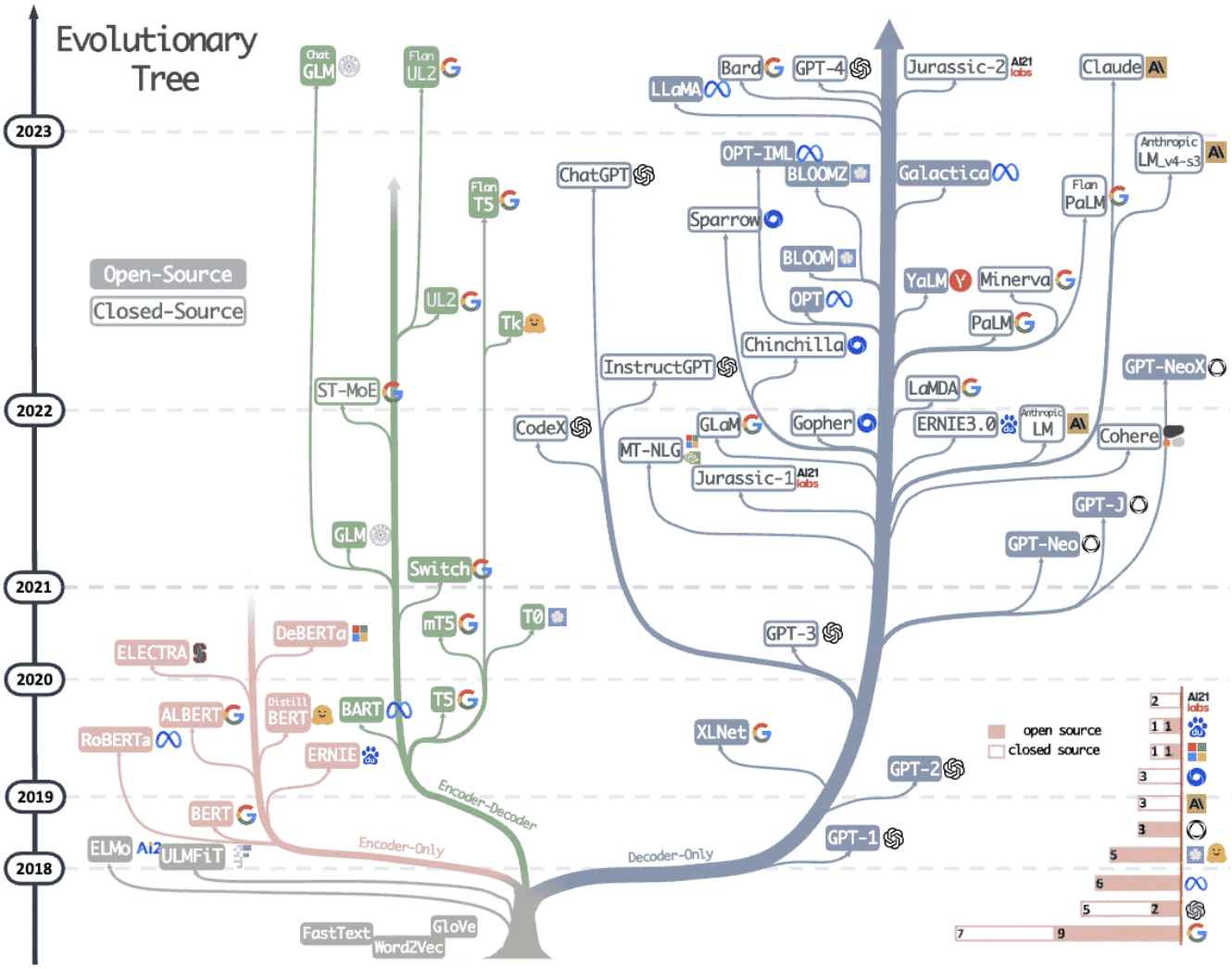


Fig. 1. The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models. Models on the same branch have closer relationships. Transformer-based models are shown in non-grey colors: decoder-only models in the blue branch, encoder-only models in the pink branch, and encoder-decoder models in the green branch. The vertical position of the models on the timeline represents their release dates. Open-source models are represented by solid squares, while closed-source models are represented by hollow ones. The stacked bar plot in the bottom right corner shows the number of models from various companies and institutions.

图来自2023年初的综述：https://arxiv.org/pdf/2304.13712

了解大模型最开始需要了解从Transformer衍生而来的三种主要分支(现在可能主要是两种分支了)，初学者需要了解 Transformer（Attention Is All You Need）、Encoder-only的Bert、Decoder-only的GPT。

## Transformer

> Attention Is All You Need. https://arxiv.org/pdf/1706.03762

最经典论文，很多问题在其中都有答案：

Attention是先分头还是先线性变换?

Attention map为啥要Scale?

既然 $attn\_map = softmax(QW_q(KW_k)^T \cdot scale) = softmax(Q(W_qW_k^T)K^T \cdot scale)$，为什么不直接把 $W_qW_k^T$ 合并成一个可训练矩阵?

学习的时候可以参考一些github的代码实现：https://github.com/jadore801120/attention-is-all-you-need-pytorch

以及参考hf上的一些博客：https://huggingface.co/blog/Esmail-AGumaan/attention-is-all-you-need

- 面试常考点

encoder和decoder模块区别、encoder和decoder计算模式区别（mask区别）、各个模块手撕（细节很多，建议自己实现一下并且和一些别人的实现比较输出）

## Bert

> BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/pdf/1810.04805

**Bert**

bert是transformers encoder-only发展分支上的工作。

需要了解模型结构、预训练任务MLM和NSP

实现参考huggingface：https://github.com/huggingface/transformers/blob/main/src/transformers/models/bert/modeling_bert.py

要想试试在一些下游任务上bert的表现参考hf的api：https://huggingface.co/docs/transformers/en/model_doc/bert，建议不仅要试，还要仔细看具体是用什么分类头？用哪些embedding这些细节?

**后续工作**

了解后续的一些工作（deberta等）：https://github.com/huggingface/transformers/blob/main/src/transformers/models/deberta/modeling_deberta.py

**sentence-bert**

一个重要的问题：bert提取的embedding如何用到下游？是用CLS Token Embedding还是Pooling Context Embedding？是否可以直接计算两个embedding的cosine来计算语义相似度？（当然是不能）

sentence-bert使得原始bert能够克服各向异性(Anisotropic，https://blog.csdn.net/sunghosts/article/details/128616424)，结果是能使用cosine计算词句之间的语义相似度。

了解一些目前比较常用的embedding模型(gte、bge、mgte)，这也是了解RAG的基础。

**sgpt**

> https://arxiv.org/pdf/2202.08904

用decoder-only的LLM来生成语义embedding的工作（加权每个position的embedding），现在包括Qwen在内也有很多用LLM来embedding的。

- 考点

主要需要了解bert基本结构以及怎么用在下游任务（分类、匹配等，bert仍然是打工过程中比较重要的模型，现在

一般是用gte这种做语义召回）、sentence-bert怎么训已经怎么用？手撕模块?

## decoder-only LLM

这个就是目前最主流的大模型chatbot，就是我们最常用的GPT系列。

需要了解的概念：自回归生成、掩码、KV cache、位置编码、训练（PT和SFT（几种常用的Parameter-Efficient Fine-Tuning）、RLRF）、上下文窗口。

一个问题：LLM的基本模块是否就是Transformer的decoder block？有什么不同?

自回归生成：https://huggingface.co/docs/transformers/zh/llm_tutorial

KV cache：https://kylinchen.cn/2023/08/21/KVCache/

> 视频：*https://www.youtube.com/watch?v=80bIUggRJf4*

- 训练侧

**预训练**：需要了解Task(loss 设计)，mask，Teacher-forcing技术、优化器、显存占用计算、混合精度训练

**SFT**：mask和预训练有什么区别？loss设计。Parameter-Efficient Fine-Tuning的几种框架

> 试试在自己的卡上能训练的nanogpt：https://github.com/karpathy/nanoGPT

**RLHF**：PPO、DPO

**框架**：deepspeed框架、3D parallel（data parallel、tensor parallel【Megatron】、pipeline parellel）、torchrun

- 推理侧

**解码方式**：*greedy decoding*、*multinomial sampling*、*beam-search*、*top-k*、*top-p*、*temperature*、*contrastive decoding*

> 文档：https://huggingface.co/docs/transformers/v4.28.1/en/main_classes/text_generation

> 源码：https://github.com/huggingface/transformers/blob/v4.28.1/src/transformers/generation/configuration_utils.py#L38

**Chain of Thought (CoT)**：通过设计多轮prompt利用LLM的zero-shot能力： https://blog.csdn.net/philosophyatmath/article/details/134273962

**Retrieval-Augmented Generation (RAG)**：通过将检索模块（Retriever）与生成模块（Generator）结合，解决了传统生成模型对外部知识的依赖问题：https://luxiangdong.com/2023/09/25/ragone/

**效率优化入门**：https://hugging-face.cn/docs/transformers/llm_optims

## 常见LLM模型的架构?

最好参考huggingface的transformers的实现

示例Qwen-2: https://github.com/huggingface/transformers/blob/main/src/transformers/models/qwen2/modeling_qwen2.py

或者huggingface model repo下面的modeling_xxx.py文件：

示例DeepSeek-V2.5：https://huggingface.co/deepseek-ai/DeepSeek-V2.5/blob/main/modeling_deepseek.py

- 面试常考点

模型架构差异、位置编码、FFN、激活的细节?