# Large Language Model

绪论

2017年，一支由Google公司8名青年工程师和科学家组成的团队发表了论文《Attention Is All You Need》，首次提出了Transformer模型，Transformer 完全基于注意力机制，取代了以往序列到序列模型中常用的循环神经网络（RNN）和卷积神经网络（CNN），在自然语言处理（NLP）领域产生了深远影响，Transformer架构具有的众多优势使其成为了构建大语言模型的理想选择。

2023年1月23日，OpenAI公司发布ChatGPT-3.5免费开放使用，LLM第一次走入公众视野，ChatGPT-3.5具有强大的自然语言理解和生成能力，能够进行多轮对话，引发了又一次AI热潮。

2023年7月18日，Meta发布了开源大语言模型Llama2，包含多个尺寸的版本。Llama的发布打破了大型科技公司在AI领域的技术垄断，使得更多人能够接触和使用先进的 AI 技术，降低了技术门槛，促进了 AI 技术的普及和应用。这为学术界、工业界和个人开发者提供了更多机会，推动了 AI 技术的快速发展。

2023年12月7日，Google发布了多模态大语言模型Gemini，不仅支持文本任务，还支持语音、图像、视频等多种文件输入，轻松进行转换。

智能客服、手机助手、自动驾驶、全屋智能，AI已经融入我们生活中的方方面面，随着AI的发展如火如荼，各大语言模型功能愈发强大与智能，全球头部科技公司的AI竞赛也到了白热化阶段，AI技术迎来了新的机遇和挑战，AGI(通用人工智能)的实现将不再遥远。

# 第一章 大语言模型基础

## 1.1 常见大模型

### 1.1.1 商业大模型

商业大模型指非开源、具有营利性的大语言模型。

ChatGPT：OpenAI

Claude：Anthropic

Gemini：Google

### 1.1.2 开源大模型

Llama：Meta

Gemma：Google

Bloom：Hugging Face

## 1.2 大模型使用-初阶：Ollama本地部署大模型

Ollama中包含数量众多的预训练模型，一键部署极大的降低了大模型本地使用的门槛，推理完全依赖本地设备硬件无需互联网具有极强的隐私性，同时对部署设备的硬件有一定要求，适合生产环境应用。

进入Ollama官网点击Download，选择符合自己系统的版本，Windows配置环境变量`OLLAMA\_MODELS`可指定下载模型的保存位置。

官网主页点击右上角Models，搜索想要部署的模型点击进入，复制右上角命令进入cmd执行即可。

详细步骤不在此进行赘述，请参考如下网址：

<https://blog.csdn.net/weixin_43012017/article/details/138253446>

# 第二章 大模型使用-进阶:Hugging Face

Hugging Face是一家致力于自然语言处理(NLP)和人工智能的科技公司，其开源的NLP库和工具使得在学术界和工业界广受欢迎，本章着重讲自然语言处理库Transformers。

## 2.1 Transformers简介

为 PyTorch、TensorFlow 和 JAX 打造的先进的机器学习工具。

Transformers 提供了可以轻松地下载并且训练先进的预训练模型的 API 和工具。使用预训练模型可以减少计算消耗和碳排放，并且节省从头训练所需要的时间和资源。这些模型支持不同模态中的常见任务，比如：

* 自然语言处理：文本分类、命名实体识别、问答、语言建模、摘要、翻译、多项选择和文本生成。
* 机器视觉：图像分类、目标检测和语义分割。
* 音频：自动语音识别和音频分类。
* 多模态：表格问答、光学字符识别、从扫描文档提取信息、视频分类和视觉问答。

Transformers 支持在 PyTorch、TensorFlow 和 JAX 上的互操作性. 这给在模型的每个阶段使用不同的框架带来了灵活性；在一个框架中使用几行代码训练一个模型，然后在另一个框架中加载它并进行推理。模型也可以被导出为 ONNX 和 TorchScript 格式，用于在生产环境中部署。

Transformer支持任务文档：

https://huggingface.co/docs/transformers/v4.42.0/en/task\_summary#object-detection

安装与快速上手参考以下链接：

https://huggingface.co/docs/transformers/v4.40.2/zh/installation

<https://huggingface.co/docs/transformers/v4.40.2/zh/quicktour>

Github 地址：https://github.com/huggingface/transformers

### 2.1.1 pipeline

Pipeline是Transformers中一个高度封装的API，简化了使用模型进行推理的方法。这些pipelines是抽象了库中大部分复杂代码的对象，提供了一个专用于多个任务的简单API，包括文本分类、掩码语言建模、情感分析、特征提取和问答等。请参阅[任务摘要](https://huggingface.co/docs/transformers/v4.42.0/zh/task_summary)以获取使用示例。快速获取请参考表2.1-1：

|  |  |  |  |
| --- | --- | --- | --- |
| 任务 | 描述 | 模态 | Pipeline |
| 文本分类 | 为给定的文本序列分配一个标签 | NLP | pipeline(task=“sentiment-analysis”) |
| 文本生成 | 根据给定的提示生成文本 | NLP | pipeline(task=“text-generation”) |
| 命名实体识别 | 为序列里的每个 token 分配一个标签（人, 组织, 地址等等） | NLP | pipeline(task=“ner”) |
| 问答系统 | 通过给定的上下文和问题, 在文本中提取答案 | NLP | pipeline(task=“question-answering”) |
| 掩盖填充 | 预测出正确的在序列中被掩盖的token | NLP | pipeline(task=“fill-mask”) |
| 文本摘要 | 为文本序列或文档生成总结 | NLP | pipeline(task=“summarization”) |
| 文本翻译 | 将文本从一种语言翻译为另一种语言 | NLP | pipeline(task=“translation”) |
| 图像分类 | 为图像分配一个标签 | Computer vision | pipeline(task=“image-classification”) |
| 图像分割 | 为图像中每个独立的像素分配标签（支持语义、全景和实例分割） | Computer vision | pipeline(task=“image-segmentation”) |
| 目标检测 | 预测图像中目标对象的边界框和类别 | Computer vision | pipeline(task=“object-detection”) |
| 音频分类 | 给音频文件分配一个标签 | Audio | pipeline(task=“audio-classification”) |
| 自动语音识别 | 将音频文件中的语音提取为文本 | Audio | pipeline(task=“automatic-speech-recognition”) |
| 视觉问答 | 给定一个图像和一个问题，正确地回答有关图像的问题 | Multimodal | pipeline(task=“vqa”) |
| Document question answering | answer a question about the document, given a document and a question | Multimodal | pipeline(task=“document-question-answering”) |
| Image captioning | generate a caption for a given image | Multimodal | pipeline(task=“image-to-text”) |

表2.1-1 pipeline支持任务列表

有两种pipelines抽象类需要注意：

pipeline()，它是封装所有其他pipelines的最强大的对象。

针对特定任务pipelines，适用于[音频](https://huggingface.co/docs/transformers/v4.42.0/zh/main_classes/pipelines#audio)、[计算机视觉](https://huggingface.co/docs/transformers/v4.42.0/zh/main_classes/pipelines#computer-vision)、[自然语言处理](https://huggingface.co/docs/transformers/v4.42.0/zh/main_classes/pipelines#natural-language-processing)和[多模态](https://huggingface.co/docs/transformers/v4.42.0/zh/main_classes/pipelines#multimodal)任务。

Pipeline还支持自定义，详细内容见：<https://huggingface.co/docs/transformers/v4.42.0/zh/main_classes/pipelines#pipelines>

Kaggle Jupyter Notebook: [code\LLM\_NO.1\_pipelines.ipynb](code/LLM_NO.1_pipelines.ipynb)

#### [实现一个新的pipeline](https://huggingface.co/docs/transformers/v4.42.0/zh/add_new_pipeline)

首先继承基类 Pipeline，其中包含实现 preprocess、\_forward、postprocess 和 \_sanitize\_parameters 所需的 4 个方法。

### 2.1.2 AutoClass

在幕后，是由 AutoModelForSequenceClassification 和 AutoTokenizer 一起支持你在上面用到的 pipeline()。AutoClass 是一个能够通过预训练模型的名称或路径自动查找其架构的快捷方式。你只需要为你的任务选择合适的 AutoClass 和它关联的预处理类。

**使用AutoClass加载预训练实例**

由于存在许多不同的Transformer架构，因此为您的checkpoint创建一个可用架构可能会具有挑战性。通过AutoClass可以自动推断并从给定的checkpoint加载正确的架构, 这也是🤗 Transformers易于使用、简单且灵活核心规则的重要一部分。from\_pretrained()方法允许您快速加载任何架构的预训练模型，因此您不必花费时间和精力从头开始训练模型。生成这种与checkpoint无关的代码意味着，如果您的代码适用于一个checkpoint，它将适用于另一个checkpoint - 只要它们是为了类似的任务进行训练的 - 即使架构不同。