第二通道技术规格书：图文语义一致性检测

模块名称：Channel 2: Semantic Consistency Detection
文件路径：channel_2_consistency/
定位：系统语义层防线，负责检测跨模态（图片内容 vs 文本描述）的逻辑匹配度。

1. 核心技术目标

本模块旨在构建一个跨模态的语义对齐检测器，不关注图片像素是否经过 PS 篡改，而是聚焦于检测 **“移花接木” (Cheapfakes)** 类造假，即使用真实的图片配以虚假或错误的文字描述。

1.1 覆盖范围

1. **完全不符 (Irrelevant Mismatch)**
   - **场景**：图片内容与文本描述风马牛不相及。
   - **示例**：图片显示“森林火灾”，文本描述“市中心商场打折促销”。
2. **属性冲突 (Attribute Conflict)**
   - **场景**：图片主体正确，但在环境、天气、情感基调等关键属性上存在矛盾。
   - **示例**：图片显示“阳光明媚的公园”，文本描述“台风登陆，暴雨如注”。

1.2 检测原理 (Contrastive Learning)

本模块基于 **CLIP (Contrastive Language-Image Pre-training)** 模型的对比学习原理：

- **向量映射**：将图像 (I) 和文本 (T) 映射到同一个高维向量空间。
- **相似度计算**：计算图像特征向量 (V_I) 和文本特征向量 (V_T) 之间的**余弦相似度 (Cosine Similarity)**。
- **判定逻辑**：
  - 向量夹角小（相似度高） $\rightarrow$ 图文描述同一事物 $\rightarrow$ **Real**。
  - 向量夹角大（相似度低） $\rightarrow$ 图文语义分离 $\rightarrow$ **Fake**。

2. 技术选型与模型架构

首选方案：OpenAI CLIP

- **具体版本**：openai/clip-vit-base-patch32 (基于 HuggingFace Transformers)
- **核心优势**：
  - **Zero-Shot 能力**：无需针对特定的新闻数据集进行微调，具备极强的通用常识推理能力。
  - **轻量级高效**：ViT-Base 版本模型体积适中 (~600MB)，在普通显卡甚至 CPU 环境下均可流畅运行推理。
  - **生态成熟**：HuggingFace 提供标准 API，易于集成和维护。

3. 接口定义与目录结构 (Interface & Directory)

本模块代码必须严格放置于 channel_2_consistency/ 目录下，并对外暴露统一接口。

3.1 实际目录结构要求

```
MultiChannel-Reasoning-System/
├── data/
│   ├── images/            # [数据仓] 图片源
│   └── YuanJing_AI_Data_Standard.xlsx  # [标准表]
├── channel_2_consistency/   # [本模块工作区]
│   ├── clip_check.py       # [主代码] 必须包含 ConsistencyDetector 类
│   └── utils.py           # [工具类] (可选)
└── main_demo.py           # [调用方] 根目录主程序
```

3.2 输入输出规范

维度	参数名	类型	描述
Input 1	image_path	str	待检测图片的**相对路径**。例如 ./data/images/002_fire.jpg。代码需具备处理相对路径的能力。
Input 2	text	str	待检测的新闻文本内容。注意 CLIP 模型通常有输入长度限制（77 tokens），代码层需做截断处理。
Output 1	score	float	**图文一致性分数** (0.0 - 1.0)。 > 0.25 (经验阈值) 判定为匹配 (Real)。 < 0.2 判定为不匹配 (Fake)。
Output 2	message	str	**诊断信息**。例如 "High Consistency" 或 "Semantic Mismatch Detected"。

Visual	-	-	本通道主要输出数值结论，暂无特定图像产物。演示时可直接展示 Score 进度条。

4. Excel 数据字段对代码逻辑的影响

请务必理解 data/YuanJing_AI_Data_Standard.xlsx 中各字段如何影响本模块的开发与测试。

4.1 关键字段解析

字段名称	对 Channel 2 代码逻辑的影响
Text_Content	**核心语义输入**。 代码需读取此字段内容，作为 CLIP 模型的 Text Encoder 输入。这是本通道区别于第一通道的关键。若文本过长，代码内部需做截断处理。
Image_Path	**核心视觉输入**。 作为 CLIP 模型的 Image Encoder 输入。
Ch2_Consis_Label	**Ground Truth (标准答案)**。 用于验证模型准确率。 1 = 图文匹配 (Real) 0 = 图文不符 (Fake) **特别注意**：本字段与 Ch1_Tamper_Label 是**正交独立**的。一张 P 过的假图（Ch1=1），如果文字描述了 P 出来的内容，那么图文可能是一致的（Ch2=1）。本通道只负责判断“图和文说的是不是一回事”。

4.2 数据构造与验收标准 (SOP)

为验证模型对图文不符情况的检测能力，需构造以下两类数据放入 data/images/ 并录入 Excel 进行测试。

Type A: 移花接木样本 (Label=0)

- **操作**：找一张真实的灾难图（如森林火灾），配一段完全无关的文字（如"某商场打折"）。
- **Excel设置**：Ch2_Consis_Label 设为 0。
- **预期**：CLIP 输出分数极低 (< 0.2)。

Type B: 真实匹配样本 (Label=1)

- **操作**：找一张图，使用 Image Captioning 工具或人工撰写一段准确描述。
- **Excel设置**：Ch2_Consis_Label 设为 1。
- **预期**：CLIP 输出分数较高 (> 0.25)。

5. 开发里程碑

1. **环境搭建**：安装 transformers, torch, pillow 库。
2. **模型集成**：在 clip_check.py 中加载 openai/clip-vit-base-patch32，替换 Mock 逻辑。
3. **截断处理**：实现对超长文本的自动截断（取前 77 个 Token），防止模型报错。

6. 代码实现骨架

请将以下代码保存为 channel_2_consistency/clip_check.py。代码已包含 Mock 逻辑以应对环境缺失情况。

```
import os
import random
# from PIL import Image
# from transformers import CLIPProcessor, CLIPModel
# import torch

class ConsistencyDetector:
    def __init__(self):
        """
        初始化检测器
        任务：加载 OpenAI CLIP 模型 (ViT-Base-Patch32)。
        注意：首次运行会自动下载模型权重 (~600MB)，请保持网络通畅。
        """
        print("[Ch2-Init] Loading CLIP Model (openai/clip-vit-base-patch32)...")

        # ----------------------------------------------------------
        # TODO: [Step 1] 集成真实 CLIP 模型
        # self.device = "cuda" if torch.cuda.is_available() else "cpu"
        # self.model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32").to(self.device)
        # self.processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

```
        # ------------------------------------------------------------
        pass

    def check(self, image_path, text):
        """
        执行图文一致性检测

        Args:
            image_path (str): 图片相对路径
            text (str): 待检测的文本内容

        Returns:
            score (float): 0.0 - 1.0 (匹配度)
            message (str): 诊断信息
        """
        # 1. 路径处理
        if not os.path.exists(image_path):
            # 尝试修复路径
            abs_path = os.path.abspath(image_path)
            if os.path.exists(abs_path):
                image_path = abs_path
            else:
                return 0.0, "Error: File Not Found"

        # 文本截断处理 (CLIP通常限制77 token)
        short_text = text[:70]

        print(f"[Ch2-Analysis] Matching Image '{os.path.basename(image_path)}' with Text:
'{short_text}...'")

        # ------------------------------------------------------------
        # TODO: [Step 2] 接入真实 CLIP 推理
        # image = Image.open(image_path)
        # inputs = self.processor(text=[short_text], images=image, return_tensors="pt",
padding=True).to(self.device)
        #
        # with torch.no_grad():
        #     outputs = self.model(**inputs)
        #     logits_per_image = outputs.logits_per_image  # image-text similarity score
        #     probs = logits_per_image.softmax(dim=1)      # label probabilities
        #
        #     # 简单的相似度归一化处理 (CLIP raw logits 需要 sigmoid 或 softmax 处理)
        #     # 这里建议直接取 logits_per_image.item() / 100 粗略归一化，或者使用 Cosine
```

```
Similarity
        #     score = probs[0][0].item()
        # ----------------------------------------------------------

        # ============================================================
        # [Mock Logic] 模拟逻辑 (用于联调)
        # 基于 Excel 中的逻辑构造：图文不符 (移花接木) vs 匹配
        # ============================================================

        file_name = image_path.lower()
        score = 0.0
        msg = "Mismatch"

        # 场景 A: 典型的图文不符 (移花接木)
        # 假设我们在 Excel 里故意放了一些 'fire' 的图，但配了 'sunny' 的文字
        # 或者 ID 为 002 (你的示例数据: Forest Fire 图 + 商场火灾文 -> 看起来相关其实不符)

        is_mismatch_mock = False

        # 模拟逻辑：如果图是 fire，文是 mall (Excel ID 002 case)
        if "fire" in file_name and "商场" in text:
            is_mismatch_mock = True
        # 模拟逻辑：如果图是 sunny，文是 台风 (Excel ID 003 case)
        elif "sunny" in file_name and "台风" in text:
            is_mismatch_mock = True

        if is_mismatch_mock:
            # 返回低分，表示不匹配
            score = 0.15
            msg = "Semantic Mismatch Detected"
        else:
            # 返回高分，表示匹配 (ID 001 case)
            score = 0.88
            msg = "High Consistency"

        return score, msg

# 单例导出
detector = ConsistencyDetector()

def check_consistency(image_path, text):
    """
    外部调用接口
```

```
    """
    return detector.check(image_path, text)
```