

```
!pip install -q transformers datasets peft accelerate evaluate jiwer
!pip install -q librosa soundfile # 音频处理库
```

```
84.1/84.1 kB 3.5 MB/s eta 0:00:00
3.2/3.2 MB 30.5 MB/s eta 0:00:00
```

双击（或按回车键）即可修改

```
from datasets import load_dataset, Audio
from transformers import WhisperProcessor

# 1. 加载模型对应的处理器
model_id = "openai/whisper-small"
processor = WhisperProcessor.from_pretrained(model_id, language="Chinese", task="transcribe")

# 2. 【修改点】加载 polyai/minds14 数据集（完全公开，无需登录）
print("正在加载 MINDS-14 中文数据...")
# "zh-CN" 是中文子集
dataset = load_dataset("polyai/minds14", "zh-CN", split="train", streaming=True)
dataset = dataset.take(50)

# 3. 数据预处理函数
def prepare_dataset(batch):
    audio = batch["audio"]

    # 提取声学特征
    # Whisper 能够处理重采样，只要告诉它原始采样率是多少
    batch["input_features"] = processor.feature_extractor(audio["array"], sampling_rate=audio["sampling_rate"])

    # 【关键修改点】MINDS-14 的文本列名叫 "transcription"，不是 "sentence"
    batch["labels"] = processor.tokenizer(batch["transcription"]).input_ids
    return batch

# 4. 强制重采样到 16000Hz (Whisper 标准)
dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
dataset = dataset.map(prepare_dataset)

print("数据准备完毕！(已替换为无门槛数据集)")
```

正在加载 MINDS-14 中文数据...

README.md: 23.5k/? [00:00<00:00, 1.74MB/s]

数据准备完毕！(已替换为无门槛数据集)

```
import torch
from dataclasses import dataclass
from typing import Any, Dict, List, Union

@dataclass
class DataCollatorSpeechSeq2SeqWithPadding:
    processor: Any

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
        # 处理音频输入 (input_features)
        input_features = [{"input_features": feature["input_features"]} for feature in features]
        batch = self.processor.feature_extractor.pad(input_features, return_tensors="pt")

        # 处理文本标签 (labels)
        label_features = [{"input_ids": feature["labels"]} for feature in features]
        labels_batch = self.processor.tokenizer.pad(label_features, return_tensors="pt")

        # 把 Padding 部分的标签设为 -100，这样计算 Loss 时会忽略它们
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -100)

        # 捲头去尾 (Whisper 的特殊要求)
        if (labels[:, 0] == self.processor.tokenizer.bos_token_id).all().cpu().item():
            labels = labels[:, 1:]

        batch["labels"] = labels
        return batch

    data_collator = DataCollatorSpeechSeq2SeqWithPadding(processor=processor)
```

```

from transformers import WhisperForConditionalGeneration
from peft import LoraConfig, get_peft_model, TaskType

# 1. 加载基座模型
print("正在加载 Whisper-Small...")
model = WhisperForConditionalGeneration.from_pretrained(model_id)

# 强制开启梯度检查点（节省显存）
model.config.use_cache = False

# 2. 定义 LoRA 配置（关键！）
config = LoraConfig(
    r=8,                      # Rank: 秩，8 是个经典值
    lora_alpha=32,             # Alpha: 缩放系数
    target_modules=["q_proj", "v_proj"], # 只微调 Attention 的 Q 和 V
    lora_dropout=0.05,
    bias="none",
)

# 3. 施加魔法：将 LoRA 挂载到 Whisper 上
model = get_peft_model(model, config)

# 4. 打印参数变化（截图这张表放在论文里！）
model.print_trainable_parameters()
# 预期输出：trainable params: ~300k, trainable%: ~0.15% (非常小！)

```

```

正在加载 Whisper-Small...
config.json: 1.97k/? [00:00<00:00, 150kB/s]

model.safetensors: 100% 967M/967M [00:11<00:00, 68.5MB/s]

generation_config.json: 3.87k/? [00:00<00:00, 222kB/s]

trainable params: 884,736 || all params: 242,619,648 || trainable%: 0.3647

```

```

from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer

training_args = Seq2SeqTrainingArguments(
    output_dir="./whisper-lora-zh-demo", # 输出目录
    per_device_train_batch_size=8,         # T4 显卡 Batch 设为 8 没问题
    gradient_accumulation_steps=1,        # 梯度累积
    learning_rate=1e-3,                  # LoRA 的学习率通常比全量微调大 (1e-3 vs 1e-5)
    max_steps=50,                      # 演示只跑 50 步
    fp16=True,                         # 开启半精度混合训练（加速+省显存）
    logging_steps=10,                   # 每 10 步打印一次日志
    save_steps=25,                     # 每 25 步保存一次
    remove_unused_columns=False,        # 必须设为 False
    report_to=["tensorboard"],
)

trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=dataset, # 这里用 streaming 数据集
    data_collator=data_collator,
    tokenizer=processor.feature_extractor,
)

print("开始微调训练...")
trainer.train()

```

```
/tmp/ipython-input-3565224546.py:16: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.
    trainer = Seq2SeqTrainer(
开始微调训练...
/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py:668: UserWarning: 'pin_memory' argument is
    warnings.warn(warn_msg)

-----
ImportError                                                 Traceback (most recent call last)
/tmp/ipython-input-3565224546.py in <cell line: 0>()
      23
      24     print("开始微调训练...")
--> 25     trainer.train()

-----
/usr/local/lib/python3.12/dist-packages/datasets/features/audio.py in decode_example(self, value,
token_per_repo_id)
    170         from .torchcodec import AudioDecoder
    171     else:
--> 172         raise ImportError("To support decoding audio data, please install 'torchcodec'.")
    173
    174     if not self.decode:

ImportError: To support decoding audio data, please install 'torchcodec'.
```

NOTE: If your import is failing due to a missing package, you can  
manually install dependencies using either !pip or !apt.

出错出错，新一轮尝试-----

To view examples of installing some common dependencies, click the

```
# 1. 卸载当前可能有问题的库
!pip uninstall -y datasets

# 2. 安装稳定的旧版本（配合 soundfile 使用最稳）
!pip install datasets==2.21.0
```

```
Found existing installation: datasets 4.0.0
Uninstalling datasets-4.0.0:
  Successfully uninstalled datasets-4.0.0
Collecting datasets==2.21.0
  Downloading datasets-2.21.0-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (3.20.
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (2.
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21
后续步骤：\[说明错误\]
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (4
Requirement already satisfied: xxhash in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (3.6.0)
Requirement already satisfied: multiprocessing in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (0
Collecting fsspec<=2024.6.1,>=2023.1.0 (from fsspec[http]<=2024.6.1,>=2023.1.0->datasets==2.21.0)
  Downloading fsspec-2024.6.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (3.13.2
Requirement already satisfied: huggingface-hub>=0.21.2 in /usr/local/lib/python3.12/dist-packages (from datasets=
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (25.0
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from datasets==2.21.0) (6.
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp->
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp->dataset
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp->datasets==
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12/dist-packages (from aiohttp->dataset
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.12/dist-packages (from aiohttp->data
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp->dataset
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12/dist-packages (from aiohttp->dataset
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggin
Requirement already satisfied: hf-xt><2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32.2->da
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests>=2.32
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas->da
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets==2.
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas->datasets==
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->
Downloading datasets-2.21.0-py3-none-any.whl (527 kB) 527.3/527.3 kB 17.1 MB/s eta 0:00:00
Downloading fsspec-2024.6.1-py3-none-any.whl (177 kB) 177.6/177.6 kB 13.7 MB/s eta 0:00:00
Installing collected packages: fsspec, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.0
    Uninstalling fsspec-2025.3.0:
      Successfully uninstalled fsspec-2025.3.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This b
gcsfs 2025.3.0 requires fsspec==2025.3.0, but you have fsspec 2024.6.1 which is incompatible.
Successfully installed datasets-2.21.0 fsspec-2024.6.1
```

```

!pip uninstall -y datasets

# 2. 安装稳定的 datasets 版本（解决 torchcodec 报错）和其他依赖
!pip install -q datasets==2.21.0
!pip install -q transformers peft accelerate evaluate jiwer librosa soundfile bitsandbytes

print("✅ 依赖安装完毕！请点击菜单栏 '代码执行程序' -> '重启会话'，然后从下一步继续运行。")

```

```

Found existing installation: datasets 2.21.0
Uninstalling datasets-2.21.0:
  Successfully uninstalled datasets-2.21.0
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 6.2 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.1/59.1 MB 21.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 126.2 MB/s eta 0:00:00

```

✅ 依赖安装完毕！请点击菜单栏 '代码执行程序' -> '重启会话'，然后从下一步继续运行。

```

# =====
# (重启会话后，从这里开始往下运行)
# 第二步：检查环境 & 导入库
# =====

import torch
import os

print(f"PyTorch Version: {torch.__version__}")
if not torch.cuda.is_available():
    raise RuntimeError("❌ 未检测到 GPU！请在菜单栏选择 '代码执行程序' -> '更改运行时类型' -> 'T4 GPU'。")
else:
    print(f"✅ GPU 就绪: {torch.cuda.get_device_name(0)}")

from datasets import load_dataset, Audio
from transformers import (
    WhisperProcessor,
    WhisperForConditionalGeneration,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer
)
from peft import LoraConfig, get_peft_model, TaskType
from dataclasses import dataclass
from typing import Any, Dict, List, Union

# =====
# 第三步：准备数据集（使用公开的 polyai/minds14 中文子集）
# =====
model_id = "openai/whisper-small"
processor = WhisperProcessor.from_pretrained(model_id, language="Chinese", task="transcribe")

print("正在加载数据集 (polyai/minds14)...")
# 加载中文子集, streaming=True 模式不占硬盘
dataset = load_dataset("polyai/minds14", "zh-CN", split="train", streaming=True)
dataset = dataset.take(50) # 演示只取前 50 条，实际训练请加大

def prepare_dataset(batch):
    audio = batch["audio"]
    # 提取 Log-Mel 声学特征
    batch["input_features"] = processor.feature_extractor(audio["array"], sampling_rate=audio["sampling_rate"]).input_features
    # 处理文本标签 (Minds14 的文本列名是 'transcription')
    batch["labels"] = processor.tokenizer(batch["transcription"]).input_ids
    return batch

# 强制重采样到 16000Hz 并应用预处理
dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
dataset = dataset.map(prepare_dataset)
print("✅ 数据集准备完毕")

# =====
# 第四步：定义数据整理器 (Data Collator)
# =====
@dataclass
class DataCollatorSpeechSeq2SeqWithPadding:
    processor: Any

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
        input_features = [{"input_features": feature["input_features"]} for feature in features]
        batch = self.processor.feature_extractor.pad(input_features, return_tensors="pt")

        label_features = [{"input_ids": feature["labels"]} for feature in features]
        labels_batch = self.processor.tokenizer.pad(label_features, return_tensors="pt")

```

```

        # 将 padding 部分的 loss 设为 -100 (忽略)
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -100)

        if (labels[:, 0] == self.processor.tokenizer.bos_token_id).all().cpu().item():
            labels = labels[:, 1:]

        batch["labels"] = labels
        return batch

data_collator = DataCollatorSpeechSeq2SeqWithPadding(processor=processor)

# =====
# 第五步：加载模型并挂载 LoRA (核心算法步骤)
# =====
print("正在加载预训练模型...")
model = WhisperForConditionalGeneration.from_pretrained(model_id)
model.config.use_cache = False # 训练时必须关闭缓存

# 定义 LoRA 配置
config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none"
)

# 挂载 LoRA
model = get_peft_model(model, config)
print("✅ LoRA 挂载成功! 参数量对比如下: ")
model.print_trainable_parameters()

# =====
# 第六步：配置与开始训练
# =====
training_args = Seq2SeqTrainingArguments(
    output_dir="../whisper-lora-zh-demo",
    per_device_train_batch_size=8,
    gradient_accumulation_steps=1,
    learning_rate=1e-3,           # LoRA 学习率
    warmup_steps=5,
    max_steps=50,                # 演示跑 50 步, 毕设实际跑 500-1000 步
    fp16=True,                   # 开启半精度 (T4 GPU 必备)
    logging_steps=10,
    save_steps=25,
    remove_unused_columns=False,
    report_to=["tensorboard"],
    dataloader_num_workers=0,      # 避免多进程报错
    dataloader_pin_memory=False   # 避免内存警告
)

trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=dataset,
    data_collator=data_collator,
    processing_class=processor.feature_extractor, # 修复 tokenizer 弃用警告
)

print("🚀 开始微调训练...")
trainer.train()
print("✅ 训练完成! ")

# =====
# 第七步：简单验证推理 (可选)
# =====
model.eval()
print("\n正在进行推理验证...")
# 取一条数据测试
sample = next(iter(dataset))
input_tensor = torch.tensor(sample["input_features"]).unsqueeze(0).cuda()

with torch.no_grad():
    generated_ids = model.generate(input_tensor, language="Chinese")

transcription = processor.batch_decode(generated_ids, skip_special_tokens=True)[0]
print(f"📝 原始文本: {processor.decode(sample['labels'], skip_special_tokens=True)}")
print(f"🤖 模型识别: {transcription}")

# 保存 LoRA 权重

```

```
"--> 125     save_path = "my_graduation_lora_model"
126     model.save_pretrained(save_path)
127     print(f"💡 LoRA 权重已保存至: {save_path} 文件夹")
```

PyTorch Version: 2.9.0+cu126  
GPU 就绪: Tesla T4  
/usr/local/lib/python3.12/dist-packages/huggingface\_hub/utils/\_auth.py:94: UserWarning:  
The secret `HF\_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/t>)  
You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public models or datasets.  
warnings.warn(  
preprocessor\_config.json: 185k/? [00:00<00:00, 16.0MB/s]  
tokenizer\_config.json: 283k/? [00:00<00:00, 20.5MB/s]  
vocab.json: 836k/? [00:00<00:00, 22.9MB/s]  
tokenizer.json: 2.48M/? [00:00<00:00, 75.6MB/s]  
merges.txt: 494k/? [00:00<00:00, 23.3MB/s]  
normalizer.json: 52.7k/? [00:00<00:00, 4.63MB/s]  
added\_tokens.json: 34.6k/? [00:00<00:00, 3.57MB/s]  
special\_tokens\_map.json: 2.19k/? [00:00<00:00, 205kB/s]

正在加载数据集 (polyai/minds14)...

Downloading readme: 100% 23.5k/23.5k [00:00<00:00, 79.5kB/s]

✓ 数据集准备完毕  
正在加载预训练模型...  
config.json: 1.97k/? [00:00<00:00, 173kB/s]  
model.safetensors: 100% 967M/967M [00:06<00:00, 233MB/s]  
generation\_config.json: 3.87k/? [00:00<00:00, 393kB/s]

✓ LoRA 挂载成功! 参数量对比如下:  
trainable params: 884,736 || all params: 242,619,648 || trainable%: 0.3647  
🚀 开始微调训练...  
You're using a WhisperTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `\_\_call\_\_` method  
[ 8/50 00:04 < 00:29, 1.43 it/s, Epoch 0.14/9223372036854775807]

### Step Training Loss

```
-----  
ValueError Traceback (most recent call last)  
/tmp/ipython-input-715517452.py, in <cell line: 0>()  
125  
126     print("🚀 开始微调训练...")  
--> 127     trainer.train()  
128     print("✓ 训练完成! ")  
129  
-----  
◆ 3 frames -----  
/usr/local/lib/python3.12/dist-packages/accelerate/data\_loader.py, in __iter__(self)  
886  
887         if batch is None:  
--> 888             raise ValueError(  
889                 f"Batch does not contain any data (`{batch}`). At the end of all iterable data  
available before expected stop iteration."  
890             )
```

ValueError: Batch does not contain any data ('None'). At the end of all iterable data available before expected stop iteration.

后续步骤: [说明错误](#)

```
# =====  
# 第三步: 准备数据集 (修正版: 关闭 streaming, 避免数据不够用的报错)  
# =====  
model_id = "openai/whisper-small"
```

```

processor = WhisperProcessor.from_pretrained(model_id, language="Chinese", task="transcribe")

print("正在加载数据集 (polyai/minds14)...")
# 【核心修改】streaming=False (下载到本地), 并移除 .take(50)
# 这样数据量充足, 且支持自动循环训练 (Epochs)
dataset = load_dataset("polyai/minds14", "zh-CN", split="train", streaming=False)

def prepare_dataset(batch):
    audio = batch["audio"]
    # 提取 Log-Mel 声学特征
    batch["input_features"] = processor.feature_extractor(audio["array"], sampling_rate=audio["sampling_rate"])
    # 处理文本标签
    batch["labels"] = processor.tokenizer(batch["transcription"]).input_ids
    return batch

# 强制重采样到 16000Hz 并应用预处理
dataset = dataset.cast_column("audio", Audio(sampling_rate=16000))
# num_proc=1 避免多进程潜在报错
dataset = dataset.map(prepare_dataset, num_proc=1)

print(f"✅ 数据集准备完毕, 共有 {len(dataset)} 条数据")

```

正在加载数据集 (polyai/minds14)...

Downloading data: 100% 32.4M/32.4M [00:01<00:00, 21.8MB/s]

Generating train split: 100% 502/502 [00:00<00:00, 3750.99 examples/s]

Map: 100% 502/502 [01:37<00:00, 56.56 examples/s]

✅ 数据集准备完毕, 共有 502 条数据

```

# =====
# 第六步：配置与开始训练（修正版）
# =====

training_args = Seq2SeqTrainingArguments(
    output_dir=".whisper-lora-zh-demo",
    per_device_train_batch_size=8,
    gradient_accumulation_steps=1,
    learning_rate=1e-3,
    warmup_steps=5,
    max_steps=50,           # 现在即使数据不够 400 条, 它也会自动跑第二轮, 不会报错了
    fp16=True,
    logging_steps=10,
    save_steps=25,
    remove_unused_columns=False,
    report_to=["tensorboard"],
    dataloader_num_workers=0,
    dataloader_pin_memory=False
)

trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=dataset,
    data_collator=data_collator,
    processing_class=processor.feature_extractor,
)

print("🚀 开始微调训练...")
trainer.train()
print("✅ 训练完成!")

```

🚀 开始微调训练... [50/50 02:05, Epoch 0/1]

### Step Training Loss

Step	Training Loss
10	1.762900
20	1.174700
30	0.970700
40	1.010500
50	1.027900

✅ 训练完成!

```

import torch
import random

# 1. 确保模型处于评估模式

```

```
model.eval()

# 2. 从数据集中随机抽取 3 条数据
# 因为我们之前把 dataset 转成了 list 格式（非流式），所以可以直接用索引
num_samples = 3
indices = random.sample(range(len(dataset)), num_samples)

print("=*60")
print(f"📊 实验结果展示（随机抽取 {num_samples} 条）")
print("=*60")

for idx in indices:
    sample = dataset[idx]

    # 3. 准备输入
    input_features = torch.tensor(sample["input_features"]).unsqueeze(0).cuda()

    # 4. 模型推理
    # language="Chinese" 强制指定中文，避免模型“不知所措”
    with torch.no_grad():
        generated_ids = model.generate(input_features, language="Chinese")

    # 5. 解码文本
    prediction = processor.batch_decode(generated_ids, skip_special_tokens=True)[0]
    reference = processor.decode(sample["labels"], skip_special_tokens=True)

    # 6. 打印对比
    print(f"🎧 音频索引: {idx}")
    print(f"✅ 真实标签: {reference}")
    print(f"🤖 模型预测: {prediction}")

    # 简单判断完全匹配
    if prediction == reference:
        print("🌟 结果: 完美匹配!")
    else:
        print("⚠️ 结果: 存在差异（可能是同音字或标点）")
    print("-" * 60)

=====
📊 实验结果展示（随机抽取 3 条）
=====
The attention mask is not set and cannot be inferred from input because pad token is
🎧 音频索引: 327
✅ 真实标签: 我想询问有关系接付款的问题
🤖 模型预测: 我想询问有关机接付款的问题
⚠️ 结果: 存在差异（可能是同音字或标点）

-----
🎧 音频索引: 57
✅ 真实标签: 请帮我开设一个联名账户可以吗谢谢
🤖 模型预测: 请帮我开设一个联名账户可以吗谢谢
🌟 结果: 完美匹配!

-----
🎧 音频索引: 12
✅ 真实标签: 你好我想查询我的账户余额显示我的账户余额请帮我查看我的账户余额
🤖 模型预测: 你好我想查询我的账户余额显示我的账户余额请帮我查看我的账户余额
🌟 结果: 完美匹配!
```

```
# 第一步：因为转换需要读取 .ipynb 文件，先将当前的笔记下载到 Colab 环境中
# 1. 点击左侧“文件”图标 -> 点击“上传”图标（或直接把下载好的 .ipynb 拖进去）
# 2. 或者直接运行下面这行命令，将当前的笔记（假设叫 your_notebook.ipynb）转换

# 假设您的文件叫 train_lora.ipynb
# 如果不知道文件名，可以先手动下载 .ipynb 再上传回左边文件栏
!jupyter nbconvert --to html /content/Untitled0.ipynb

# 运行完后，点击左侧文件夹图标，刷新一下，就能看到 train_lora.html 了
# 右键点击该 html 文件，选择“下载”即可。
```

```
--no-input
    Exclude input cells and output prompts from converted document.
        This mode is ideal for generating code-free reports.
        Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --TemplateExporter.no_input=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
    Equivalent to: [-WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [-WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides export.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
        ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'qtdocx']
        or a dotted object name that represents the import path for an
        ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
```