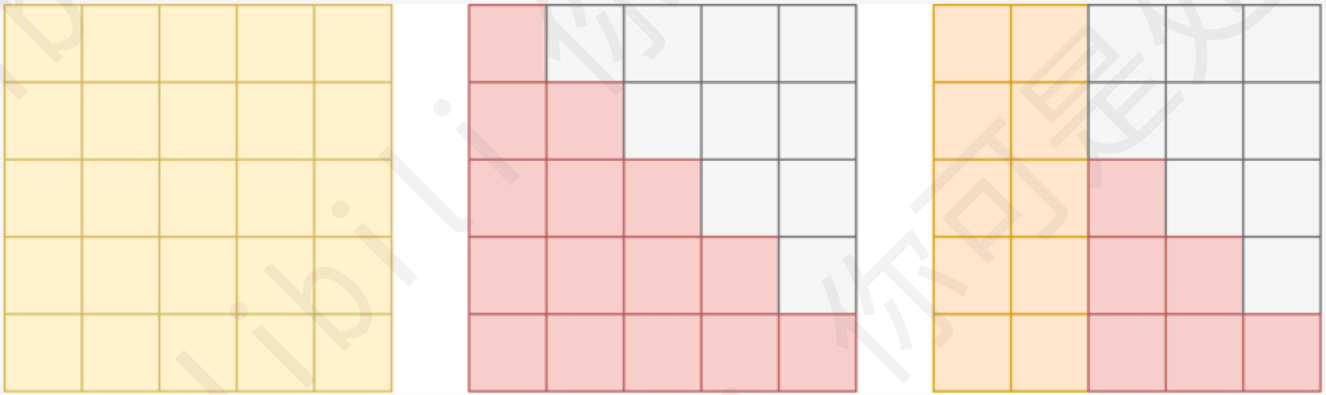
Model简介

- **Transformer**
  - 原始的Transformer为编码器（Encoder）、解码器（Decoder）模型
  - Encoder部分接收输入并构建其完整特征表示，Decoder部分使用Encoder的编码结果以及其他的输入生成目标序列
  - 无论是编码器还是解码器，均由多个TransformerBlock堆叠而成
  - TransformerBlock由注意力机制（Attention）和FFN组成
- **注意力机制**
  - 注意力机制的使用是Transformer的一个核心特性，在计算当前词的特征表示时，可以通过注意力机制有选择性的告诉模型要使用哪些上下文



Model简介

- **模型类型**
  - 编码器模型：自编码模型，使用Encoder，拥有双向的注意力机制，即计算每一个词的特征时都看到完整上下文
  - 解码器模型：自回归模型，使用Decoder，拥有单向的注意力机制，即计算每一个词的特征时都只能看到上文，无法看到下文
  - 编码器解码器模型：序列到序列模型，使用Encoder+Decoder，Encoder部分使用双向的注意力，Decoder部分使用单向注意力



编码器模型　　解码器模型　　编码器解码器模型

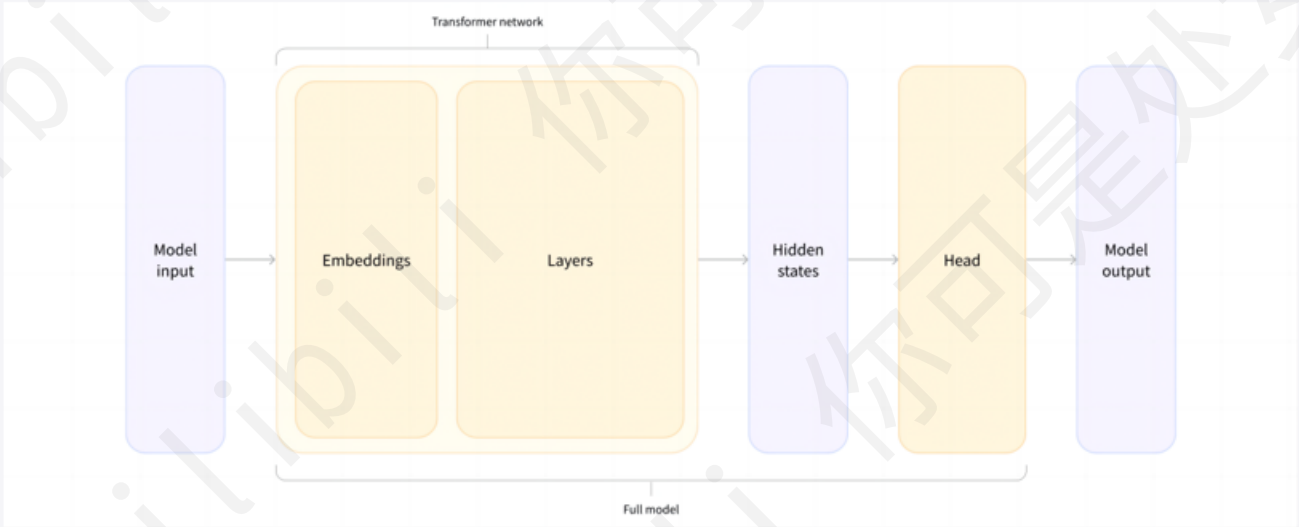
Model简介

- **模型类型**
  - 编码器模型：自编码模型，使用Encoder，拥有双向的注意力机制，即计算每一个词的特征时都看到完整上下文
  - 解码器模型：自回归模型，使用Decoder，拥有单向的注意力机制，即计算每一个词的特征时都只能看到上文，无法看到下文
  - 编码器解码器模型：序列到序列模型，使用Encoder+Decoder，Encoder部分使用双向的注意力，Decoder部分使用单向注意力

模型类型	常用预训练模型	适用任务
编码器模型，自编码模型	ALBERT, BERT, DistilBERT, RoBERTa	文本分类、命名实体识别、阅读理解
解码器模型，自回归模型	GPT, GPT-2, Bloom, LLaMA	文本生成
编码器解码器模型，序列到序列模型	BART, T5, Marian, mBART, GLM	文本摘要、机器翻译

Model Head

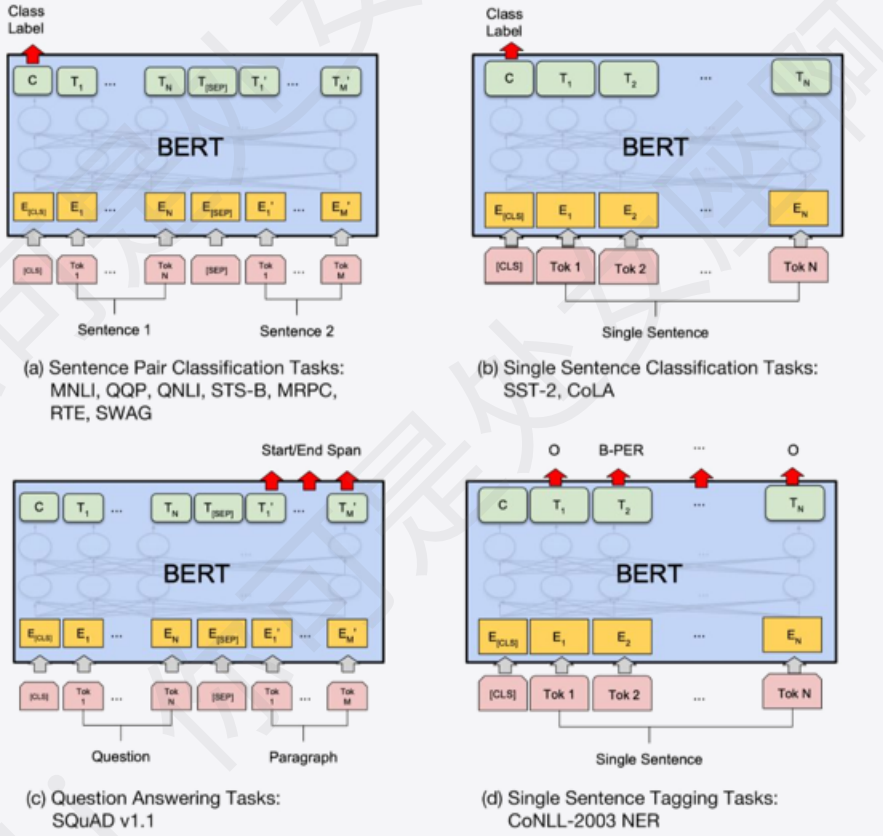
- **什么是Model Head**
  - Model Head 是连接在模型后的层，通常为1个或多个全连接层
  - Model Head 将模型的编码的表示结果进行映射，以解决不同类型的任务



基础组件之Model

Model Head

- **Transformers中的Model Head**
  - *Model（模型本身，只返回编码结果）
  - *ForCausalLM
  - *ForMaskedLM
  - *ForSeq2SeqLM
  - *ForMultipleChoice
  - *ForQuestionAnswering
  - *ForSequenceClassification
  - *ForTokenClassification
  - ... ...



Model基本使用方法

- **模型加载与保存**
  - 在线加载
  - 模型下载
  - 离线加载
  - 模型加载参数
- **模型调用**
  - 不带model head的模型调用
  - 带model head的模型调用

模型微调代码实例

- **任务类型**
  - 文本分类
- **使用模型**
  - hfl/rbt3
- **数据集地址**
  - https://github.com/SophonPlus/ChineseNlpCorpus

```
model.eval()
sen = "我觉得这家店的味道还是不错!"
with torch.inference_mode():
    inputs = tokenizer(sen, return_tensors="pt")
    batch = {k: v.cuda() for k, v in inputs.items()}
    logits = model(**batch).logits
    pred = torch.argmax(logits, dim=-1)
    print(f"评论: {sen}\n模型预测结果: {model.config.id2label.get(pred.item())}")
```

```
评论: 我觉得这家店的味道还是不错!
模型预测结果: 好评!
```

```
from transformers import pipeline

pipe = pipeline("text-classification", model=model, tokenizer=tokenizer, device=0)
```

```
pipe(sen)
```

```
[{'label': '好评! ', 'score': 0.9716703295707703}]
```