Lesson 6 笔记

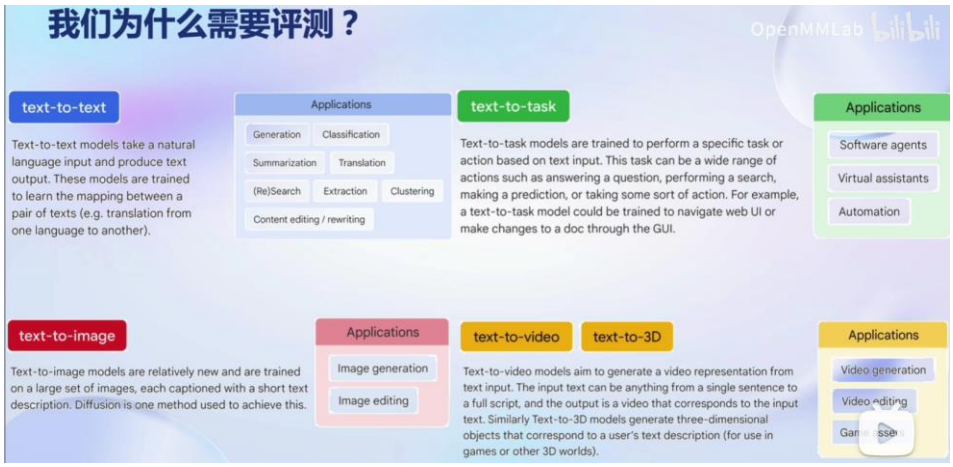
OpenCompass 大模型评测

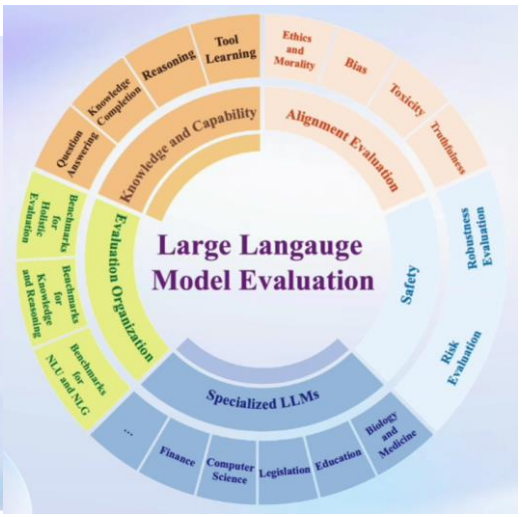
这节课分为 3 个部分，为什么要做大模型评测、OpenCompass 工具介绍和实际上手操作。

为了精确评估各种大模型的性能，国际上已经建立了多个系统评估框架。例如，斯坦福大学的 HELM、纽约大学与谷歌合作开发的 SuperGLUE、以及加州大学伯克利分校的 MMLU 都是评价模型多方面能力的重要工具。谷歌的 Big-Bench 及中国的 CLUE 和 CUGE 数据集则专注于模型在特定语言处理能力上的表现。

OpenCompass 推出了一个全面、高效和可扩展的评测方案。该方案通过一个分布式自动化评测系统，对语言和多模态模型的性能、安全性进行全面系统的评估，旨在更全面地理解和利用这些先进的人工智能技术。

为啥要做评测呢？这里有三个问题。





如何评测大语言模型？

基座模型

中国的首都是哪里？ → 中国的首都是

俄罗斯的首都是哪里？ → 北京
美国的首都是哪里？

对话模型（经过指令微调的模型）

中国的首都是哪里？

北京

客观评测

问答题

中国的首都是哪里？

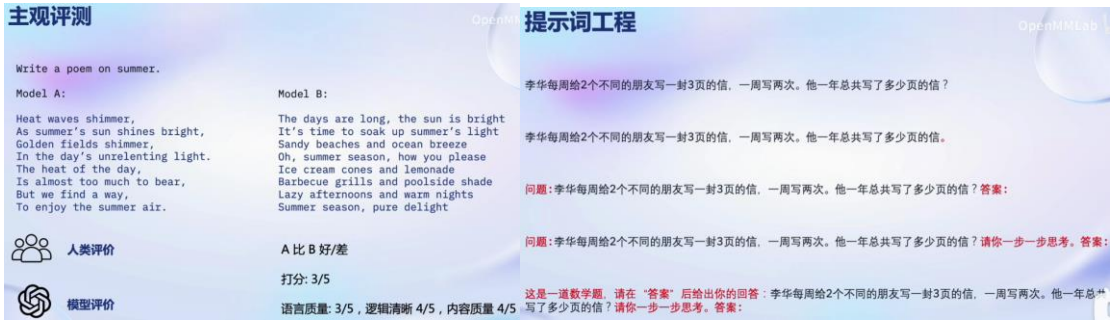
北京
北京。
中国的首都是北京。
中国的首都是北京（Beijing）。
答案是北京。

多选题

中国的首都是哪里？
A. 北京 B. 上海 C. 香港 D. 东京

A
A.
北京.
A. 北京

判断题/分类题/...



OpenCompass 介绍

评测对象

本算法库的主要评测对象为语言大模型与多模态大模型。我们以语言大模型为例介绍评测的具体模型类型。

- 基座模型：一般是经过海量的文本数据以自监督学习的方式进行训练获得的模型（如 OpenAI 的 GPT-3，Meta 的 LLaMA），往往具有强大的文字续写能力。
- 对话模型：一般是在的基座模型的基础上，经过指令微调或人类偏好对齐获得的模型（如 OpenAI 的 ChatGPT、上海人工智能实验室的书生·浦语），能理解人类指令，具有较强的对话能力。

工具架构

- 模型层：大模型评测所涉及的主要模型种类，OpenCompass 以基座模型和对话模型作为重点评测对象。
- 能力层：OpenCompass 从本方案从通用能力和特色能力两个方面来进行评测维度设计。在模型通用能力方面，从语言、知识、理解、推理、安全等多个能力维度进行评测。在特色能力方面，从长文本、代码、工具、知识增强等维度进行评测。
- 方法层：OpenCompass 采用客观评测与主观评测两种评测方式。客观评测能便捷地评估模型在具有确定答案（如选择，填空，封闭式问答等）的任务上的能力，主观评测能评估用户对模型回复的真实满意度，OpenCompass 采用基于模型辅助的主观评测和基于人类反馈的主观评测两种方式。
- 工具层：OpenCompass 提供丰富的功能支持自动化地开展大语言模型的高效评测。包括分布式评测技术，提示词工程，对接评测数据库，评测榜单发布，评测报告生成等诸多功能。

能力维度-设计思路

OpenCompass 为准确、全面地评估大型语言模型，从通用人工智能的角度，结合学术前沿和工业实践，提出了一套面向实际应用的模型能力评价体系。该体系分为通用能力和特色能力两部分。通用能力包括六大维度：学科综合能力、知识能力、语言能力、理解能力、推理能力和安全能力。这六大维度共同构成了一个立体和全面的模型能力评价框架。

评测方法

OpenCompass 采取客观评测与主观评测相结合的方法。针对具有确定性答案的能力维度和场景，通过构造丰富完善的评测集，对模型能力进行综合评价。针对体现模型能力的开放式或半开放式的问题、模型安全问题等，采用主客观相结合的评测方式。

客观评测

针对具有标准答案的客观问题，我们可以通过使用定量指标比较模型的输出与标准答案的差异，并根据结果衡量模型的性能。同时，由于大语言模型输出自由度较高，在评测阶段，我们需要对其输入和输出作一定的规范和设计，尽可能减少噪声输出在评测阶段的影响，才能对模型的能力有更加完整和客观的评价。

为了更好地激发出模型在题目测试领域的能力，并引导模型按照一定的模板输出答案，OpenCompass 采用提示词工程 （prompt engineering）和语境学习（in-context learning）进行客观评测。

在客观评测的具体实践中，通常采用下列两种方式进行模型输出结果的评测：

判别式评测：该评测方式基于将问题与候选答案组合在一起，计算模型在所有组合上的困惑度（perplexity），并选择困惑度最小的答案作为模型的最终输出。例如，若模型在 问题? 答案 1 上的困惑度为 0.1，在 问题? 答案 2 上的困惑度为 0.2，最终我们会选择 答案 1 作为模型的输出。

生成式评测：该评测方式主要用于生成类任务，如语言翻译、程序生成、逻辑分析题等。具体实践时，使用问题作为模型的原始输入，并留白答案区域待模型进行后续补全。我们通常还需要对其输出进行后处理，以保证输出满足数据集的要求。

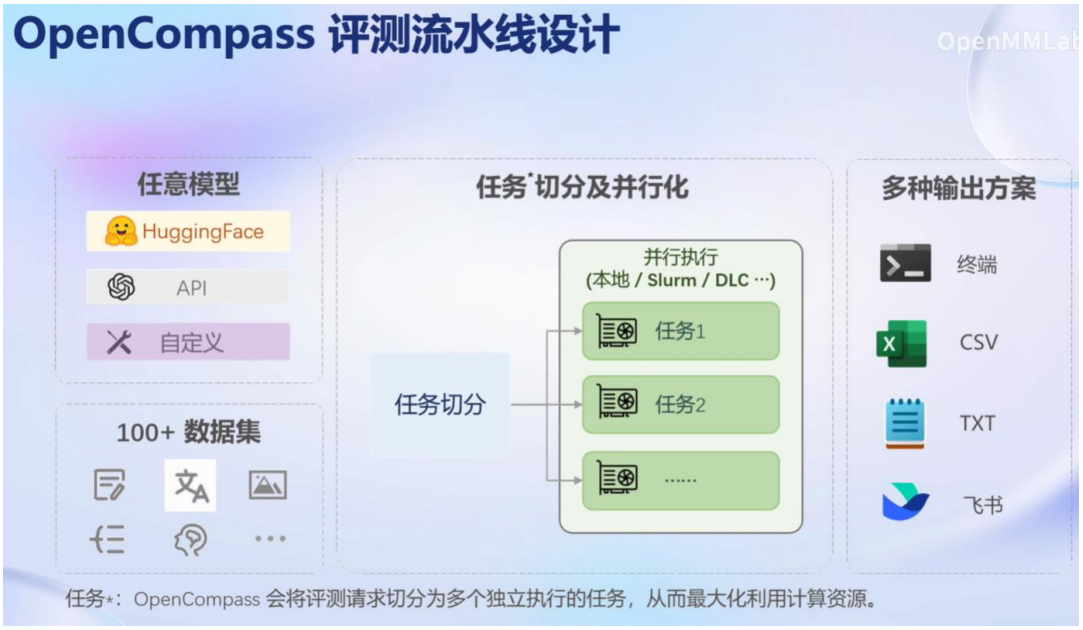
主观评测

语言表达生动精彩，变化丰富，大量的场景和能力无法凭借客观指标进行评测。针对如模型安全和模型语言能力的评测，以人的主观感受为主的评测更能体现模型的真实能力，并更符合大模型的实际使用场景。

OpenCompass 采取的主观评测方案是指借助受试者的主观判断对具有对话能力的大语言模型进行能力评测。在具体实践中，我们提前基于模型的能力维度构建主观测试问题集合，并将不同模型对于同一问题的不同回复展现给受试者，收集受试者基于主观

感受的评分。由于主观测试成本高昂，本方案同时也采用使用性能优异的大语言模拟人类进行主观打分。在实际评测中，本文将采用真实人类专家的主观评测与基于模型打分的主观评测相结合的方式开展模型能力评估。

在具体开展主观评测时，OpenComapss 采用单模型回复满意度统计和多模型满意度比较两种方式开展具体的评测工作。



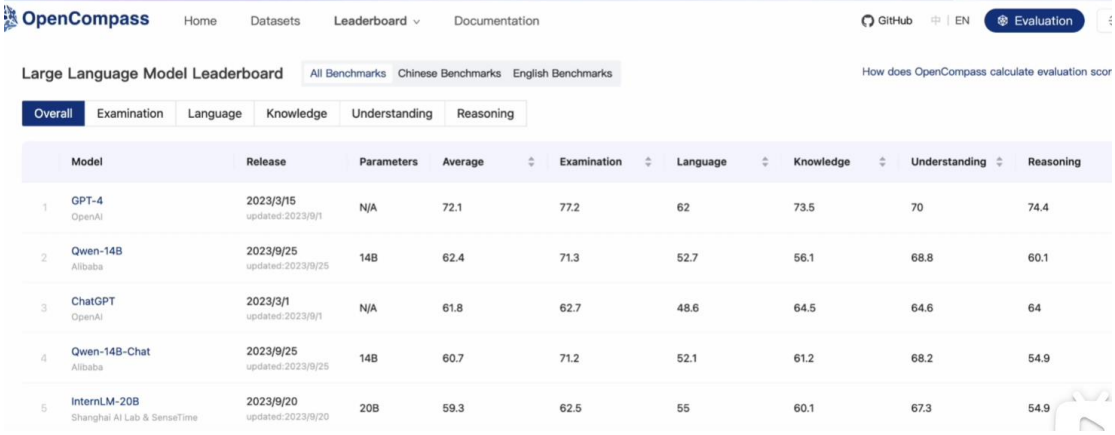
在 OpenCompass 中评估一个模型通常包括以下几个阶段：配置 -> 推理 -> 评估 -> 可视化。

配置：这是整个工作流的起点。您需要配置整个评估过程，选择要评估的模型和数据集。此外，还可以选择评估策略、计算后端等，并定义显示结果的方式。

推理与评估：在这个阶段，OpenCompass 将会开始对模型和数据集进行并行推理和评估。推理阶段主要是让模型从数据集产生输出，而评估阶段则是衡量这些输出与标准答案的匹配程度。这两个过程会被拆分为多个同时运行的“任务”以提高效率，但请注意，如果计算资源有限，这种策略可能会使评测变得更慢。

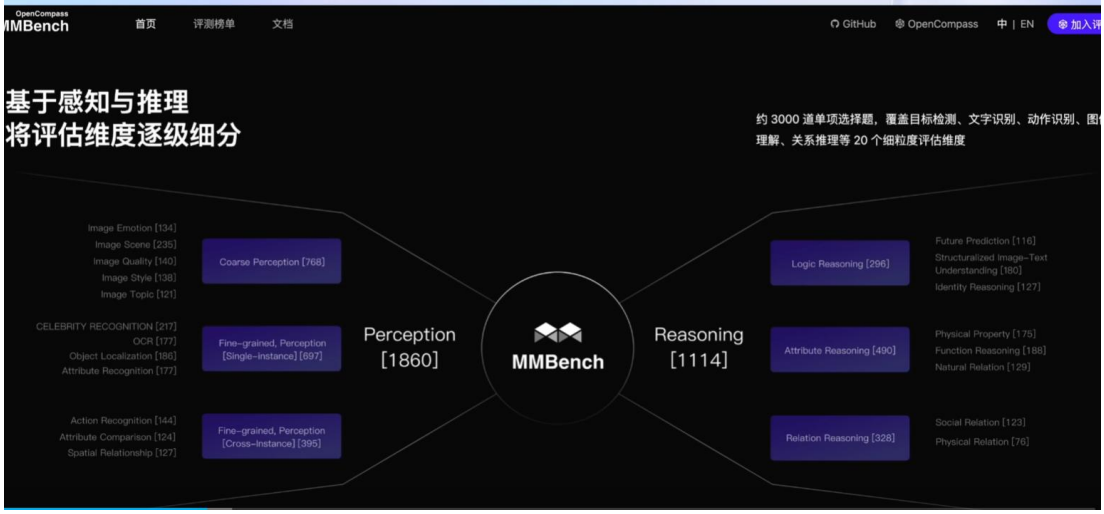
可视化：评估完成后，OpenCompass 将结果整理成易读的表格，并将其保存为 CSV 和 TXT 文件。你也可以激活飞书状态上报功能，此后可以在飞书客户端中及时获得评测状态报告。

大模型能力对比



	Model	Release	Parameters	Average	Examination	Language	Knowledge	Understanding	Reasoning
1	GPT-4 OpenAI	2023/3/15 updated:2023/9/1	N/A	72.1	77.2	62	73.5	70	74.4
2	Qwen-14B Alibaba	2023/9/25 updated:2023/9/25	14B	62.4	71.3	52.7	56.1	68.8	60.1
3	ChatGPT OpenAI	2023/3/1 updated:2023/9/1	N/A	61.8	62.7	48.6	64.5	64.6	64
4	Qwen-14B-Chat Alibaba	2023/9/25 updated:2023/9/25	14B	60.7	71.2	52.1	61.2	68.2	54.9
5	InternLM-20B Shanghai AI Lab & SenseTime	2023/9/20 updated:2023/9/20	20B	59.3	62.5	55	60.1	67.3	54.9

OpenCompass 前沿探索（多模态）



OpenCompass 前沿探索（法律领域）

开始实践

面向 GPU 的环境安装

conda create --name opencompass --clone=/root/share/conda_envs/internlm-base

source activate opencompass

git clone https://github.com/open-compass/opencompass

cd opencompass

pip install -e .

解压评测数据集到 data/ 处

cp /share/temp/datasets/OpenCompassData-core-20231110.zip /root/opencompass/

unzip OpenCompassData-core-20231110.zip

将会在 opencompass 下看到 data 文件夹

列出所有跟 internlm 及 ceval 相关的配置

python tools/list_configs.py internlm ceval

启动评测

确保按照上述步骤正确安装 OpenCompass 并准备好数据集后，可以通过以下命令评测 InternLM-Chat-7B 模型在 C-Eval 数据集上的性能。由于 OpenCompass 默认并行启动评估过程，我们可以在第一次运行时以 --debug 模式启动评估，并检查是否存在问题。在 --debug 模式下，任务将按顺序执行，并实时打印输出。

python run.py --datasets ceval_gen --hf-path /share/temp/model_repos/internlm-chat-7b/ --tokenizer-path /share/temp/model_repos/internlm-chat-7b/ --tokenizer-kwargs padding_side='left' truncation='left' trust_remote_code=True --model-kwargs trust_remote_code=True device_map='auto' --max-seq-len 2048 --max-out-len 16 --batch-size 4 --num-gpus 1 –debug

命令解析

--datasets ceval_gen \

--hf-path /share/temp/model_repos/internlm-chat-7b/ \ # HuggingFace 模型路径

--tokenizer-path /share/temp/model_repos/internlm-chat-7b/ \ # HuggingFace tokenizer 路径（如果与模型路径相同，可以省略）

--tokenizer-kwargs padding_side='left' truncation='left' trust_remote_code=True \ # 构建 tokenizer 的参数

--model-kwargs device_map='auto' trust_remote_code=True \ # 构建模型的参数

--max-seq-len 2048 \ # 模型可以接受的最大序列长度

--max-out-len 16 \ # 生成的最大 token 数

--batch-size 2 \ # 批量大小

--num-gpus 1 # 运行模型所需的 GPU 数量

--debug

正常就会有以下结果：

[2024-01-12 18:23:55,076] [opencompass.openicl.icl_inferencer.icl_gen_inferencer]

[INFO] Starting inference process...

除了通过命令行配置实验外，OpenCompass 还允许用户在配置文件中编写实验的完整配置，并通过 run.py 直接运行它。配置文件是以 Python 格式组织的，并且必须包括 datasets 和 models 字段。

示例测试配置在 configs/eval_demo.py 中。此配置通过 继承机制 引入所需的数据集和模型配置，并以所需格式组合 datasets 和 models 字段。

```
from mmengine.config import read_base

with read_base():
    from .datasets.siqa.siqa_gen import siqa_datasets
    from .datasets.winograd.winograd_ppl import winograd_datasets
    from .models.opt.hf_opt_125m import opt125m
    from .models.opt.hf_opt_350m import opt350m

datasets = [*siqa_datasets, *winograd_datasets]

models = [opt125m, opt350m]
```

运行任务时，只需将配置文件的路径传递给 run.py：

```
python run.py configs/eval_demo.py
```

OpenCompass 提供了一系列预定义的模型配置，位于 configs/models 下。以下是与 opt-350m（configs/models/opt/hf_opt_350m.py）相关的配置片段：

```
# 使用 `HuggingFaceCausalLM` 评估由 HuggingFace 的 `AutoModelForCausalLM` 支持的模型

from opencompass.models import HuggingFaceCausalLM

# OPT-350M

opt350m = dict(
        type=HuggingFaceCausalLM,
        # `HuggingFaceCausalLM` 的初始化参数
        path='facebook/opt-350m',
        tokenizer_path='facebook/opt-350m',
        tokenizer_kwargs=dict(
```

```
            padding_side='left',
            truncation_side='left',
            proxies=None,
            trust_remote_code=True),
        model_kwargs=dict(device_map='auto'),
        # 下面是所有模型的共同参数，不特定于 HuggingFaceCausalLM
        abbr='opt350m',                 # 结果显示的模型缩写
        max_seq_len=2048,               # 整个序列的最大长度
        max_out_len=100,                # 生成的最大 token 数
        batch_size=64,                  # 批量大小
        run_cfg=dict(num_gpus=1),       # 该模型所需的 GPU 数量
    )
```

使用配置时，我们可以通过命令行参数 --models 指定相关文件，或使用继承机制将模型配置导入到配置文件中的 models 列表中。

与模型类似，数据集的配置文件也提供在 configs/datasets 下。用户可以在命令行中使用 --datasets，或通过继承在配置文件中导入相关配置

下面是来自 configs/eval_demo.py 的与数据集相关的配置片段：

```
from mmengine.config import read_base  # 使用 mmengine.read_base() 读取基本配置

with read_base():
    # 直接从预设的数据集配置中读取所需的数据集配置
    from .datasets.winograd.winograd_ppl import winograd_datasets  # 读取 Winograd 配置，基于 PPL（困惑度）进行评估
    from .datasets.siqa.siqa_gen import siqa_datasets  # 读取 SIQA 配置，基于生成进行评估

datasets = [*siqa_datasets, *winograd_datasets]          # 最终的配置需要包含所需的评估数据集列表 'datasets'
```

数据集配置通常有两种类型：'ppl' 和 'gen'，分别指示使用的评估方法。其中 ppl 表示辨别性评估，gen 表示生成性评估。

此外，configs/datasets/collections 收录了各种数据集集合，方便进行综合评估。OpenCompass 通常使用 base_medium.py 进行全面的模型测试。要复制结果，只需导入该文件，例如：

python run.py --models hf_llama_7b --datasets base_medium

评估完成后，评估结果表格将打印如下：

dataset	version	metric	mode	opt350m	opt125m
siqa	e78df3	accuracy	gen	21.55	12.44
winograd	b6c7ed	accuracy	ppl	51.23	49.82

所有运行输出将定向到 `outputs/demo/` 目录，结构如下：

```
outputs/default/
├── 20200220_120000
├── 20230220_183030     # 每个实验一个文件夹
│   ├── configs         # 用于记录的已转储的配置文件。如果在同一个实验文件夹中重新运行了不同的实验，可能会保留多个配置
│   ├── logs            # 推理和评估阶段的日志文件
│   │   ├── eval
│   │   └── infer
│   ├── predictions   # 每个任务的推理结果
│   ├── results       # 每个任务的评估结果
│   └── summary       # 单个实验的汇总评估结果
├── ...
```

打印评测结果的过程可被进一步定制化，用于输出一些数据集的平均分 (例如 MMLU, C-Eval 等)。