API

环境变量

暴露的参数，通过容器配置环境变量传入

1. OPERATION_TYPE： 操作类型
   - 推理模式（INFERENCE）
   - 评估模式（EVALUATION）
   - 裁判模式（JUDGE）
2. INFERENCE_RESULT: 推理结果路径，文件夹，多模型采用 inference_{MODEL_CONFIG_ID}_{DATASET_CONFIG_ID}.jsonl 文件名输出
3. EVALUATION_RESULT: 评估结果路径，文件夹，多模型采用 evaluation_{MODEL_CONFIG_ID}_{DATASET_CONFIG_ID}.json 文件名输出
4. DATASET_CONFIGS: 支持传入多个数据集，使用 json 数组字符串形式，数组中 json 对象参考下方单个数据集配置设置
5. MODEL_CONFIGS: 支持传入多模型配置，使用 json 数组字符串形式，数组中 json 对象参考下方单个模型配置设置
6. JUDGE_MODE：裁判模式，
   - 打分模式（SINGLE）
   - 对比模式（MULTIPLE）
7. INFERENCE_MODE：推理模式，
   - 覆盖原答案（OVERWRITE）
   - 不覆盖原答案（NOT_OVERWRITE）
8. PROMPT：提示词
9. PROMPT_MODE: 配合 PROMPT 参数使用，不同模式组装出的提示词不同，参数值如下
   - System 角色注入模式(SYSTEM_PROMPT)：使用 system 角色传入 prompt
   - 双 Human 轮次模式(DUAL_HUMAN)：通过两次 human 角色提问来传入 prompt
   - Prompt 合并输入模式(PROMPT_MERGE)：通过替换 prompt 中 `{{}}` 关键字为 `\n[{用户问题}]\n`

```
// SYSTEM_PROMPT
[{"role": "system", "content": "{prompt}"},{"role": "human", "content": "{question}"},{
// DUAL_HUMAN
[{"role": "human", "content": "{prompt}"},{"role": "human", "content": "{question}"},{"
// PROMPT_MERGE，会合并prompt和问题作为新的
[{"role": "human", "content": "{prompt}{question}"},{"role": "bot", "content": "{anwser
```

数据集配置（DATASET_CONFIGS）

1. DATASET_CONFIG_ID: 数据集配置id，唯一即可
2. DATASET_TYPE: 数据集类型，支持用户自定义数据集和内置数据集
   - BUILT_IN：内置数据集
   - CUSTOM：用户自定义数据集
3. EVALUATION_METRICS: 评估指标，多个指标使用 "," 分隔开，不传时使用通用的默认指标进行评估
   - 通用指标：BLEU-4,rouge1,rouge2,rougeL,rougeLsum
   - 其他指标

内置数据集配置（BUILT_IN）

1. BUILT_IN_DATASET: 内置数据集名称，部分指标和数据集存在特定的 n-n 关系，请注意区分，内置数据集评估指标目前固定，暂不支持自定义

自定义数据集配置（CUSTOM）

1. CUSTOM_DATASET_PATH: 自定义数据集路径
   - 支持文件夹和jsonl文件，多个路径使用 "," 分隔
   - 文件夹内需包含数据集对应的jsonl文件
   - jsonl文件内容请使用平台数据集标准格式
   - **注意**：多路径进行评估时会先合并成一个jsonl文件

示例

```
[
  {
    "DATASET_CONFIG_ID": "gsm8k_0",
    "DATASET_TYPE": "CUSTOM",
    "EVALUATION_METRICS": "BLEU-4,rouge1,rouge2,rougeL",
    "BUILT_IN_DATASET": "",
    "CUSTOM_DATASET_PATH": "/your_dataset_dir/gsm8k_0.jsonl"
  }
]
```

MODEL_CONFIGS

1. MODEL_CONFIG_ID: 模型配置ID，唯一即可
2. MODEL_TYPE: 模型类型
   - API: 推理服务API，仅提供API，兼容 OPENAI 规范推理接口
   - BUILT_IN: 内置模型，根据内置配置决定模型加载和使用方式
   - CUSTOM: 自定义模型
3. TEMPERATURE: 控制采样温度（0 表示贪婪解码），默认0
4. TOP_P: 控制要考虑的排名靠前的 token 的累积概率的浮点数。必须在(0, 1]之间。设置为 1 表示考虑所有 token。，默认1
5. PRESENCE_PENALTY: 对已经出现过的 token 增加/减少其再次出现的概率，默认0

API

1. API_TYPE: API类型，用来区分请求和返回数据格式
   - OpenAI: OpenAI 格式
   - Spark: 星火API 格式
2. API_URL: API地址
3. API_KEY: API的key
4. API_MODEL: API的model，有些api需要设置，不设置默认使用MODEL_CONFIG_ID
5. API_EXTRA_CONFIG: 其他配置，json字符串形式

```
// 讯飞云协议相关参数参考
{
  "DOMAIN": "",
  "APP_ID": "",
  "API_KEY": "",
  "API_SECRET": ""
}
```

BUILT_IN

1. BUILD_IN_MODEL_NAME: 内置模型名称

CUSTOM

1. CUSTOM_MODEL_NAME: 自定义模型名称
2. BASE_MODEL_PATH: 基础模型路径
3. LORA_WEIGHT_PATH: LoRA权重路径，有的话就会加载，否则不加载
4. NUMS_GPUS: 占卡数量

示例


```
[
  {
    "MODEL_CONFIG_ID": "qwen-plus-2025-01-25",
    "MODEL_TYPE": "API",
    "TEMPERATURE": "",
    "TOP_P": "",
    "PRESENCE_PENALTY": "",
    "API_TYPE": "OpenAI",
    "API_URL": "https://dashscope.aliyuncs.com/compatible-mode/v1/chat/completions",
    "API_KEY": "sk-8c019c61a5524a4fa6222ff0e9de9130",
    "API_MODEL": "qwen-plus-2025-01-25",
  },
  {
    "MODEL_CONFIG_ID": "qwen-plus-2025-01-25",
    "MODEL_TYPE": "CUSTOM",
    "TEMPERATURE": "",
    "TOP_P": "",
    "PRESENCE_PENALTY": "",
    "CUSTOM_MODEL_NAME": "",
    "BASE_MODEL_PATH": "/iflytek/base_model",
    "LORA_WEIGHT_PATH": "",
    "NUMS_GPUS": 1
  }
]
```


数据集格式

输入数据集

输入数据集目前支持平台一问一答类型数据，裁判模式（JUDGE）需要额外 prediction 字段用来标识推理结果

注意：
裁判模式为对比模式时

- 相同数据集来源，不同模型的推理结果存储于单个jsonl文件时，jsonl文件内容需满足特定格式

```
// 多个模型推理结果，对比模式
{
    "input": "1 + 1 = ?",
    "target": "2",
    "predictions": {
      "model_A": "3",
      "model_B": "2"
    }
}
```

- 相同数据集来源，不同模型的推理结果存储于多个jsonl文件时，文件路径需同时放在数据集配置 `DATASET_CONFIGS` 下的 `CUSTOM_DATASET_PATH` 参数下，用逗号分隔，每个jsonl文件格式如下，后续也会合并为单个jsonl，合并结果格式同上，模型名称会取文件名称

```
// 单个模型推理结果，打分模式
{
    "input": "1 + 1 = ?",
    "target": "2",
    "prediction": "2"
}
```

推理结果

推理结果保存在 INFERENCE_RESULT 路径下，文件名为 inference_{MODEL_CONFIG_ID}_{DATASET_CONFIG_ID}.jsonl，推理结果为 jsonl 格式，每行一个 json 对象。

示例

```
{
    "input": "1 + 1 = ?",
    "target": "2",
    "prediction": "2"
}
```

字段说明

- input: 输入数据集问题
- target: 输入数据集答案

- prediction: 推理结果

评估结果

评估结果保存在 EVALUATION_RESULT 路径下，文件名为 evaluation_{MODEL_CONFIG_ID}_{DATASET_CONFIG_ID}.json

示例如下：

```
{
    "BLEU-4": {
        "score": 4.044942824477648,
        "counts": [
            637,
            233,
            90,
            36
        ],
        "totals": [
            3676,
            3666,
            3656,
            3646
        ],
        "precisions": [
            17.328618063112078,
            6.355701036552101,
            2.461706783369803,
            0.9873834339001646
        ],
        "bp": 1.0,
        "sys_len": 3676,
        "ref_len": 997
    },
    "rouge1": {
        "precision": 0.21715204862065327,
        "recall": 0.7614587955888169,
        "fmeasure": 0.3308124911310067
    },
    "rouge2": {
        "precision": 0.08071117172277745,
        "recall": 0.2744374163707398,
        "fmeasure": 0.1227120876612372
    },
    "rougeL": {
        "precision": 0.15354296085506028,
        "recall": 0.5562949883181789,
        "fmeasure": 0.23563005831748232
    }
}
```

裁判模式评估prompt

打分模式

【系统提示】

你是一名公正、客观的 AI 模型评估裁判，任务是比较两个 AI 助手在相同问题下的回答质量，并对它们分别进行打分
本次评估任务属于 {{scene}} 场景
场景定义如下：{{scene_desc}}

请严格参考以下评估标准，从多个维度对每个回答进行评分（1– {{max_score}}分），并最终判断哪一个回答更好或是

【评估标准】
{{metric}}

【评分流程】
1. 阅读用户提出的原始指令
2. 阅读模型生成的回答内容
3. （可选）参考标准答案
4. 按照每条评估标准进行判断，给出合理打分5. 给出综合评分与评分理由

【用户指令】
{{input}}

【模型回答】
{{output}}

【参考答案】
{{ref_answer}}

请你根据上述内容，认真评估模型回答，并以如下格式输出：
【输出格式】
```
{
  "score": <请填入 0~{{max_score}} 之间的整数分数>,
  "reason": "<用1–3句话简要说明打分依据，可包含优缺点>"
}
```

注意事项：
– 不要重复执行用户的问题任务。
– 请严格按照输出格式输出，避免输出多余内容。

- 打分应依据质量而非内容长度，必要时允许满分或零分。

对比模式

【系统提示】
你是一名公正、客观且专业的 AI 模型评估专家，核心任务是对多个 AI 助手（模型数量≥2）针对相同问题所生成的回
本次评估场景为{{scene}}，具体定义如下：
{{scene_desc}}
在评估过程中，需严格依据下列多维度评估标准，对每一个模型的输出回答进行仔细考量与打分，满分为{{max_score
【评估标准】
{{metric}}
【评分流程】
仔细研读用户输入内容、参考答案（若有），以及所有待评估模型的输出回答。
按照各项评估标准，逐一、严谨地对每个模型的回答进行逐项打分，确保评分公正合理，如实反映各模型回答质量，分数
综合各模型在各评估标准下的得分及其整体表现，精准判定出表现最优的模型，或确定各模型间是否为并列关系。若存在
凝练、准确地阐述评分依据，突出各模型之间的主要优劣对比点，便于理解本次评分的合理性与准确性。
【用户指令】
{{input}}
【模型回答列表】
各模型对应的回答(json格式)：
```
{
        A:{{output_a}}（模型A的回答内容）
        B:{{output_b}}（模型B的回答内容）
        C:{{output_c}}（模型C的回答内容）
        ……（依次罗列所有模型的回答）
}
```

【参考答案】（如无可留空）
{{ref_answer}}
请严格遵循如下结构化格式输出结果，不得添加 JSON 结构之外的任何额外内容：
【输出格式】
```
{
"scores": {
        "score_a": 4,  // 模型A的得分
        "score_b": 3,  // 模型B的得分
        "score_c": 5   // 模型C的得分
        ……  // 其他模型对应分数组
},
"best_model": "C",  // 可选值为模型标识（如"A"、"B"、"C"等），若存在多个最优模型，则以列表形式呈现，
"reason": "模型C在准确性、逻辑性及完整性等多方面表现突出，优于其他模型；模型A虽表述较清晰但在关键信息覆
}
```

注意事项：
- 各模型分数务必为0 ~ {{max_score}} 间的整数，如实体现回答整体质量。
- "best_model" 字段需精准填写最优模型标识，多个并列最优用列表形式，无胜出者填 "NO_WINNER"。

- "reason" 字段应简洁明了、重点突出，清晰呈现各模型间的优劣对比关键点，字数一般控制在100字以内。
- 严格遵守上述输出格式要求，确保输出内容完整准确且可直接解析利用，不得出现任何不符合格式规范的内容。