



大模型微调实践

Large model fine-tuning practice

讲师：姜夏

目录



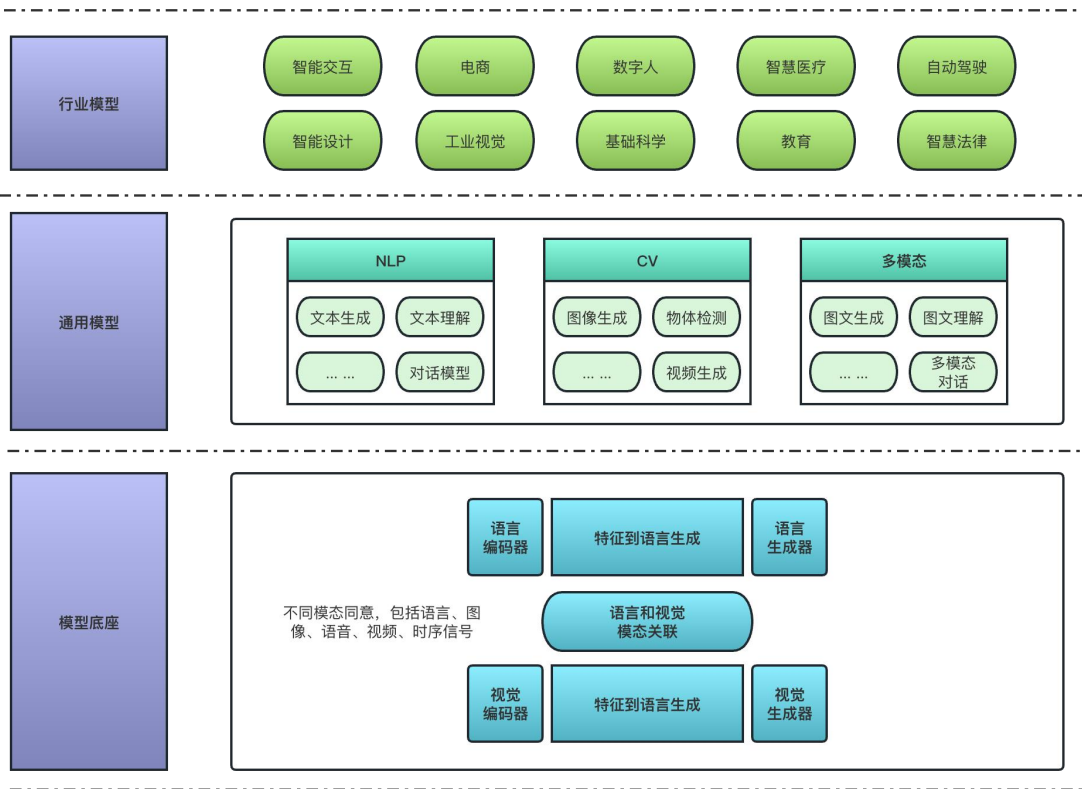
- 1 大模型问答系统设计
Large Model Question Answering System Design
- 2 大模型训练框架
Large Model Training Framework
- 3 Qwen2模型微调
Qwen2 SFT
- 4 大模型测试与评估
Large Model Testing and Evaluation
- 5 大模型部署与界面化调用
Large model deployment and interface call



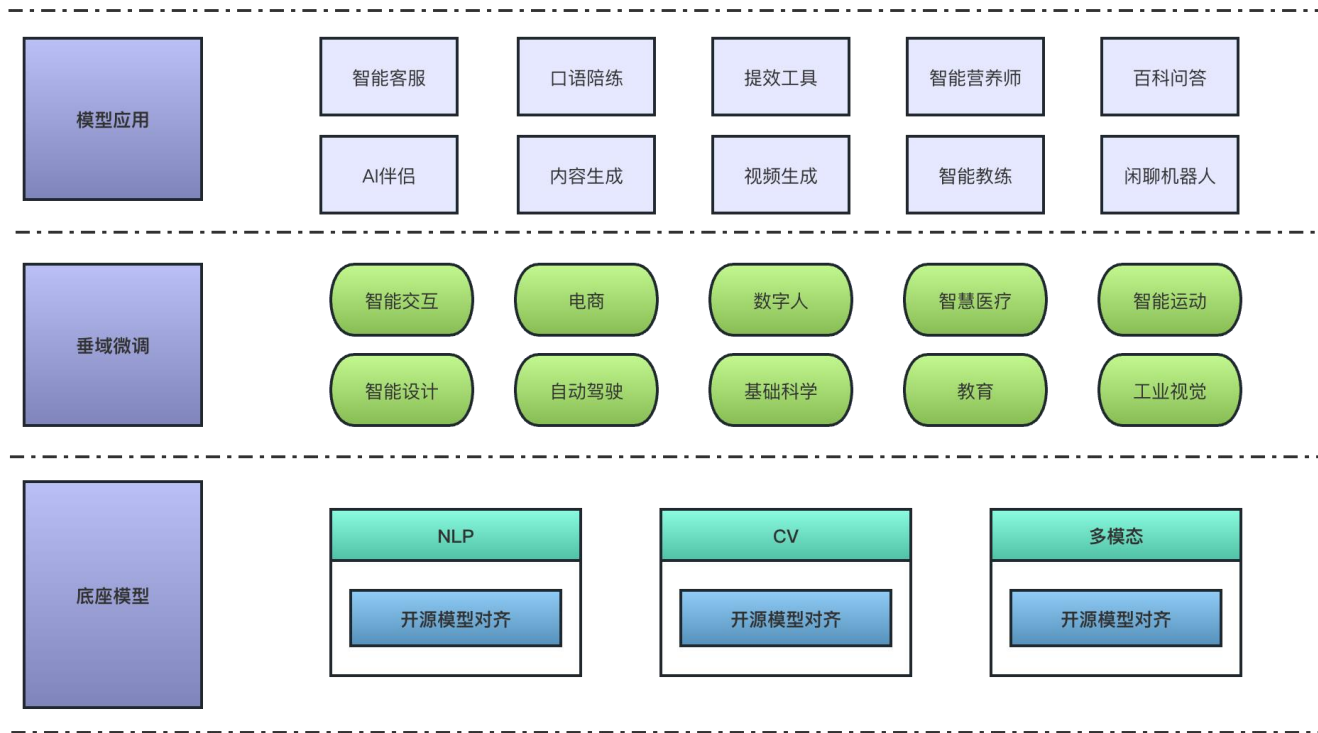
大模型问答系统设计

Large Model Question Answering System Design

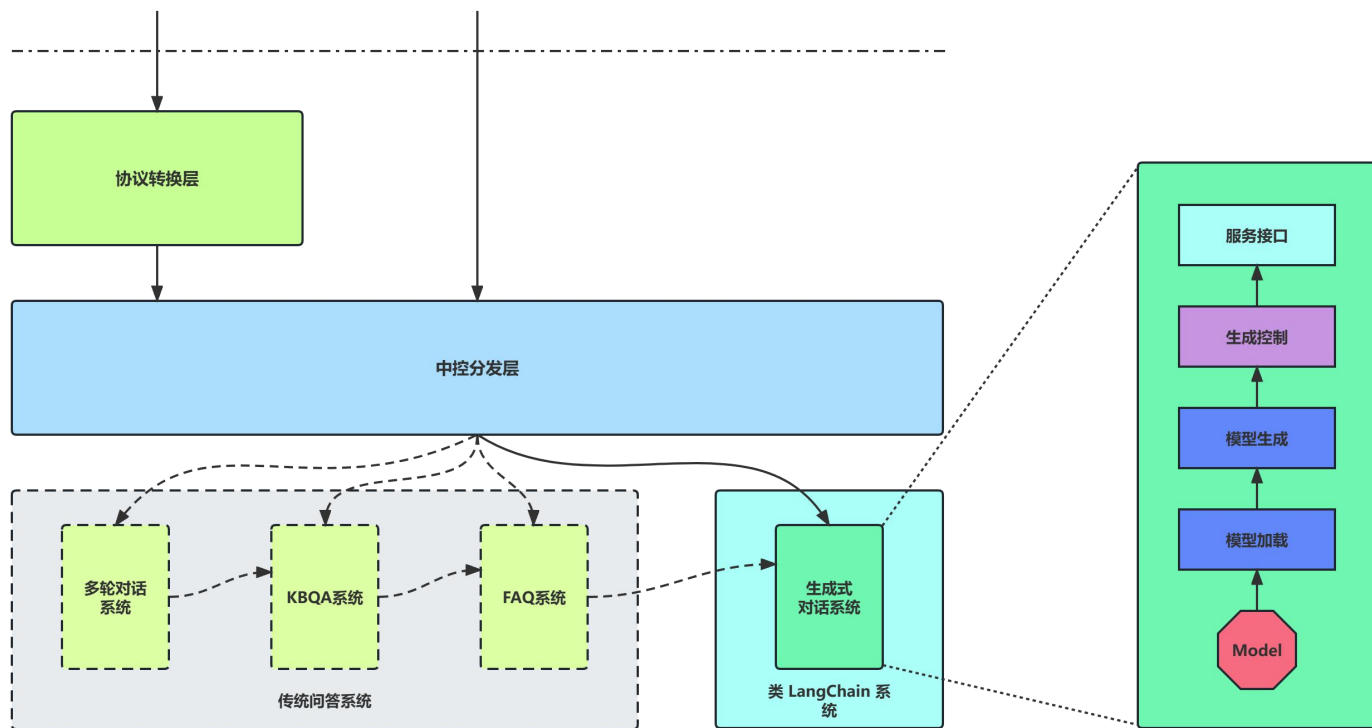
头部企业应用



中小企业应用



大模型问答系统



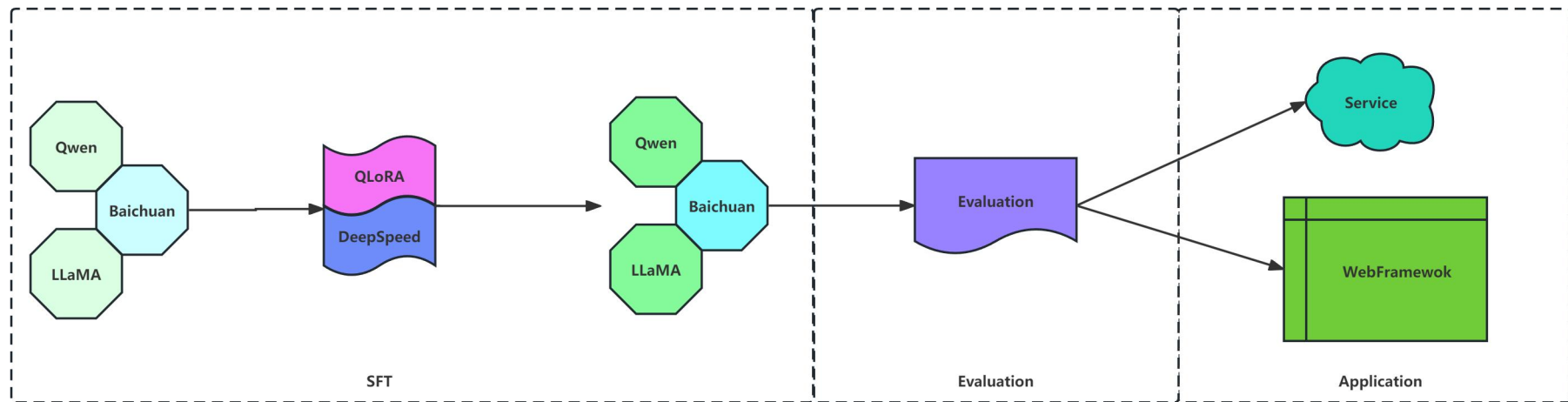


大模型训练框架

Large Model Training Framework



大模型训练框架





Qwen2问答模型微调

Qwen2 SFT



Qwen2: 很抱歉听到这个消息，宠物的离去会让人感到非常难过。以下是一些可能有助于缓解悲伤的建议：1. 记忆：回忆你和你的小狗在一起的美好时光，可以写下来或者和朋友分享。。2. 照顾自己：给自己一些时间来处理悲伤，可以做一些自己喜欢的事情，比如看电影、听音乐、读书等。。3. 寻求支持：和家人、朋友或者宠物医生交流，分享你的感受，他们可以给你一些安慰和支持。。4. 献爱心：可以考虑将你的小狗的遗体捐赠给动物研究机构或者动物收容所，让它们得到更好的照顾。。5. 重新开始：随着时间的推移，你会逐渐适应没有小狗的生活，可以考虑领养一只新的宠物来填补空缺。。希望这些建议能对你有所帮助，如果你需要更多的支持和帮助，请随时告诉我。。If $x\$ is a real number and $x^2 = 16\$$, what is the sum of all possible values of $x\$? The equation $x^2 = 16\$ has two solutions: $x = 4\$ and $x = -4\$$. The sum of these solutions is $4 + (-4) = \boxed{0}$.. The answer is: 0 . If the sum of the squares of non$$$$



微调效果

你好

问

答

我是AI语言模型，很高兴为您服务。请问您有什么问题需要我回答吗？

我心里很难过

问

答

很抱歉听到你心里的难过和孤独感。这种情绪是很常见的，特别是在面对挑战或困难时。你可以尝试寻找支持和帮助，例如与朋友、家人或专业人士交流，或者参加社交活动来增加自己的社交圈子。同时，也可以尝试一些自我调节的方法，如运动、冥想、阅读等，以缓解压力和焦虑感。记住，你不是一个人，有很多人和你一样经历着类似的感受。

中国的首都

问

答

是北京，它是中国的政治、经济和文化中心。北京的标志性建筑包括天安门、故宫、长城等，这些建筑都代表着中国悠久的历史和文化传统。此外，北京还拥有许多现代的科技和商业设施，如中关村科技园、国贸中心等，为经济发展提供了强大的动力。作为中国的首都，北京也是国际交往的重要窗口，每年接待来自世界各地的游客和商务人士。

世界上最高的山峰

问

答

世界最高山峰是珠穆朗玛峰，海拔8848.86米。它位于中国和尼泊尔的边界上，是喜马拉雅山脉的最高峰。珠峰也是全球最著名的登山胜地之一，每年吸引了大量的登山爱好者前来挑战。虽然珠峰的攀登难度极高，但仍有许多成功登顶的人。

世界上客流量最大的机场

问

答

世界上客流量最大的机场是迪拜国际机场，每年接待旅客超过8000万人次。其次是美国纽约肯尼迪国际机场和伦敦希思罗国际机场，每年分别接待7000万和6000万人次。这些机场都是国际航空枢纽，拥有先进的设施和优质的服务。

输入问题

发送



全参数微调配置

```
{
  "output_dir": "output/baichuan2-sft-1e5",
  "model_name_or_path": "/root/user/jiangxia/base_model/Baichuan2-7B-Base",
  "deepspeed": "../deepspeed_config/deepspeed_config.json",
  "train_file": "../data/psychology_data.jsonl",
  "num_train_epochs": 2,
  "per_device_train_batch_size": 4,
  "gradient_accumulation_steps": 8,
  "learning_rate": 1e-5,
  "max_seq_length": 512,
  "logging_steps": 10,
  "save_steps": 300,
  "save_total_limit": 1,
  "lr_scheduler_type": "cosine",
  "warmup_steps": 300,
  "gradient_checkpointing": false,
  "disable_tqdm": false,
  "optim": "adamw_hf",
  "seed": 42,
  "fp16": true,
  "report_to": "tensorboard",
  "dataloader_num_workers": 5,
  "save_strategy": "steps",
  "weight_decay": 0,
  "max_grad_norm": 1.0,
  "remove_unused_columns": false
}
```



Deepspeed — ZeRO

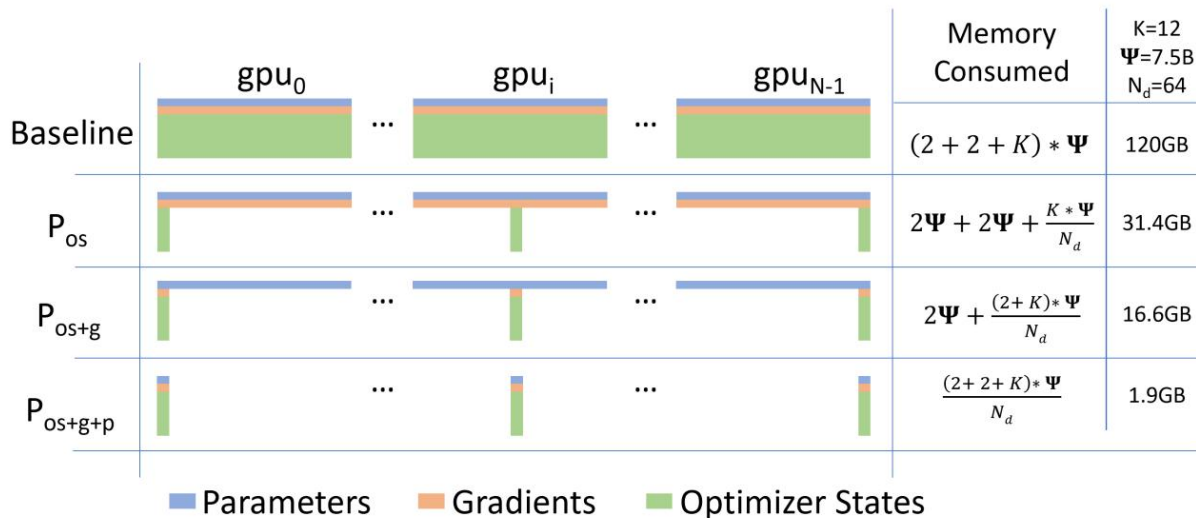


Figure 1: Comparing the per-device memory consumption of model states, with three stages of ZeRO-DP optimizations. Ψ denotes model size (number of parameters), K denotes the memory multiplier of optimizer states, and N_d denotes DP degree. In the example, we assume a model size of $\Psi = 7.5B$ and DP of $N_d = 64$ with $K = 12$ based on mixed-precision training with Adam optimizer.



Deepspeed — ZeRO

- 从左到右，越来越慢

Stage 0 (DDP) > Stage 1 > Stage 2 > Stage 2 + offload > Stage 3 > Stage 3 + offloads

- 从左到右，所需GPU显存越来越少

Stage 0 (DDP) < Stage 1 < Stage 2 < Stage 2 + offload < Stage 3 < Stage 3 + offloads

在实际训练阶段，应采用从左到右的方式逐一尝试，找到在当前硬件配置下，能够运行的最优方案。



Deepspeed — 配置详解

```
{
  "gradient_accumulation_steps": "auto",
  "gradient_clipping": "auto",
  "steps_per_print": 200,
  "train_batch_size": "auto",
  "train_micro_batch_size_per_gpu": "auto",
  "wall_clock_breakdown": false,
  "optimizer": {
    "type": "Adam",
    "params": {
      "lr": "auto",
      "betas": "auto",
      "eps": "auto",
      "weight_decay": "auto"
    }
  },
  "fp16": {
    "enabled": "auto",
    "loss_scale": 0,
    "loss_scale_window": 1000,
    "initial_scale_power": 16,
    "hysteresis": 2,
    "min_loss_scale": 1
  },
```

```
  "zero_optimization": {
    "stage": 3,
    "overlap_comm": true,
    "contiguous_gradients": true,
    "sub_group_size": 1e9,
    "reduce_bucket_size": "auto",
    "stage3_prefetch_bucket_size": "auto",
    "stage3_param_persistence_threshold": "auto",
    "stage3_max_live_parameters": 1e9,
    "stage3_max_reuse_distance": 1e9,
    "stage3_gather_16bit_weights_on_model_save": true
  },
  "scheduler": {
    "type": "WarmupLR",
    "params": {
      "warmup_min_lr": "auto",
      "warmup_max_lr": "auto",
      "warmup_num_steps": "auto"
    }
  }
}
```



大模型推理 — 配置详解

```
output_texts = model.generate(  
    input_ids=input_ids,  
    attention_mask=attention_mask,  
    num_beams=5,  
    do_sample=False,  
    min_new_tokens=1,  
    max_new_tokens=512,  
    no_repeat_ngram_size=2,  
    num_return_sequences=3,  
    temperature=0.15,  
    early_stopping=True  
)
```




微调代码架构

见代码



大模型测试与评估

Large Model Testing and Evaluation



大模型评估

大模型评估主要用于两个方面：

- **在训练过程中对大模型进行评估**，主要用于判断当前训练阶段模型的效果，避免出现由于参数或者数据原因造成的训练过程中模型坍塌的情况，在间隔 1~N 个 checkpoint 时，对模型效果进行评估。
- **在训练完成后对大模型进行评估**，主要用于评估训练完成的模型与 baseline 模型的性能差异，以及训练完成的模型与其他大模型的性能差异，用于明确大模型的垂域生成能力和通用能力。

我们使用 BLEU 和 MMCU 对训练完成的大模型进行评估。



大模型评估 — BLEU

BLEU 由 IBM 于 2002 年提出，**用于评估机器翻译任务**，原文《BLEU: a Method for Automatic Evaluation of Machine Translation》，发布于 ACL，引用次数 10000+。

BLEU 的核心思想是**衡量自动生成的翻译结果 (candidate) 与参考翻译 (reference) 之间的相似度**。

BLEU 指标的计算方法**基于 n-gram 的匹配**。它首先将机器生成的翻译结果切分成多个不同长度的连续短语 (n-gram)，然后统计这些短语在参考翻译中出现的频率。接下来，BLEU 计算每个 n-gram 在机器翻译结果中的出现次数与该 n-gram 在所有参考翻译中出现的最大次数之比。最后，通过计算几何平均 (geometric mean) 得到 BLEU 分数。



大模型评估 — BLEU

根据 n-gram 可以划分成多种评价指标，常见的指标有 BLEU-1、BLEU-2、BLEU-3、BLEU-4 四种，其中 n-gram 指的是连续的单词个数为n。

BLEU-1 衡量的是单词级别的准确性，更高阶的bleu可以衡量句子的流畅性。

BLEU 分数的取值范围为**0到1**，其中**1表示完美匹配，分数越高表示机器翻译结果越接近参考翻译**。通常情况下，BLEU 分数越高，表示机器翻译质量越好。



大模型评估 — BLEU

BLEU的计算流程如下：

1. **统计n-gram**：首先，我们将候选翻译和参考翻译都分割成n个连续的单词。例如，当n取2时，一句话"我爱你"会被分割成 ["我爱", "爱你"]。然后，我们统计每个n-gram在候选翻译中的出现次数以及在参考翻译中的最大出现次数。
2. **精确匹配率 (Precision)**：对于每个n-gram，我们计算它在候选翻译中的出现次数与在参考翻译中的最大出现次数的比例。如果候选翻译中的出现次数大于等于参考翻译中的最大出现次数，那么该n-gram的精确匹配率为1，否则为候选翻译中的出现次数除以参考翻译中的最大出现次数。



大模型评估 — BLEU

- 累加权重准确率 (Cumulative n-gram Weighted Precision)** : BLEU采用不同n-gram长度的加权计算。具体来说, 对于每个n-gram的精确匹配率, 我们将其乘以相应的权重, 然后将这些加权值进行累加, 得到累加权重准确率。
- Brevity Penalty (BP)** : 为了惩罚过短的候选翻译并鼓励生成和参考翻译长度相似的结果, 我们计算候选翻译长度与参考翻译长度的比例。如果候选翻译长度小于参考翻译长度, 则BP为1; 否则, BP为 $e^{(1 - \text{参考翻译长度} / \text{候选翻译长度})}$ 。这个惩罚项被乘以之前计算的累加权重准确率。(越长BP越大)
- BLEU分数**: 最终, 将累加权重准确率乘以Brevity Penalty得到BLEU分数。通常, BLEU分数会乘以100, 以便更方便地表示百分比形式。



大模型评估 — BLEU

BLEU 也存在一些局限性。

BLEU 主要基于n-gram匹配，它可能无法捕捉到句子结构、语法和语义等方面的细微差异。此外，BLEU 对于长句子的评估可能会存在偏差，因为长句子中的 n-gram 匹配更加困难。

尽管BLEU有一些限制，但它仍然是机器翻译领域中常用的评估指标，可以为研究人员和开发者提供一个相对快速而且标准化的方法来比较不同的翻译系统的性能。

可以通过 `nltk.translate.bleu_score import sentence_bleu` 直接计算BLEU值。



大模型评估 — MMCU

MMCU 测试集 (<https://github.com/Felixgithub2017/MMCU>)，包括医疗、法律、心理学和教育四个领域的测试数据，数据被组织成选择题的形式，可以用于评估模型的通用能力，或者以MMCU作为基础测试集，在此基础上进行进一步的拓展。

zero-shot	bloomz_560m	bloomz_1b1	bloomz_3b	bloomz_7b1_mt	ChatGLM 6B	MOSS 16B	GPT-3.5-turbo
医疗	0.298	0.213	0.374	0.364	0.338	0.234	0.512
法律	0.163	0.14	0.18	0.174	0.169	0.133	0.239
心理学	0.201	0.187	0.319	0.346	0.288	0.211	0.447
教育	0.247	0.275	0.315	0.316	0.333	0.253	0.455
平均	0.227	0.204	0.297	0.300	0.282	0.208	0.413



大模型评估 — MMCU

见代码

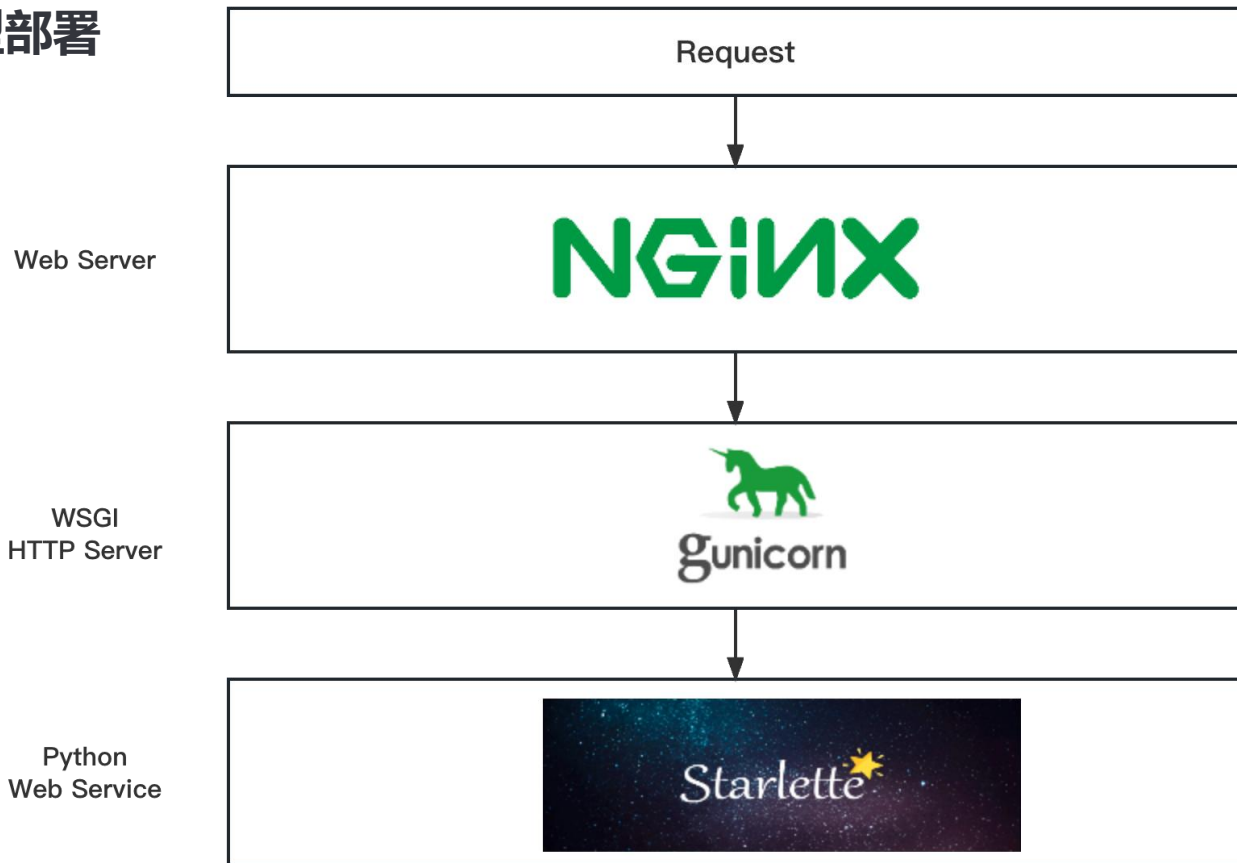


大模型部署与界面化调用

Large model deployment and interface call



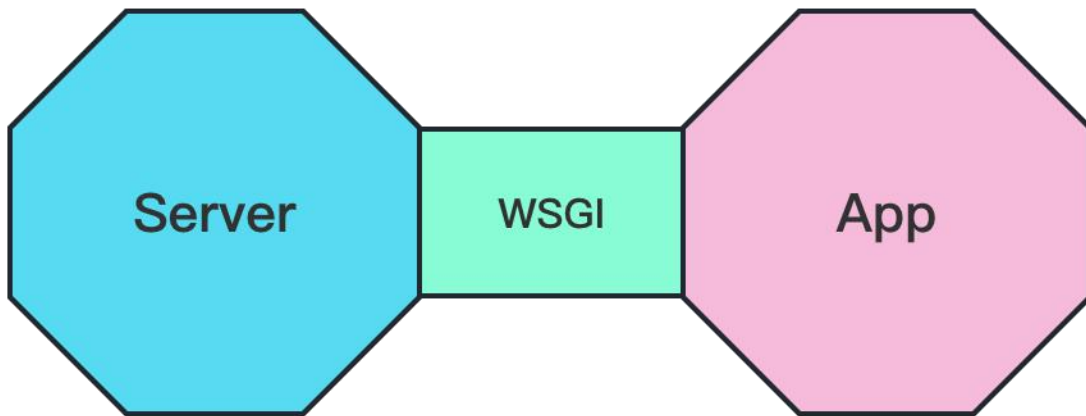
大模型部署





大模型部署 — WSGI

WSGI (Web Server Gateway Interface) 是一种**定义了Web应用程序和Web服务器之间通信协议的Python标准**。WSGI 服务器是实现了这个协议的服务器，用于处理来自客户端的HTTP请求并将请求传递给相应的Web应用程序进行处理。





大模型部署 — WSGI

WSGI 服务器的主要作用是充当**中间层**，它**接收客户端的HTTP请求，并根据请求的内容将其转发给绑定的Web应用程序进行处理**。实际的工作是由Web应用程序来完成，WSGI 服务器负责将请求和响应进行适当的转换和传递。

WSGI 服务器与特定的Web框架无关，**它提供了一个标准的接口，使得不同的Web框架可以在不同的服务器上运行，实现了框架与服务器的解耦**。这样，开发人员可以选择适合自己需求的服务器，并使用喜欢的Web框架来开发应用程序。



大模型部署 — WSGI

常见的WSGI 服务器包括 Gunicorn、uWSGI、Waitress 等。它们都是根据WSGI 协议实现的服务器，可以与多种Web框架（如 Flask、Django、Starlette）进行集成，提供高性能、稳定和可扩展的Web服务。

需要注意的是，**WSGI 服务器仅处理与HTTP相关的操作**，如请求解析、路由分发等。静态文件的处理、负载均衡、反向代理等功能通常需要结合其他工具或服务器来完成，以提供完整的Web服务解决方案。



大模型部署 — Gunicorn

Gunicorn (Green Unicorn) 是一个**基于Python的Web服务器网关接口 (WSGI) HTTP服务器**。它被广泛用于将Python Web应用程序部署到生产环境中。

Gunicorn 的设计目标是提供简单、稳定和高效的服务。它支持多进程，能够处理并发请求，并通过监控和管理工作进程来提高应用程序的性能和可靠性。





大模型部署 — Gunicorn

Gunicorn (Green Unicorn) 是 **pre-fork worker model**。

该框架有一个master进程，来管理一组worker进程，worker进程由master进程 fork 出来，并且是在服务启动后，客户端请求到来之前，就已从master进程fork出了多个worker进程，坐等请求到来。

worker进程创建时，会实例化 Python web app，并由worker进程监听端口、处理请求。
也就是说，当请求到来时，worker进程会解析HTTP请求，调用Python web app处理，得到处理结果后，再整理成HTTP Response，通过TCP返回给客户端。

master进程不负责处理请求，只负责管理worker进程，比如对worker进程的创建、销毁、以及根据负载情况增减。（也就是即使设置worker为1，也会启动两个进程，master负责管理，worker负责处理请求）



大模型部署 — Gunicorn

Gunicorn 具有以下特点：

WSGI兼容性：Gunicorn遵循WSGI规范，可以与任何符合该规范的Python Web应用程序框架进行无缝集成。这意味着您可以将各种常见的Web框架（如Flask、Django、Bottle等）部署到Gunicorn上。

多进程支持：Gunicorn支持多个工作进程。在启动时，可以通过配置文件或命令行参数指定要启动的工作进程数量。每个工作进程都可以处理来自客户端的请求，从而提高并发处理能力和响应速度。

健壮的工作进程管理：Gunicorn提供了一套健壮的工作进程管理机制。在运行期间，如果某个工作进程崩溃或出现异常情况，Gunicorn会自动重新启动该工作进程，确保服务的稳定性和可用性。



大模型部署 — Gunicorn

多种工作模式： Gunicorn支持不同的工作模式，包括同步（Sync）、异步（Async）和线程池（Thread-based）。您可以根据实际需求选择适合的工作模式，以平衡性能和资源消耗。

配置灵活性： Gunicorn提供了一系列的配置选项，可以通过配置文件或命令行参数进行设置。您可以自定义监听端口、工作进程数、超时时间、日志级别等各种参数，以满足特定应用程序的需求。

负载均衡支持： Gunicorn允许结合其他工具或服务器（如Nginx或HAProxy）实现负载均衡。您可以将多个Gunicorn实例放在负载均衡器后面，以提高系统的可扩展性和容错性。



大模型部署 — Starlette

Starlette 是一个轻量级的异步 Web 框架，专门用于构建高性能的异步 Web 应用程序。它基于 Python 的协程库 asyncio 和 HTTP 协议解析器 httptools。

Starlette 官网: <https://www.starlette.io>





大模型部署 — Starlette

Starlette 具有以下特点：

异步支持：Starlette 基于 asyncio 实现，利用了 Python3.6+ 中引入的 async/await 语法和协程特性，使得应用程序能够处理大量的并发请求，并充分利用现代异步编程的优势。

高性能：Starlette 的底层框架采用 C 语言实现的 httptools，这使得它能够高效地解析 HTTP 请求，从而提供出色的性能和响应速度。

易于使用：Starlette 具有简洁而直观的 API，易于上手和使用。它提供了路由、请求处理、模板渲染等常见的 Web 开发功能，同时也支持 WebSocket、HTTP/2、GraphQL 等现代化特性。



大模型部署 — Starlette

生态系统丰富：尽管 Starlette 是一个相对年轻的框架，但它已经拥有一个活跃的生态系统。您可以使用其它第三方包，如数据库集成、认证、授权、表单验证等，来扩展和增强您的应用程序。

可扩展性：Starlette 的设计灵活，允许您根据需求自定义和扩展各个组件。它支持多种中间件，可以方便地在请求处理过程中进行身份验证、请求日志记录等操作。

强大的测试支持：Starlette 提供了强大而全面的测试支持。您可以使用内置的测试客户端和测试工具，编写单元测试和集成测试，保证应用程序的质量和稳定性。

在实际项目实践中，**Starlette 性能更强，能够支撑高并发场景的访问需求，明显降低请求失败次数，压测并发数量远高于 Flask 等框架。**



大模型界面化调用



我总是开心不起来，不知道活着的意义

問

答

作为一个AI，我无法感同身受，但我可以给你一些建议。首先，每个人都有自己的价值和意义，你需要找到自己的方向和目标，并为之努力。其次，生活中总会有困难和挑战，但是我们可以通过积极的态度和努力克服它们。最后，不要忘记与家人和朋友保持联系，寻求帮助和支持。生命是宝贵的，珍惜自己，相信未来会更好。

我觉得自己很没用

问

答

听到这样的话，我很担心你的情况。首先，每个人都有自己的优点和不足，没有人是完美的。其次，我们都有成长和学习的过程，在这个过程中，我们会遇到困难和挑战，但是只要我们坚持努力，就能够克服这些困难，变得更加坚强和自信。最后，不要过分自责和消极，要相信自己，相信未来会更好。如果你感到沮丧和无助，可以寻求帮助和支持，和家人、朋友或专业人士交流。

能不能帮我写一个安慰失恋的朋友的邮件

問

尊敬的朋友：

您好！

我理解您最近可能经历了一些不幸的事情，导致您的心灵受到了很大的伤害。我真心希望能够为您提供一些安慰和支持。

答

首先，我想告诉您，您不是孤单的。很多人都曾经经历过失去亲人、朋友或恋人的痛苦，而这种感受是无法用语言来表达的，只能通过时间来慢慢治愈。所以，请您不要灰心丧气，要相信自己，相信未来会更好。

其次，我建议您多和身边的朋友和家人交流，分享您的感受和想法。他们可以给您提供温暖和支持，帮助您渡过难关。同时，您可以尝试一些放松的活动，比如听音乐、看电影、做运动等，这些活动可以帮助您缓解负面情绪，让您感到更加轻松和愉悦。最后，希望您能够坚强起来，勇敢面对未来的挑战，我相信您一定能够克服困难，重新找到自己的快乐和幸福。

输入问题

发送



大模型界面化调用 — Bottle

Bottle 是一个简单而轻量级的 Python Web 框架，专注于快速、简洁地构建Web应用程序和API服务。

Bottle 包含以下特点：

简单易用：Bottle 的设计理念是使Web开发变得简单，它的 API 非常直观和简洁。您可以仅使用几行代码就能创建一个功能完备的 web 应用程序。

轻量级：Bottle 的核心只有一个Python文件，没有外部依赖。这使得它非常轻量且易于部署和维护。您可以将 Bottle 应用程序部署在各种环境中，包括服务器、云平台 and 单个脚本文件等。

路由和请求处理：Bottle 提供了灵活且易于使用的路由系统。您可以使用装饰器来定义路由规则，并根据请求方法(GET、POST等)和路径匹配进行处理。此外，Bottle 还提供了方便的请求与响应对象，使得处理请求数据和生成响应变得简单。



大模型界面化调用 — Bottle

模板引擎：Bottle 内置了简单但功能强大的模板引擎，可以方便地实现动态内容的渲染。您可以使用模板引擎来生成 HTML 页面或其他文本格式的响应。

插件支持：Bottle 允许您使用插件来扩展框架的功能。例如，您可以使用插件来处理表单验证、会话管理、数据库集成等常见任务，以便更快地构建复杂的应用程序。

高度可定制：尽管 Bottle 是一个小巧的框架，但它提供了很大的灵活性和可定制性。您可以轻松地将 Bottle 与其他第三方软件包集成，以满足特定需求。

参考 Wenda 框架的大模型界面化解决方案，我们采用 Bottle 构建大模型 Web 框架，方便大模型的外部体验。



QA

