02

基础组件之Pipeline

（1）什么是Pipeline

（2）Pipeline支持的任务类型

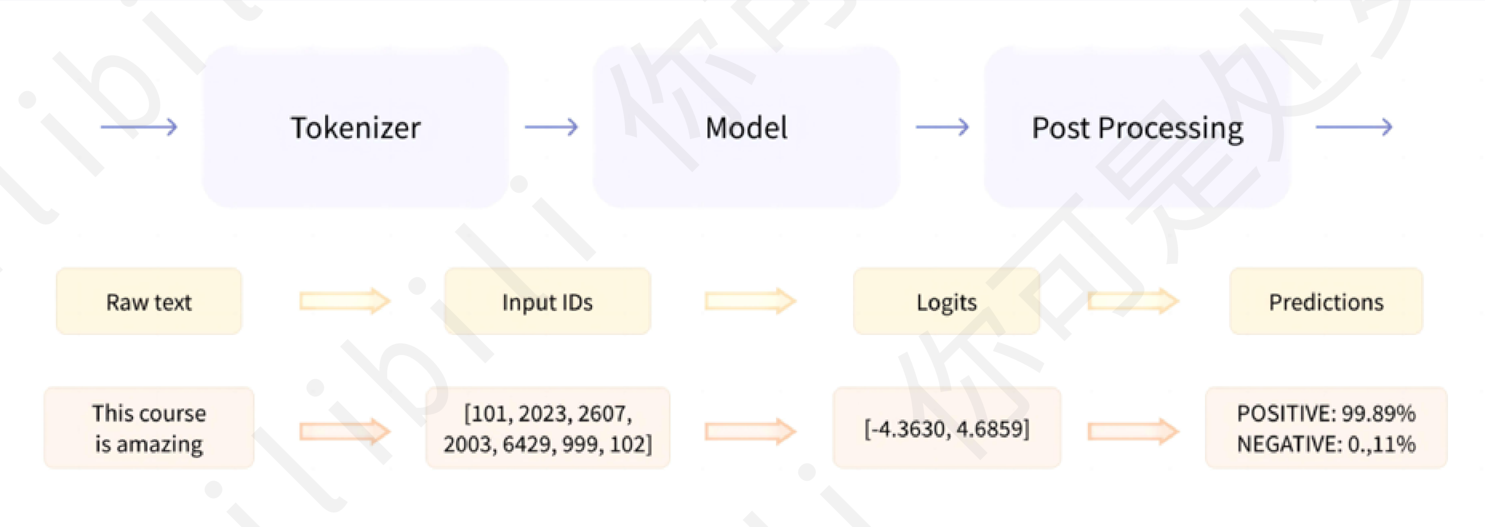
（3）Pipeline的创建与使用方式

（4）Pipeline的背后实现

基础组件之Pipeline

什么是Pipeline

- **Pipeline**
  - 将数据预处理、模型调用、结果后处理三部分组装成的流水线
  - 使我们能够直接输入文本便获得最终的答案



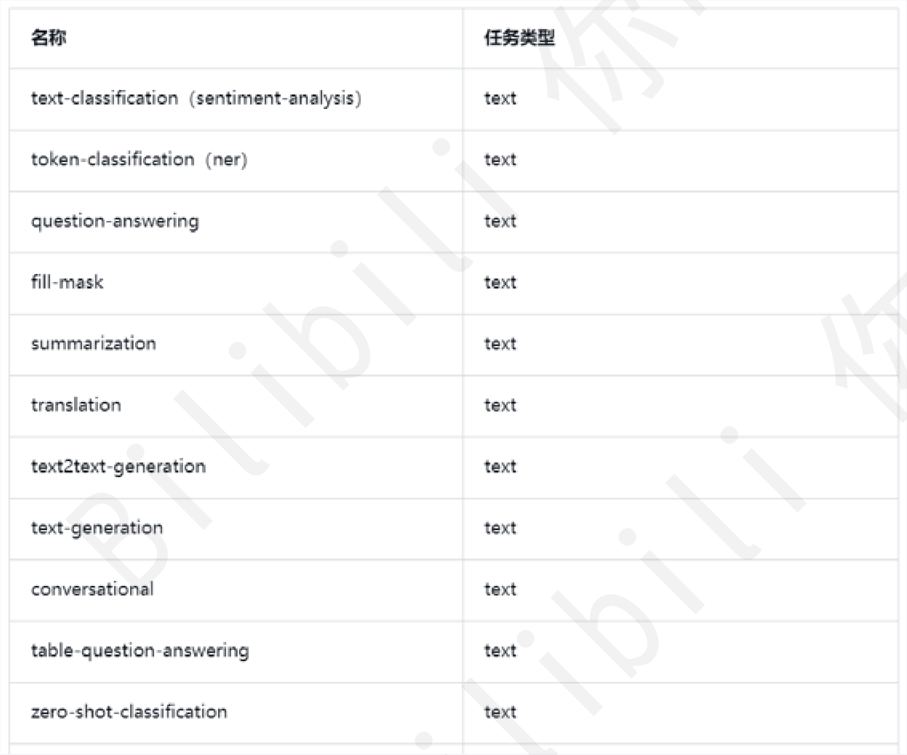
Tokenizer → Model → Post Processing

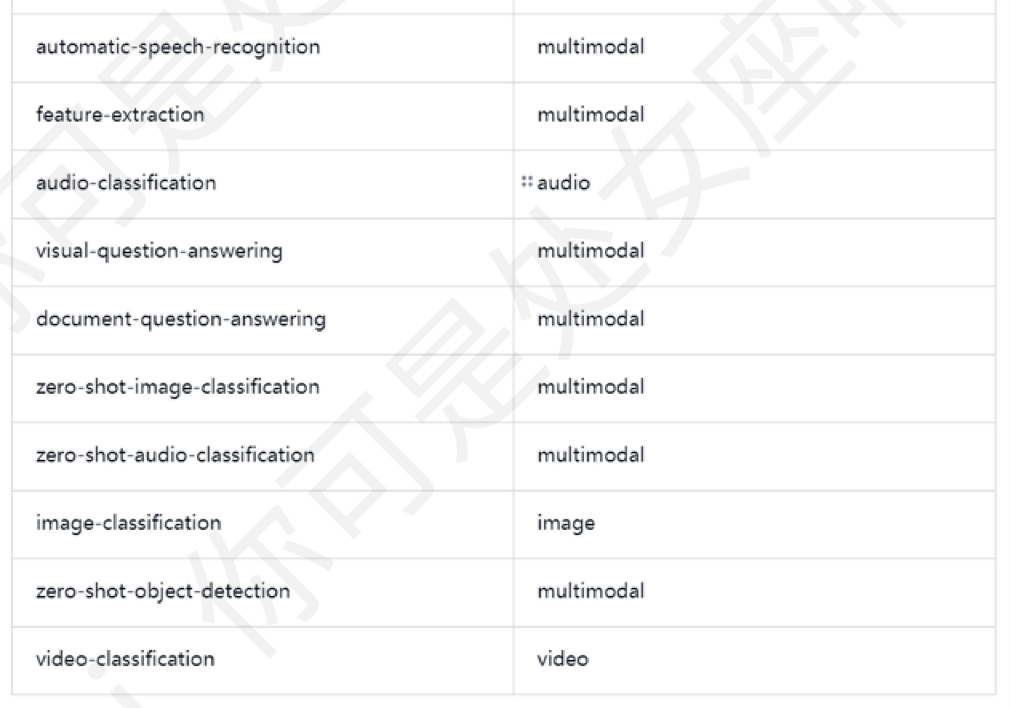
Raw text → Input IDs → Logits → Predictions

This course is amazing → [101, 2023, 2607, 2003, 6429, 999, 102] → [-4.3630, 4.6859] → POSITIVE: 99.89% NEGATIVE: 0.,11%

基础组件之Pipeline

Pipeline支持的任务类型





名称	任务类型
text-classification（sentiment-analysis）	text
token-classification（ner）	text
question-answering	text
fill-mask	text
summarization	text
translation	text
text2text-generation	text
text-generation	text
conversational	text
table-question-answering	text
zero-shot-classification	text
automatic-speech-recognition	multimodal
feature-extraction	multimodal
audio-classification	audio
visual-question-answering	multimodal
document-question-answering	multimodal
zero-shot-image-classification	multimodal
zero-shot-audio-classification	multimodal
image-classification	image
zero-shot-object-detection	multimodal
video-classification	video

基础组件之Pipeline

Pipeline创建与使用

- **根据任务类型直接创建Pipeline**
  - pipe = pipeline("text-classification")
- **指定任务类型，再指定模型，创建基于指定模型的Pipeline**
  - pipe = pipeline("text-classification", model="uer/roberta-base-finetuned-dianping-chinese")
- **预先加载模型，再创建Pipeline**
  - model = AutoModelForSequenceClassification.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")
  - tokenizer = AutoTokenizer.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")
  - pipe = pipeline("text-classification", model=model, tokenizer=tokenizer)
- **使用GPU进行推理加速**
  - pipe = pipeline("text-classification", model="uer/roberta-base-finetuned-dianping-chinese", device=0)

基础组件之Pipeline

Pipeline的背后实现

- **Step1 初始化Tokenizer**
  - tokenizer = AutoTokenizer.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")
- **Step2 初始化Model**
  - model = AutoModelForSequenceClassification.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")
- **Step3 数据预处理**
  - input_text = "我觉得不太行！"
  - inputs = tokenizer(input_text, return_tensors="pt")
- **Step4 模型预测**
  - res = model(**inputs).logits
- **Step5 结果后处理**
  - pred = torch.argmax(torch.softmax(logits, dim=-1)).item()
  - result = model.config.id2label.get(pred)