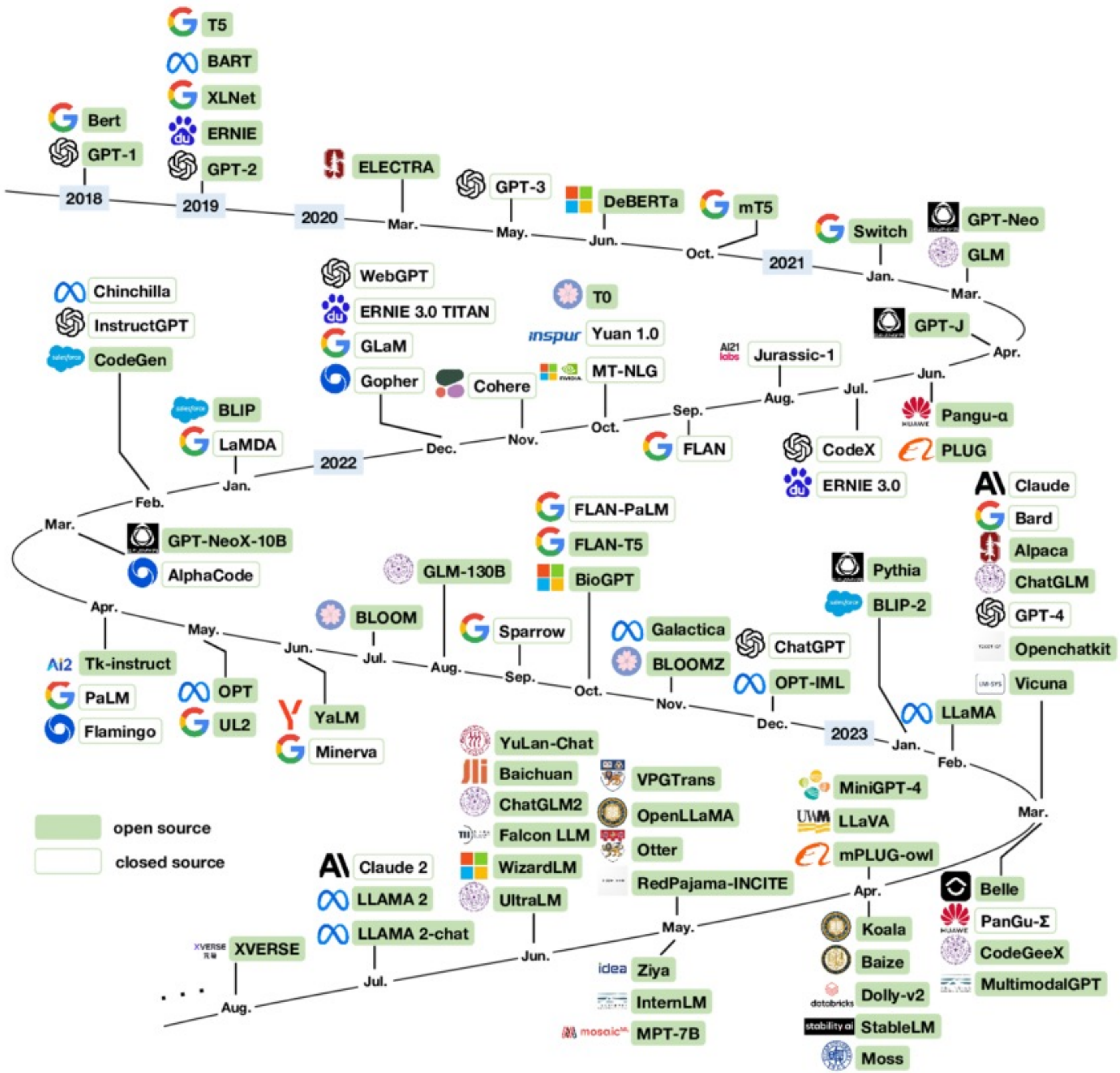
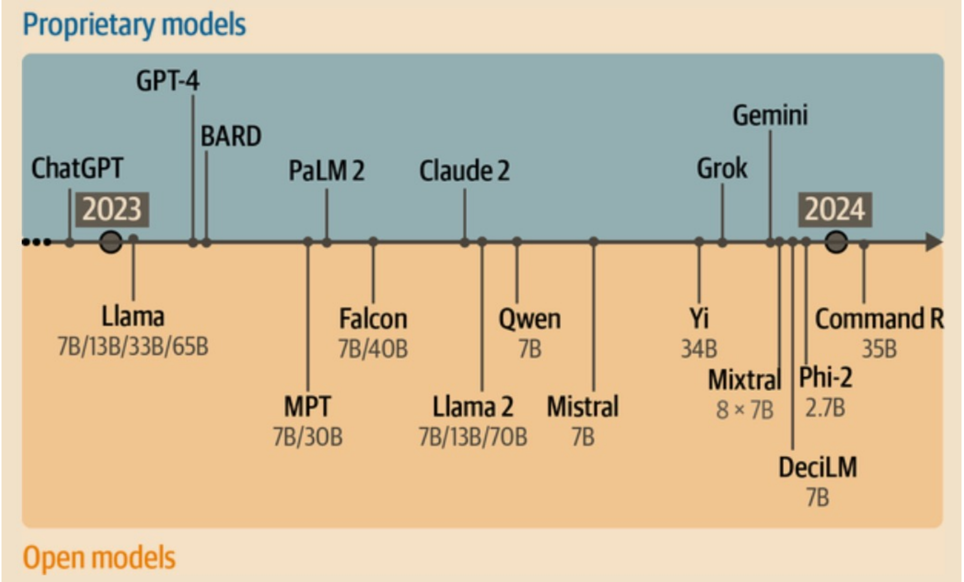
第六章 大语言模型架构


2018
Bert
GPT-1
2019
T5
BART
XLNet
ERNIE
GPT-2
2020
ELECTRA
Mar.
GPT-3
May.
DeBERTa
Jun.
mT5
Oct.
2021
Switch
Jan.
GPT-Neo
GLM
Mar.
GPT-J
Apr.
Jun.
Pangu-α
PLUG
Jurassic-1
Aug.
Jul.
CodeX
ERNIE 3.0
T0
Yuan 1.0
MT-NLG
Sep.
FLAN
Oct.
Nov.
Cohere
Dec.
WebGPT
ERNIE 3.0 TITAN
GLaM
Gopher
2022
Jan.
LaMDA
BLIP
Chinchilla
InstructGPT
CodeGen
Feb.
Mar.
GPT-NeoX-10B
AlphaCode
Apr.
Tk-instruct
PaLM
Flamingo
May.
OPT
UL2
Jun.
YaLM
Minerva
Jul.
BLOOM
Aug.
GLM-130B
Sep.
Sparrow
Oct.
FLAN-PaLM
FLAN-T5
BioGPT
Nov.
Galactica
BLOOMZ
Dec.
ChatGPT
OPT-IML
2023
Jan.
Pythia
BLIP-2
Feb.
LLaMA
Claude
Bard
Alpaca
ChatGLM
GPT-4
Openchatkit
Vicuna
Mar.
Belle
PanGu-Σ
CodeGeeX
MultimodalGPT
MiniGPT-4
LLaVA
mPLUG-owl
Apr.
Koala
Baize
Dolly-v2
StableLM
Moss
May.
Ziya
InternLM
MPT-7B
VPGTrans
OpenLLaMA
Otter
RedPajama-INCITE
Jun.
YuLan-Chat
Baichuan
ChatGLM2
Falcon LLM
WizardLM
UltraLM
Jul.
Claude 2
LLAMA 2
LLAMA 2-chat
Aug.
XVERSE
open source
closed source

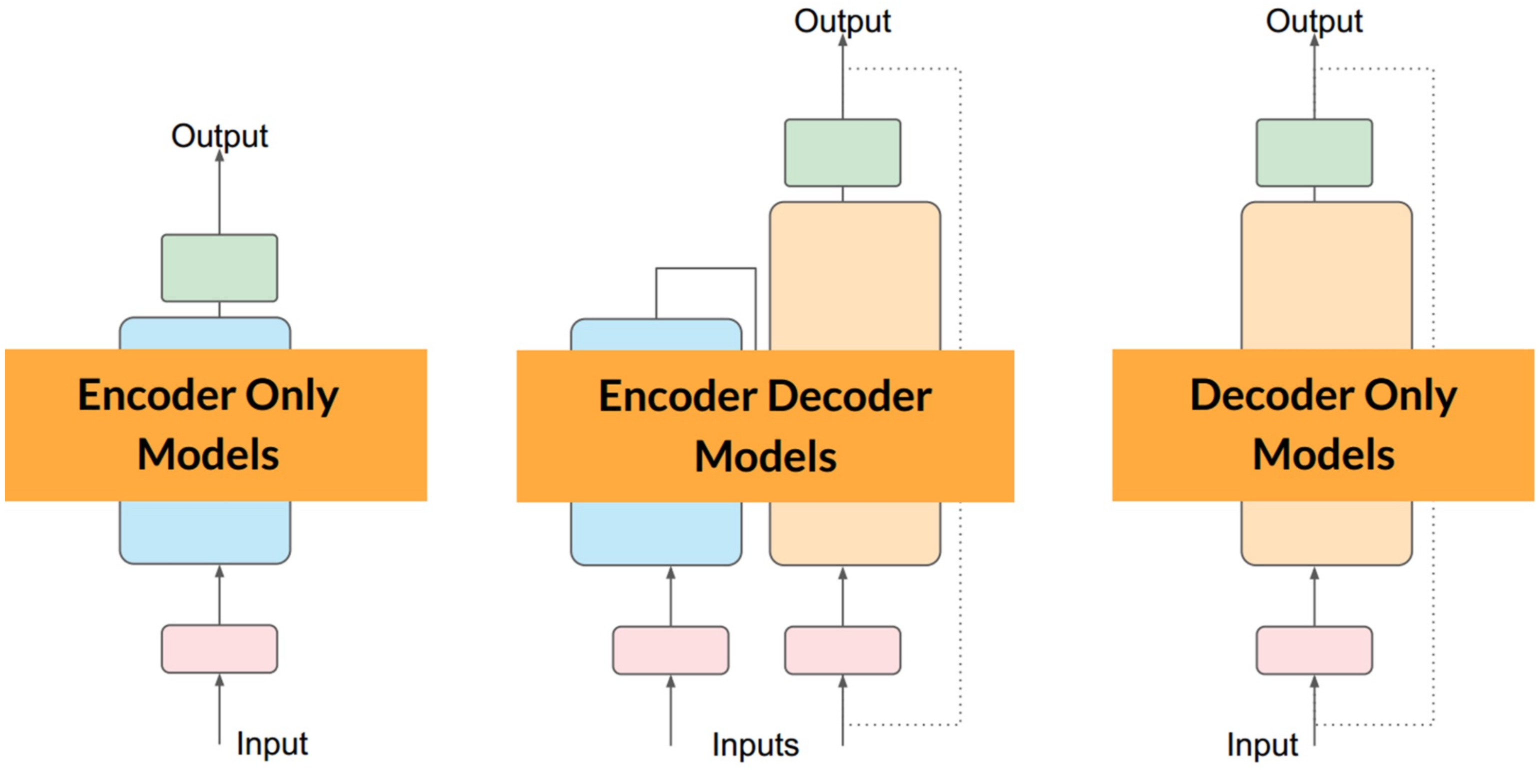

Proprietary models
GPT-4
BARD
ChatGPT
PaLM 2
Claude 2
Gemini
Grok
2023
2024
Llama
7B/13B/33B/65B
Falcon
7B/40B
Qwen
7B
Yi
34B
Command R
35B
MPT
7B/30B
Llama 2
7B/13B/70B
Mistral
7B
Mixtral
8 × 7B
Phi-2
2.7B
DeciLM
7B
Open models

目录

■1. 基于 Transformer 的模型架构

1.1 三种分类

Transformer 模型主要由两个核心部分构成：**编码器（Encoder）和解码器（Decoder）**。依托这两个关键组件的不同组合和应用，Transformer 模型发展出三种主流架构：**编码（Encoder-Only）大语言模型**、**解码（Decoder-Only）大语言模型**以及**编解码（Encoder-Decoder）大语言模型**。



■1. 基于 Transformer 的模型架构

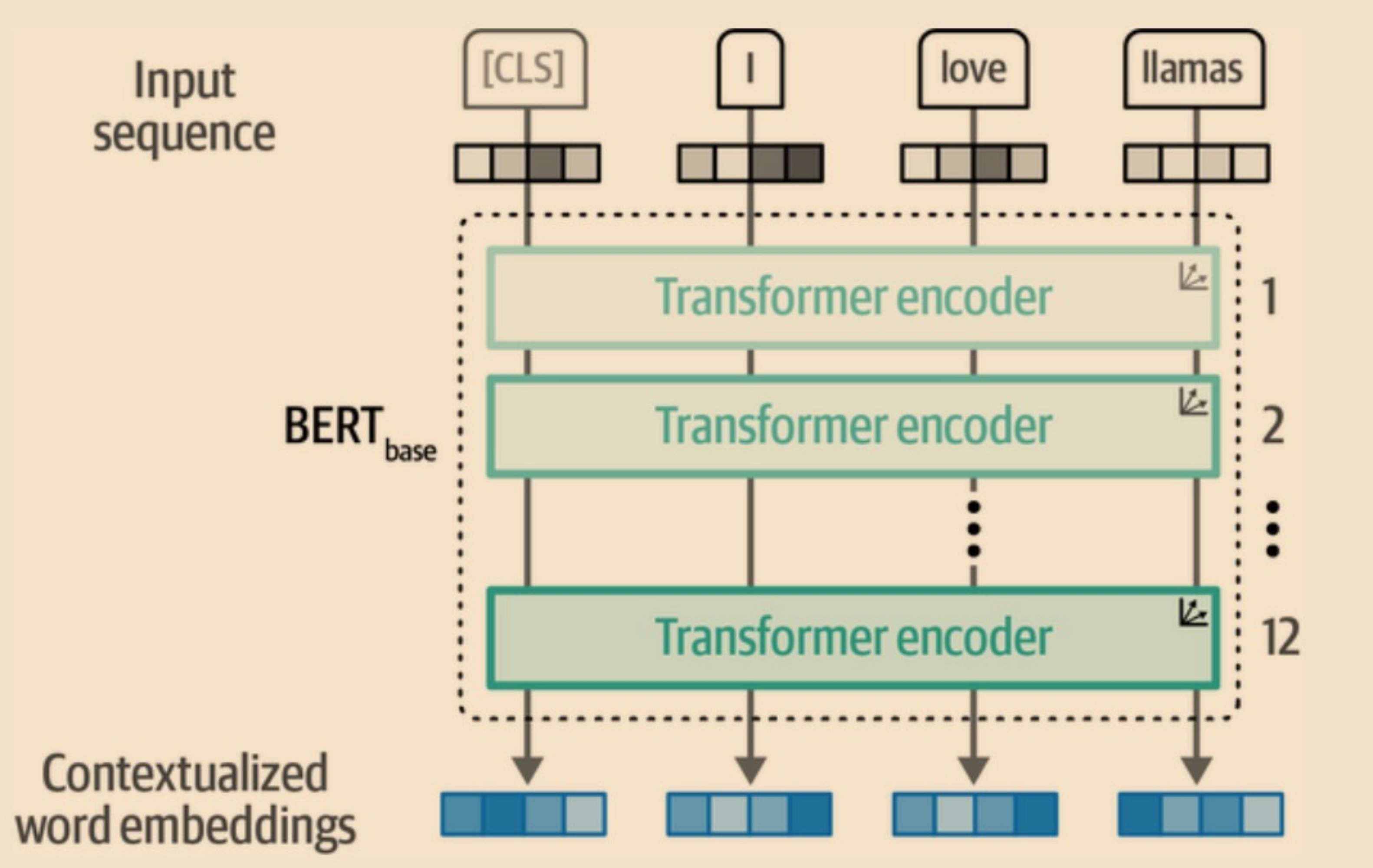
1.1 三种分类

Transformer 模型主要由两个核心部分构成：**编码器（Encoder）和解码器（Decoder）**。依托这两个关键组件的不同组合和应用，Transformer 模型发展出三种主流架构：**编码（Encoder-Only）大语言模型**、**解码（Decoder-Only）大语言模型**以及**编解码（Encoder-Decoder）大语言模型**。

模型	架构	参数量	机构
DeBERTa XXLarge [38]	Enc	1.5B	Microsoft
ALBERT [83]	Enc	Base = 12M, Large = 18M, XLarge = 60M	Google
RoBERTa [107]	Enc	356M	Meta/华盛顿大学
通义千问 [7]	Dec	1.8B、7B、14B、72B	阿里巴巴
LLaMa [169]	Dec	7B 65B	MetaAI
GPT-3 [13]	Dec	175B	OpenAI
Switch Transformers [47]	Enc-Dec	1.6T	Google
T5 [141]	Enc-Dec	11B	Google
GLM [43]	Enc-Dec	130B	清华大学

■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型：回顾BERT



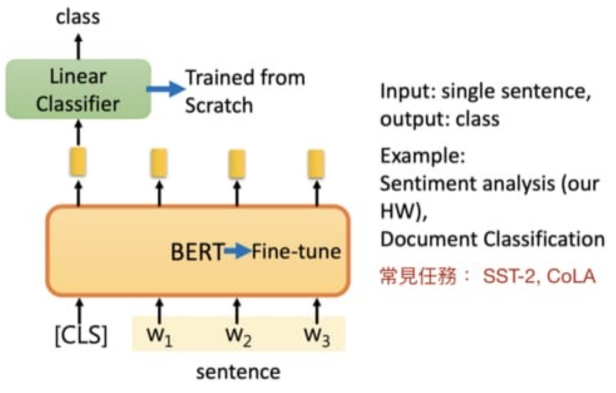
金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering
Key Laboratory of Sichuan Province

■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型

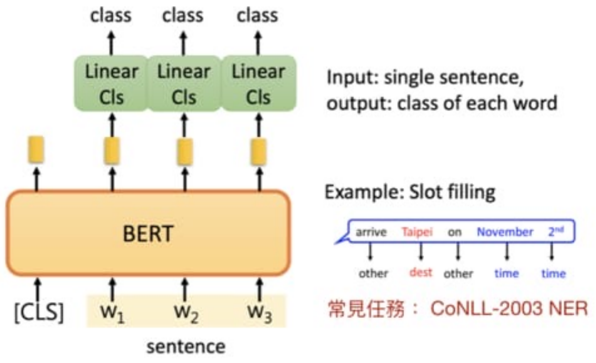
单一句子分类

bertForSequenceClassification



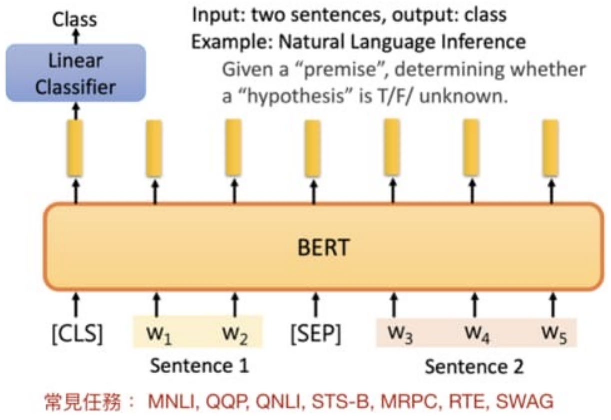
Token分类

bertForTokenClassification



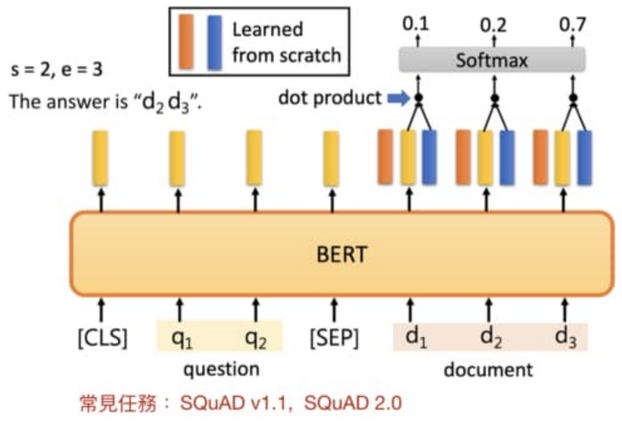
成对句子分类任务

bertForSequenceClassification



问答任务

bertForQuestionAnswering



主要特点：

- 使用编码器部分提取文本特征和语义信息
- 使用掩码技术进行训练

Encoder-Only的模型也被称为表示模型（**representation model**）。

■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

ALBERT模型是谷歌基于**BERT模型**开发的一种更高效的预训练语言表示模型，主要目的是**减少模型的参数量**和**训练成本**。

https://arxiv.org/pdf/1909.11942

Published as a conference paper at ICLR 2020

ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

Zhenzhong Lan[1] **Mingda Chen**[2*] **Sebastian Goodman**[1] **Kevin Gimpel**[2]
Piyush Sharma[1] **Radu Soricut**[1]
[1]Google Research [2]Toyota Technological Institute at Chicago
{lanzhzh, seabass, piyushsharma, rsoricut}@google.com
{mchen, kgimpel}@ttic.edu

ABSTRACT

Increasing model size when pretraining natural language representations often results in improved performance on downstream tasks. However, at some point further model increases become harder due to GPU/TPU memory limitations and longer training times. To address these problems, we present two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT (Devlin et al., 2019). Comprehensive empirical evidence shows that our proposed methods lead to models that scale much better compared to the original BERT. We also use a self-supervised loss that focuses on modeling inter-sentence coherence, and show it consistently helps downstream tasks with multi-sentence inputs. As a result, our best model establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks while having fewer parameters compared to BERT-large. The code and the pretrained models are available at https://github.com/google-research/ALBERT.

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

嵌入参数因子分解，通过将大的词汇嵌入矩阵分解为两个较小的矩阵，解决了隐藏层大小增加导致词汇嵌入参数量过大的问题

跨层参数共享，通过在不同层之间共享参数，有效控制了随网络深度增加而增长的参数量

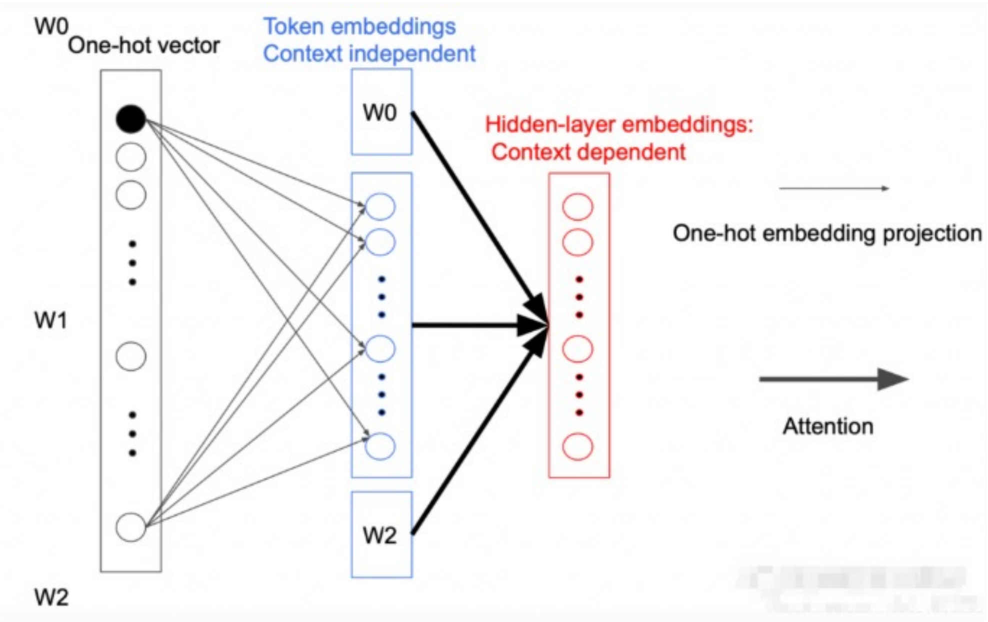
ALBERT在参数量上比同类模型减少了18倍，训练速度提升了约1.7倍，同时保持了高性能

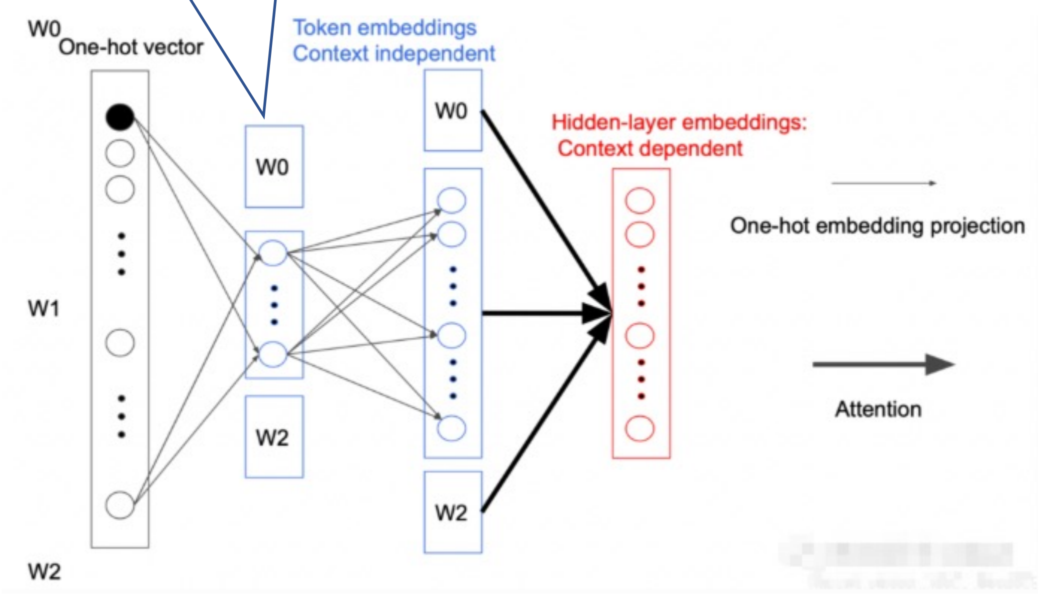
■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（1）嵌入参数因子分解

加入一个大小为E的隐藏层





Bert模型中Token Embedding的参数矩阵大小为$V \times H$

参数矩阵大小由$O(V \times H)$降为$O(V \times E + E \times H)$，当E远远小于H的时候，模型所需的参数将大大减少

■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（1）嵌入参数因子分解：代码

```
# BERT的词嵌入层
class BertEmbeddings(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.word_embeddings = nn.Embedding(
            vocab_size=config.vocab_size,  # 比如30000
            embedding_dim=config.hidden_size  # 比如768
        )
        # 参数量 = vocab_size * hidden_size
        # 例如：30000 * 768 = 23,040,000 参数
```

■1. 基于 Transformer 的模型架构

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（1）嵌入参数因子分解：代码

```python
class AlbertEmbeddings(nn.Module):
    def __init__(self, config):
        super().__init__()
        # 第一步：词映射到低维空间
        self.word_embeddings = nn.Embedding(
            vocab_size=config.vocab_size,  # 比如30000
            embedding_dim=config.embedding_size  # 比如128，这是低维的
        )
        # 第二步：低维映射到高维
        self.word_embedding_projection = nn.Linear(
            config.embedding_size,  # 输入维度128
            config.hidden_size  # 输出维度768
        )
```

In mathematics, factorization (...) or factoring consists of writing a number or another mathematical object as a product of several factors, usually smaller or simpler objects of the same kind. For example, 3 × 5 is a factorization of the integer 15.

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（1）嵌入参数因子分解：例子

中文BERT的词汇表大小大约为**2万**，参数量为

$$20000\times768 = 1536万 = \mathbf{15M},$$

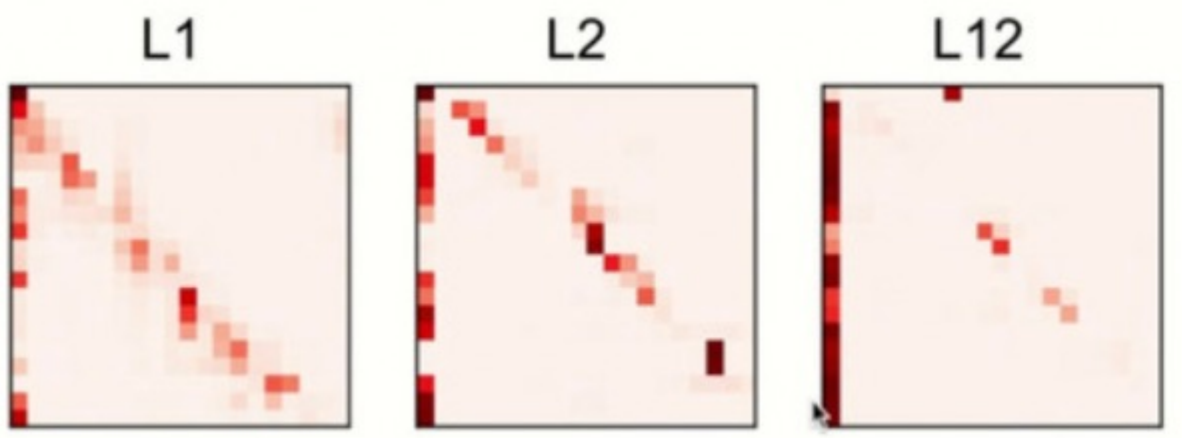
用参数**E=128**拆解矩阵，参数量为

$$20000\times128 + 128\times768=265.8304万 = \mathbf{2M+}$$

大概降低了**12M**参数矩阵大小.

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（2）跨层参数共享



Bert中有12个自注意力层，但是每一层的参数都基本相似。因此可以将一个自注意力层复制12次，只对一层自注意力层进行训练。

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

（2）跨层参数共享

共享可以分为**全连接层**、**注意力层**的参数共享；All-shared参数量降低最大，但会对最后的效果产生一定的影响；注意力层的参数对效果的减弱影响小一点。

	Model	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

(3) SOP任务（SENTENCE ORDER PREDICTION ）

ALBERT放弃了NSP任务，改用了SOP任务作为预训练任务。NSP任务的缺点：它无法区分topic prediction和**coherence** prediction，而SOP则重点关注**coherence**（inter-sentence loss）。

SOP任务

正例和NSP任务一致（判断两句话是否有顺序关系），反例则是两句的相反次序。

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

■1. 基于 Transformer 的模型架构

金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering
Key Laboratory of Sichuan Province

1.2 编码（Encoder-Only）大语言模型：ALBERT 模型

(4）ALBERT模型VS BERT模型

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

以BERT-large的基准，Bert-base的计算速度为4.7倍，ALBERT-base为5.6倍。
Albert的计算速度对比Bert其实并没有多大的提升，但由于减少了参数的量，还会对模型的性能产生一定的影响。

■1. 基于 Transformer 的模型架构

1.3 编码（Encoder-Only）大语言模型：RoBERTa 模型

RoBERTa: A Robustly Optimized BERT Pretraining Approach

Yinhan Liu[*§] Myle Ott[*§] Naman Goyal[*§] Jingfei Du[*§] Mandar Joshi[†]
Danqi Chen[§] Omer Levy[§] Mike Lewis[§] Luke Zettlemoyer[†§] Veselin Stoyanov[§]

[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,lsz}@cs.washington.edu

[§] Facebook AI
{yinhanliu,myleott,naman,jingfeidu,
danqi,omerlevy,mikelewis,lsz,ves}@fb.com

Abstract

Language model pretraining has led to significant performance gains but careful comparison between different approaches is challenging. Training is computationally expensive, often done on private datasets of different sizes, and, as we will show, hyperparameter choices have significant impact on the final results. We present a replication study of BERT pretraining (Devlin et al., 2019) that carefully measures the impact of many key hyperparameters and training data size. We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. Our best model achieves state-of-the-art results on GLUE, RACE and SQuAD. These results highlight the importance of previously overlooked design choices, and raise questions about the source of recently reported improvements. We release our

We present a replication study of BERT pretraining (Devlin et al., 2019), which includes a careful evaluation of the effects of hyperparmeter tuning and training set size. We find that BERT was significantly undertrained and propose an improved recipe for training BERT models, which we call RoBERTa, that can match or exceed the performance of all of the post-BERT methods. Our modifications are simple, they include: (1) training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern applied to the training data. We also collect a large new dataset (CC-NEWS) of comparable size to other privately used datasets, to better control for training set size effects.

When controlling for training data, our improved training procedure improves upon the pub-

Make BERT Great Again!

We present a replication study of BERT pretraining that carefully measures the impact of many key hyperparameters and training data size.

https://arxiv.org/abs/1907.11692

■1. 基于 Transformer 的模型架构

1.3 编码（Encoder-Only）大语言模型：RoBERTa 模型

RoBERTa

- 移除NSP任务
  - NSP任务对于提升模型性能并没有显著帮助
- 引入动态掩码机制
  - 将原始的训练数据复制多份，然后进行Masking，每轮的训练数据中Mask的位置也相应地发生变化
- 增加批量大小
- 延长训练时间
- 字节对编码
  - 利用bytes来替代subword中的unicode字符，可以用一个不太大的（50Kunit）的字典来对任意的输入进行编码而不用担心**未登录词**的出现。

■1. 基于 Transformer 的模型架构

金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering
Key Laboratory of Sichuan Province

1.3 编码（Encoder-Only）大语言模型：RoBERTa 模型

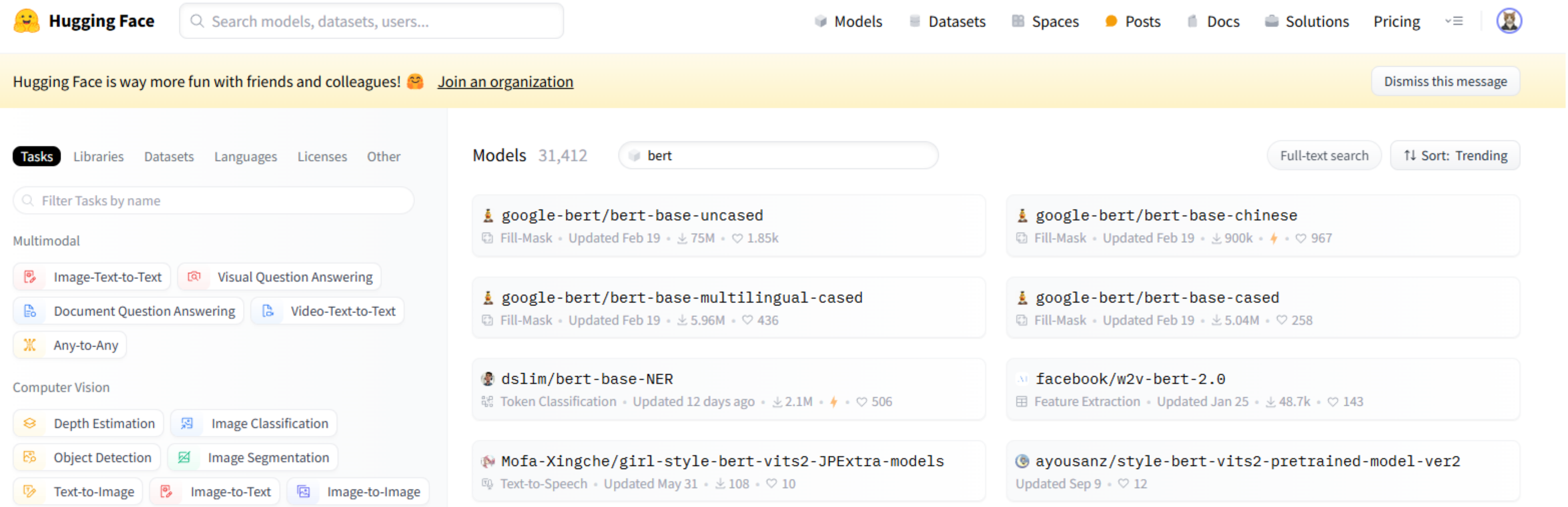
RoBERTa模型VS BERT模型

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

■1. 基于 Transformer 的模型架构

1.3 编码（Encoder-Only）大语言模型：体验

模型在线体验



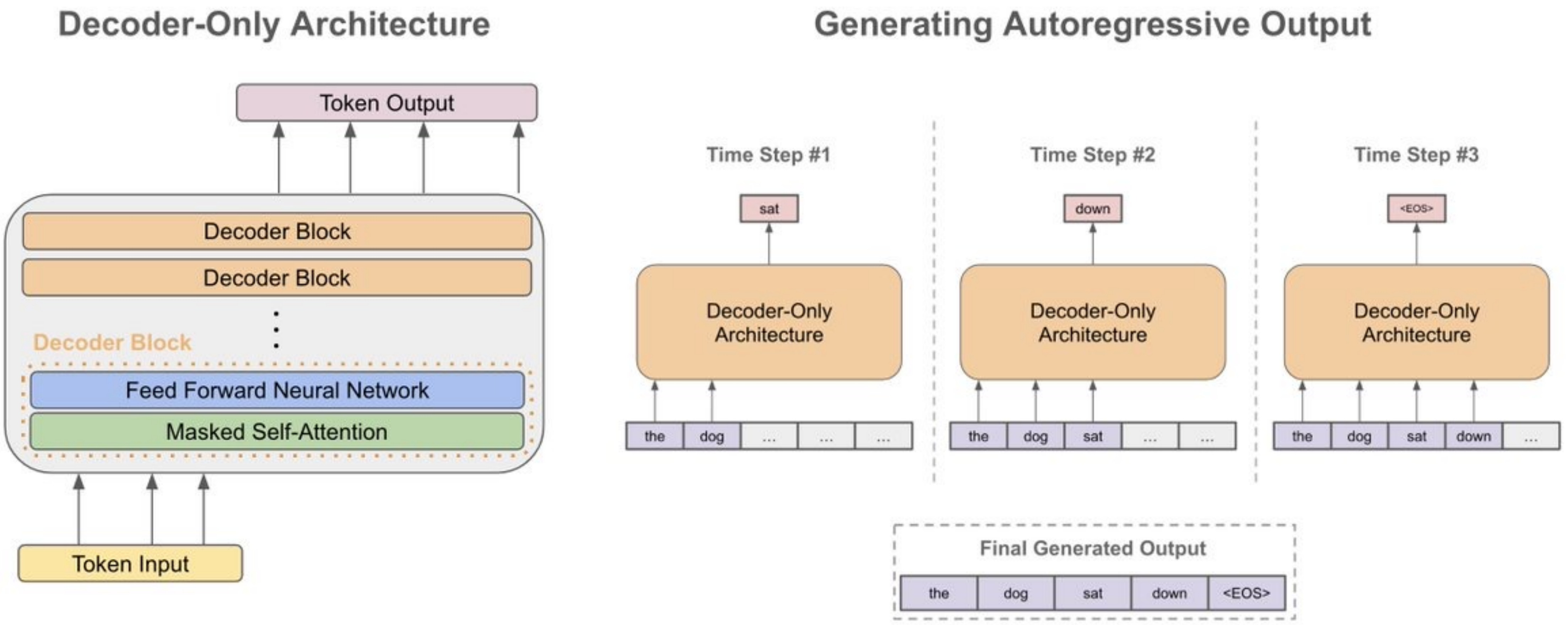
https://huggingface.co/models?sort=trending

Spaces - Hugging Face

■1. 基于 Transformer 的模型架构

1.4 解码（Decoder-Only）大语言模型

Decoder-Only的模型也被称为生成模型（**generative model**）。

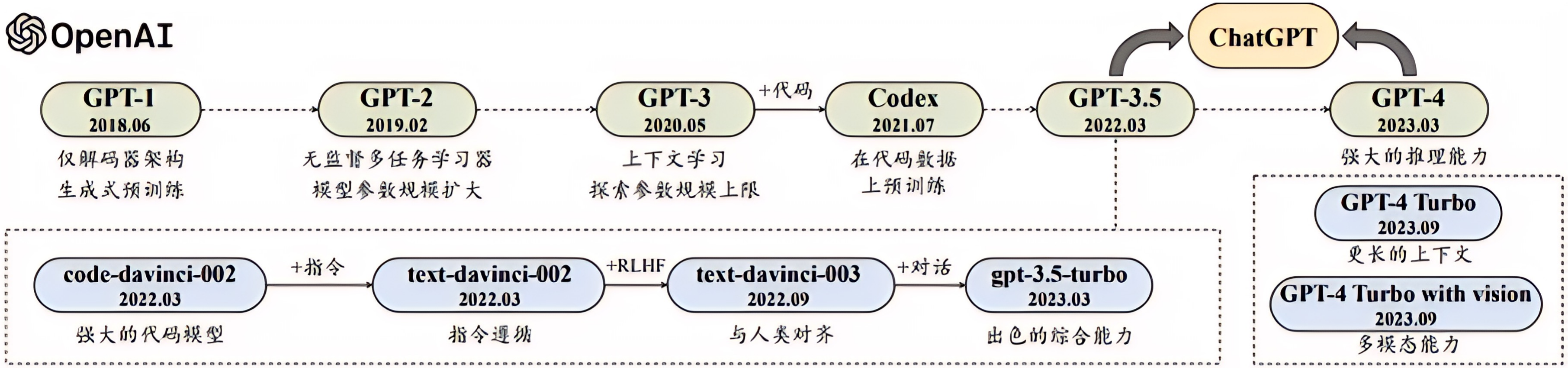


解码器处理输入的上下文信息，采用自回归生成方式逐步生成输出序列

■1. 基于 Transformer 的模型架构

金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering
Key Laboratory of Sichuan Province

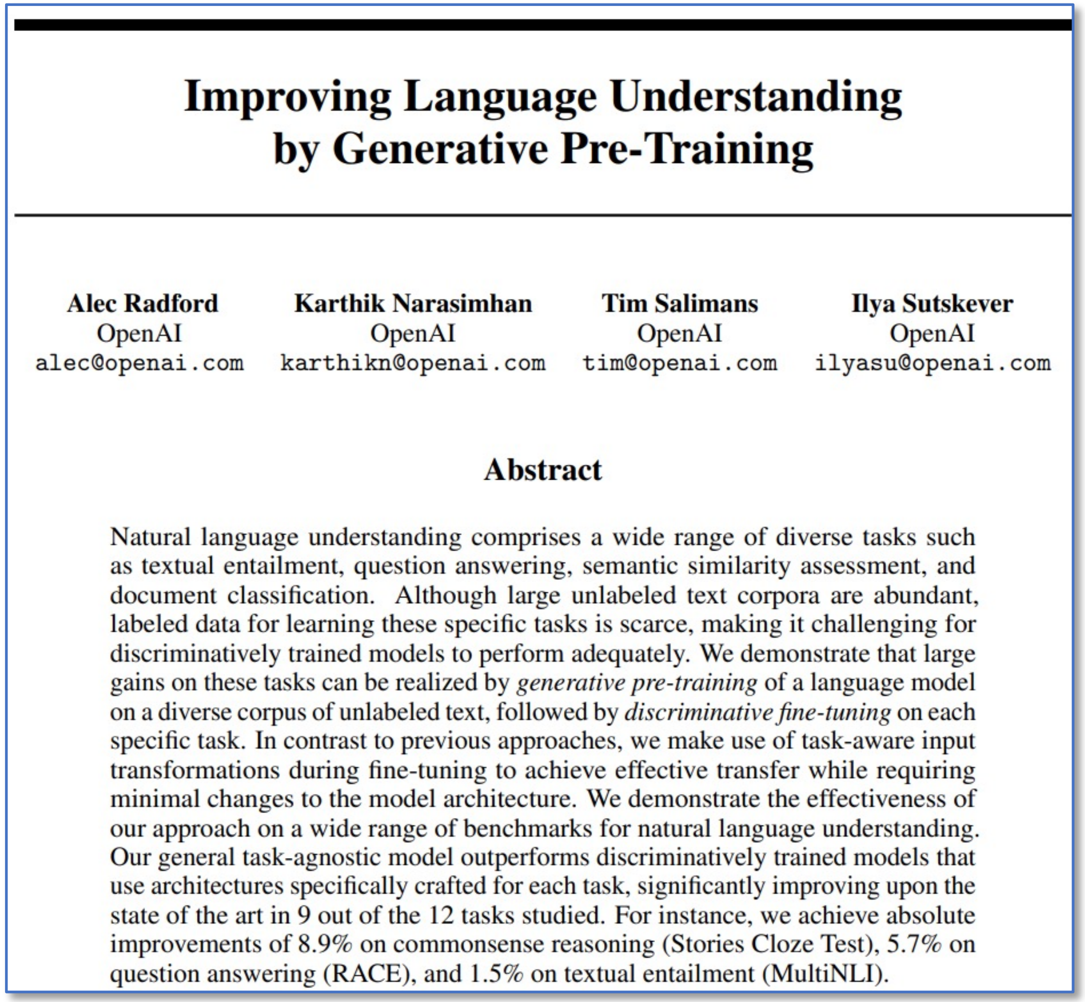
1.4 解码（Decoder-Only）大语言模型：GPT家族 比对



特性	GPT1	GPT2	GPT3
模型参数	1.17亿参数	15亿参数	1750亿参数
创新点	预训练+微调，Task-specific input transformations	预训练+Prompt+Predict，Zero-shot	预训练+Prompt+Predict，In-context learning
评价	未明确	Zero-shot新颖度拉满，但模型性能拉胯	开创性提出in-context learning概念，是Prompting祖师爷
预训练任务	语言模型预训练	语言模型预训练	语言模型预训练
微调	需要微调	尝试Zero-shot，未微调	尝试Few-shot，未微调
架构	Transformer Decoder	Transformer Decoder，改进了一些架构细节	96层的多头Transformer，Sparse Transformer架构
输入序列最大长度	512	1024	2048
数据集	7000本书籍	40GB文本	从45TB数据清洗得到570GB文本
多任务学习	不明显	明显，尝试Zero-shot	明显，通过in-context learning实现
迁移学习	有监督微调	Zero-shot	Few-shot learning（预测时给几个例子，不微调网络）

■1. 基于 Transformer 的模型架构

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-1



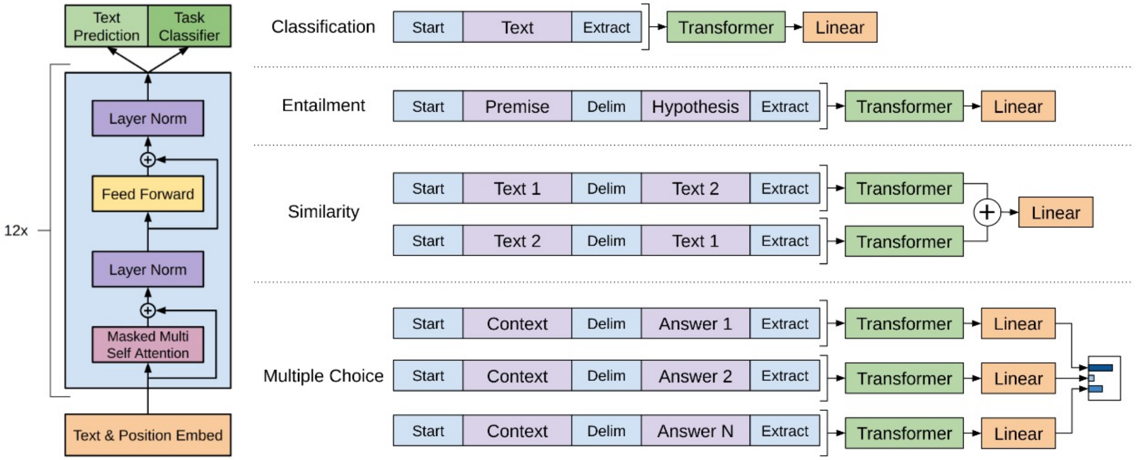
Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com
Karthik Narasimhan OpenAI karthikn@openai.com
Tim Salimans OpenAI tim@openai.com
Ilya Sutskever OpenAI ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

GPT-1的架构由12层Transformer组成，每层都使用了自注意力和前馈神经网络。GPT-1的关键特征是：**生成式预训练（无监督）+判别式任务精调（有监督）**。GPT-1在文本生成和理解任务上表现出了很好的性能，成为了当时最先进的自然语言处理模型之一。



■1. 基于 Transformer 的模型架构

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (dataset, objective)

1.核心思想：当模型的规模足够大且训练数据量足够丰富时，模型可以直接通过语言模型学习完成其他有监督学习任务，而无需在下游任务中进行微调。

2.单向Transformer模式：GPT-2继续沿用了GPT-1的单向Transformer解码器模式，但通过增加更多的网络参数和使用更大的数据集来提升模型性能。

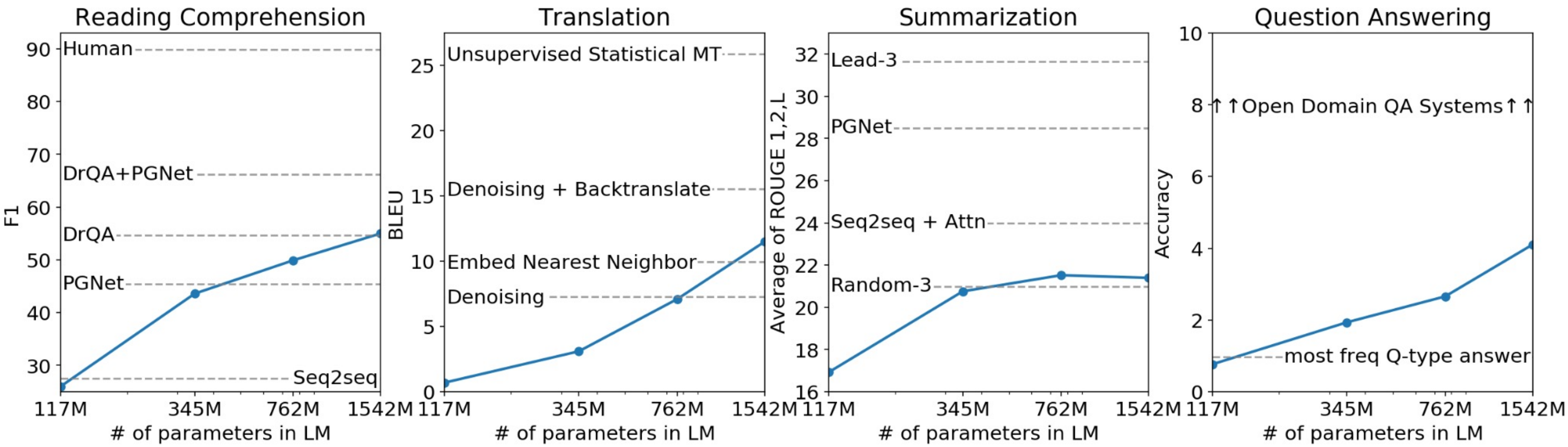
3.零样本学习（Zero-shot）：GPT-2提出了零样本学习的概念，即预训练好的模型能够直接被应用于多种下游任务而无需针对特定任务进行训练或微调。

4.多任务学习框架：通过扩大参数规模和无监督预训练，GPT-2探索了一种新的多任务学习框架，旨在提高模型的通用性和灵活性，减少对特定任务微调的依赖。

https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

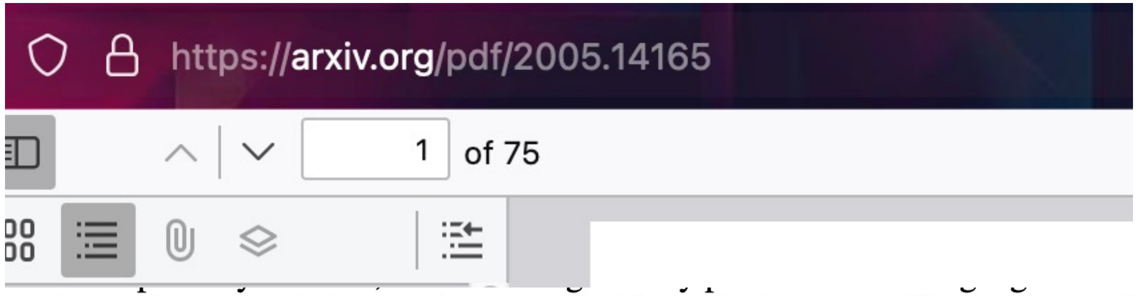
■1. 基于 Transformer 的模型架构

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-2



1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-3

GPT-3首次在大规模语言模型中展示了强大的**few-shot**学习能力，它推动了"提示工程"(prompt engineering)这一新兴领域的发展。



a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion

parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam
Girish Sastry Amanda Askell Sandhini Agarwal Ariel Herbert-Voss
Gretchen Krueger Tom Henighan Rewon Child Aditya Ramesh
Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Abstract

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks. We also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora.

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task agnostic model to perform a desired task.

https://arxiv.org/pdf/2005.14165

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-3

GPT-3

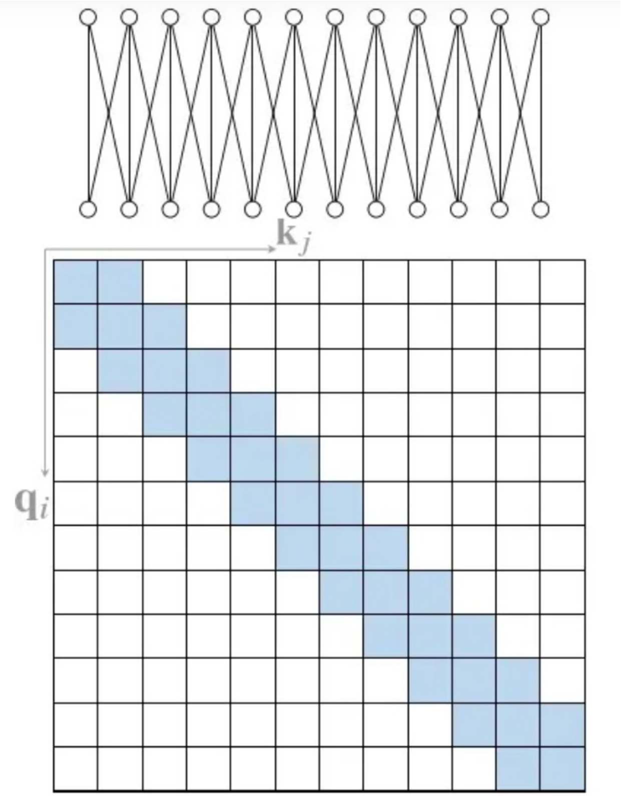
- 增加词向量长度 — 将词向量长度增至12,888
- 增加上下文窗口 — 上下文窗口大小扩展到2048个词元
- 引入新的注意力机制 — 采用交替稠密和局部带状稀疏的注意力机制
- 使用局部敏感哈希技术 — 将输入序列划分为多个桶，并在每个桶内计算注意力权重

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-3

局部带状稀疏注意力

在很多类型的数据中，比如文本、图像和时间序列等，一个元素（或令牌、像素等）与其邻近元素的关联性通常要比与远处元素的关联性更强。

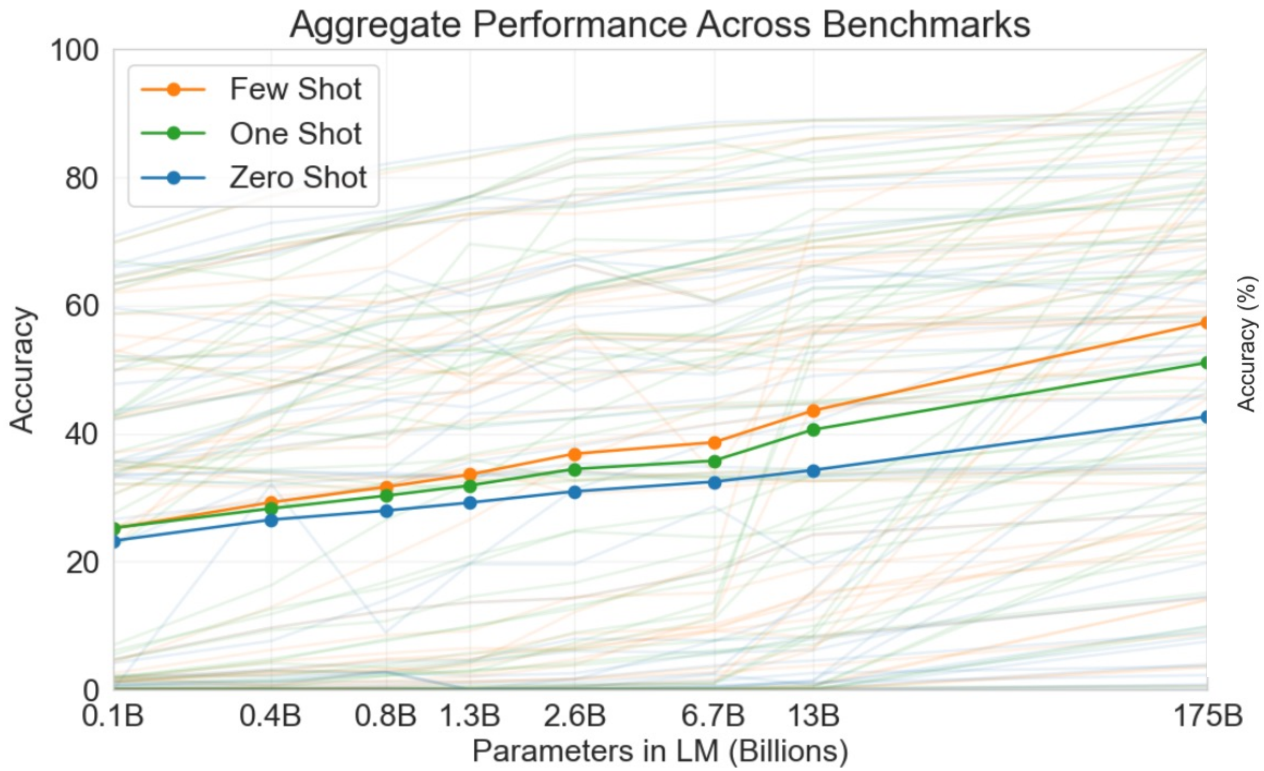
通过将注意力限制在一个局部区域内，带状注意力机制能够显著减少计算量，因为它只计算和存储那些可能高度相关的元素对之间的关系。

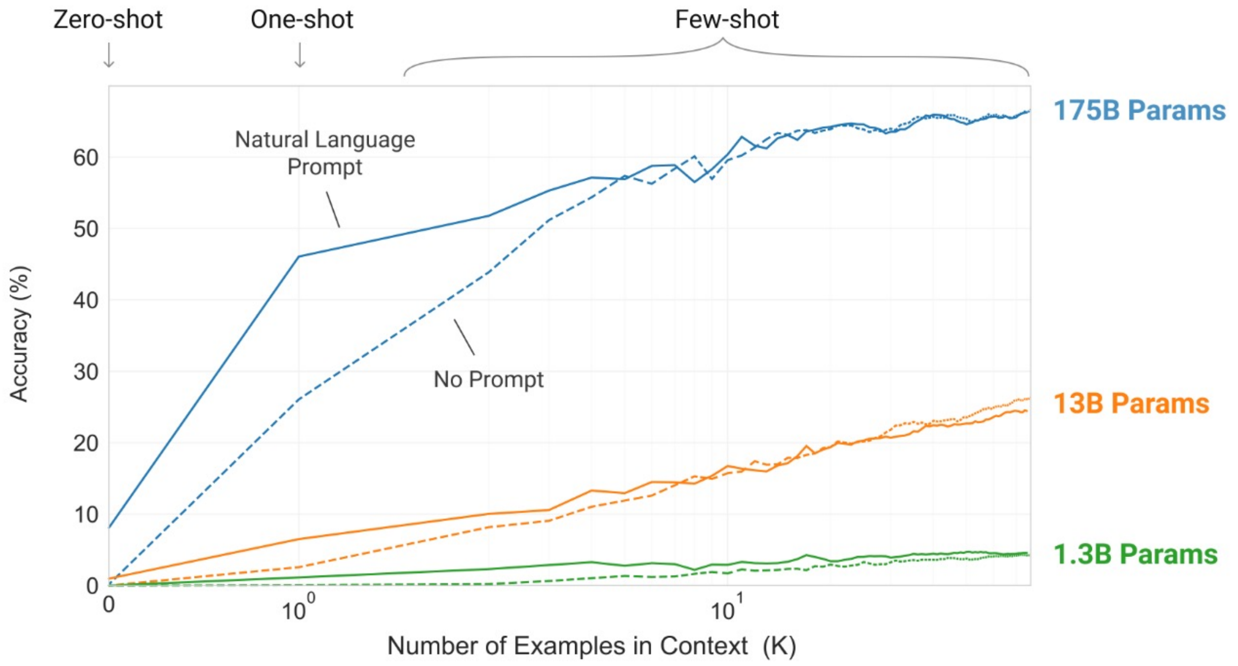


■1. 基于 Transformer 的模型架构

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-3

Few-Shot学习





1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-3.5

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang*
Pamela Mishkin* Chong Zhang Sandh
John Schulman Jacob Hilton Fraser
Amanda Askell† Peter We
Jan Leike*
OpenAI

are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are

GPT-3.5就是使用Instruct GPT训练的，其架构和GPT-3一致。它是ChatGPT的最初版本。

https://arxiv.org/abs/2203.02155

最早的深度RLHF：Deep Reinforcement Learning from Human Preferences，https://arxiv.org/pdf/1706.03741

1.4 解码（Decoder-Only）大语言模型：GPT家族 GPT-4

GPT-4模型

GPT-4 | OpenAI

GPT-4 can accept images as inputs and generate captions, classifications, and analyses.

Input

What can I make with these ingredients?

Output

There are many options for what you can make with these ingredients! Some possibilities include:

- Pancakes or waffles
- Crepes
- French toast
- Omelette or frittata
- Quiche
- Custard or pudding
- Cake or cupcakes
- Muffins or bread- Cookies or biscuits

These are just a few examples, but the possibilities are endless!

可以为我的宠物创建一张图片吗？希望以有趣且富有创意的方式表现。

当然可以！请告诉我你宠物的样子，比如它的种类、颜色、特点，还有你希望它出现在什么样的场景中，越详细越好，这样我可以为你生成一张更符合你期待的图片。

一只黑色的猫

你想让这只黑色的猫出现在什么样的场景或主题下呢？比如它是在玩耍、探险，还是坐在一个特别的背景前？你可以添加任何创意元素来个性化这张图片！

草地

这里是一只在草地上玩耍的黑色猫咪形象，希望你喜欢这个充满阳光和活力的场景！如果有任何其他调整或想法，随时告诉我！

给"ChatGPT"发送消息

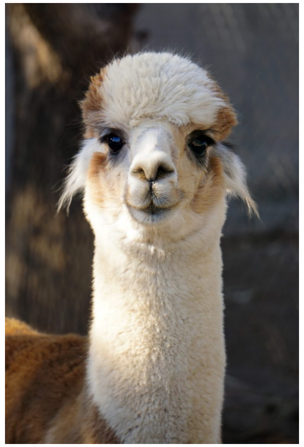
LLM Playground

■1. 基于 Transformer 的模型架构

金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering
Key Laboratory of Sichuan Province

1.5 解码（Decoder-Only）大语言模型： LLaMA模型

https://arxiv.org/abs/2302.13971



LLaMA（Large Language Model Meta AI） 是由 Meta AI 推出开放且高效的的大语言模型，它提供四种不同规模的版本： 7B、 13B、 33B 和 65B。

LLaMA 的显著特点在于其训练数据完全基于公开数据集，而且其模型结构和训练方法也均已公开。

llama

noun [C]

UK /ˈlɑː.mə/ US /ˈlɑː.mə/

Add to word list

a large South American animal with a long neck and long hair, often kept for its meat, milk, or fur and to carry heavy loads

美洲駝（產於南美）

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron*, Thibaut Lavril*, Gautier Izacard*, Xavier Martinet
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave*, Guillaume Lample*

Meta AI

Abstract

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community[1].

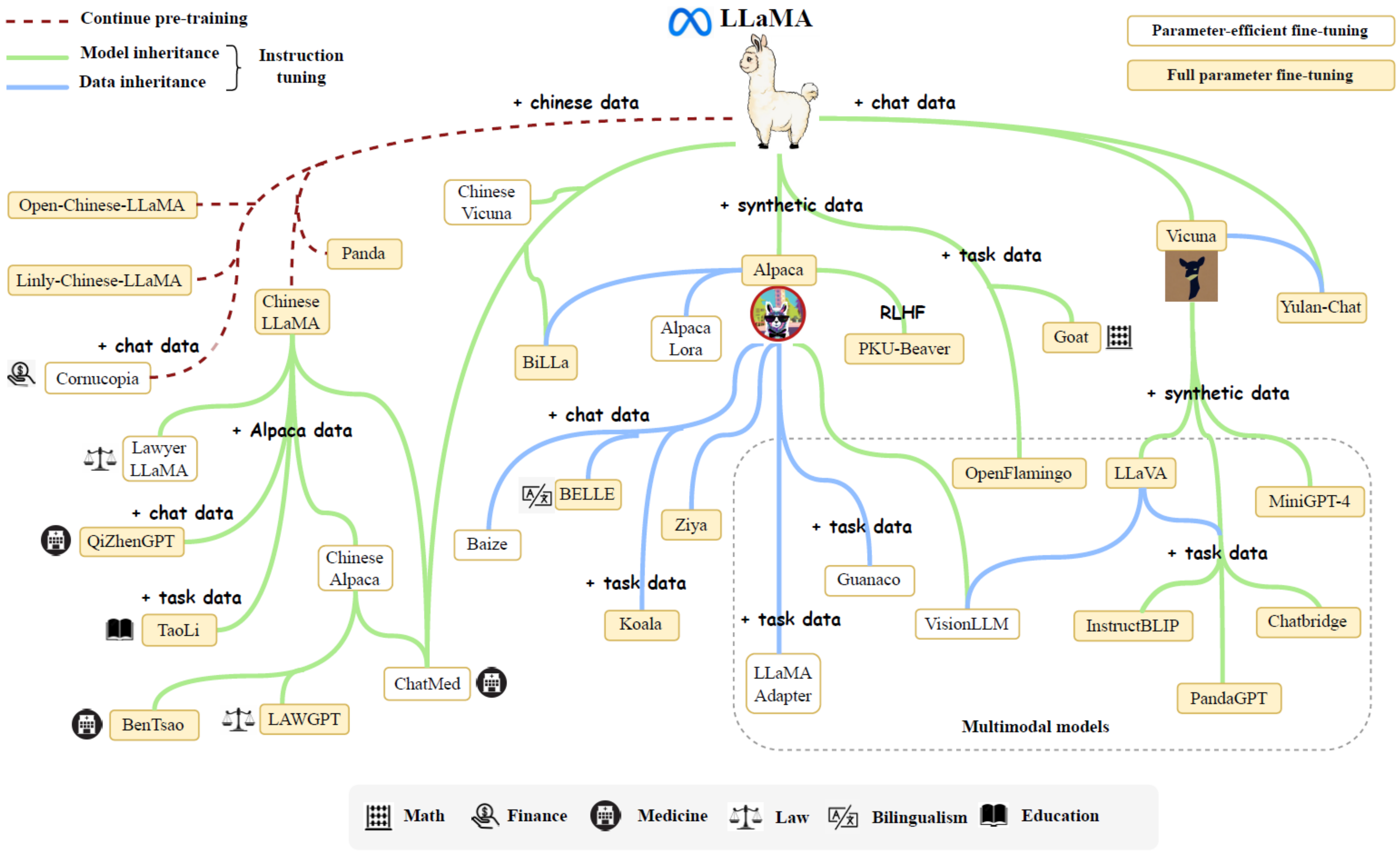
1 Introduction

Large Languages Models (LLMs) trained on massive corpora of texts have shown their ability to perform new tasks from textual instructions or from a few examples (Brown et al., 2020). These few-shot properties first appeared when scaling models to a sufficient size (Kaplan et al., 2020), resulting in a

performance, a smaller one trained longer will ultimately be cheaper at inference. For instance, although Hoffmann et al. (2022) recommends training a 10B model on 200B tokens, we find that the performance of a 7B model continues to improve even after 1T tokens.

The focus of this work is to train a series of language models that achieve the best possible performance at various inference budgets, by training on more tokens than what is typically used. The resulting models, called *LLaMA*, ranges from 7B to 65B parameters with competitive performance compared to the best existing LLMs. For instance, LLaMA-13B outperforms GPT-3 on most benchmarks, despite being 10× smaller. We believe that this model will help democratize the access and study of LLMs, since it can be run on a single GPU. At the higher-end of the scale, our 65B-parameter model is also competitive with the best large language models such as Chinchilla or PaLM-540B.

金融智能与金融工程四川省重点实验室
Financial Intelligence and Financial Engineering Key Laboratory of Sichuan Province



■1. 基于 Transformer 的模型架构

1.5 解码（Decoder-Only）大语言模型： LLaMA模型

LLaMA

完全基于公开数据集进行训练	增强了模型的透明度和可验证性
引入RMSNorm归一化处理	在每个Transformer子层的输入端进行归一化，增强训练的稳定性
采用ROPE策略	更有效地处理不同长度的文本序列
使用SwiGLU激活函数	进一步提升了模型的表达能力

[Llama-3.1-Nemotron-70B-Instruct-HF · Hugging Face]

[LLM Playground]

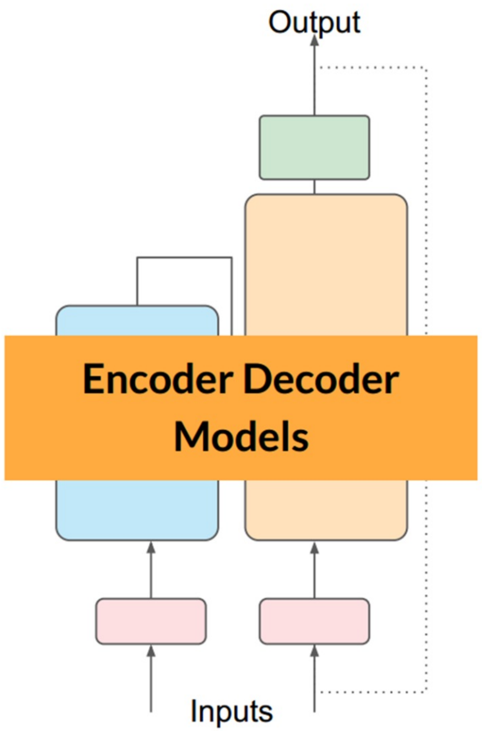
■1. 基于 Transformer 的模型架构

1.6 编解码（Encoder-Decoder）大语言模型

核心思想

通过编码器对输入序列进行编码，提取其特征和语义信息，然后将编码结果传递给解码器。解码器根据编码结果生成相应的输出序列。

优势：能够有效处理输入序列与输出序列之间的关系，从而提升机器翻译和对话生成等任务的准确性。



缺点是模型复杂度较高，训练时间和计算资源消耗较大

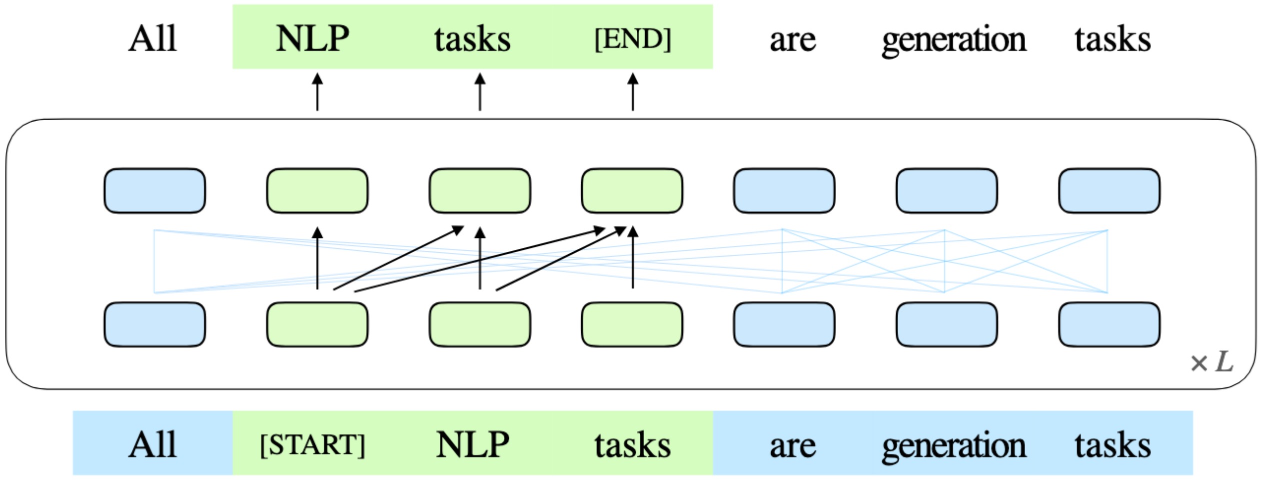
■1. 基于 Transformer 的模型架构

1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

GLM（General Language Model）模型是由清华大学研究团队开发的一款大语言模型，它基于自回归空白填充目标进行预训练。

GLM: General Language Model Pretraining with Autoregressive Blank Infilling

Zhengxiao Du[*1,2] **Yujie Qian**[*3] **Xiao Liu**[1,2] **Ming Ding**[1,2] **Jiezhong Qiu**[1,2]
Zhilin Yang[†1,4] **Jie Tang**[†1,2]
[1]Tsinghua University [2]Beijing Academy of Artificial Intelligence (BAAI)
[3]MIT CSAIL [4]Shanghai Qi Zhi Institute
zx-du20@mails.tsinghua.edu.cn yujieq@csail.mit.edu
{zhiliny,jietang}@tsinghua.edu.cn

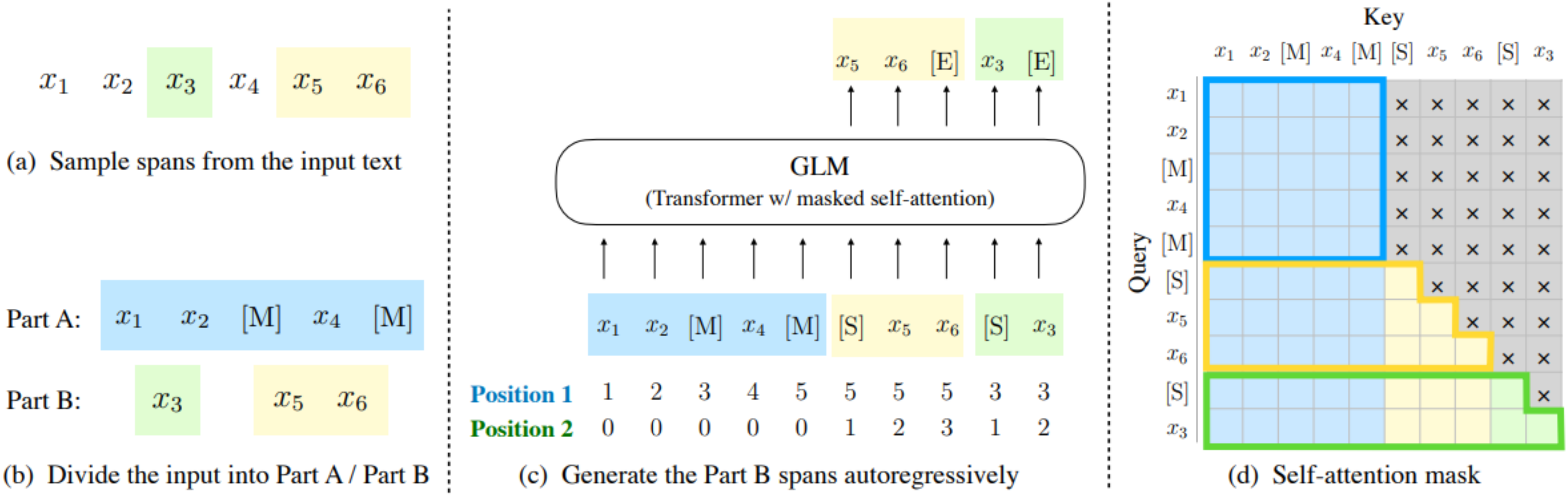


https://arxiv.org/abs/2103.10360

■1. 基于 Transformer 的模型架构

1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

1. 自回归空白填充预训练



1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

1. 自回归空白填充预训练

输入处理

- 对于给定的输入文本$\boldsymbol{x} = [x_1, \ldots, x_n]$，模型随机选择 m 个文本片段 $\{s_1, \ldots, s_m\}$，每个片段 s_i由一个连续的词元序列 $[s_{i,1}, \ldots, s_{i,l_i}]$表示。

文本损坏

- 每个选中的文本片段被单个 [MASK] 标记替换，生成损坏文本 x_{corrupt}。模型将输入 x 分为两部分：A部分包括损坏文本 x_{corrupt}，B部分包含被掩码的片段。A部分使用双向注意力机制，B部分则使用单向注意力机制。文本片段的首尾加上起始标志 [S] 和终止标志 [E]，以实现自回归生成。

文本重排

- 为了捕捉不同跨度之间的相互依赖关系，GLM 通过排列组合的方式对输入文本片段（B部分）进行重排，并将A部分和B部分组合成序列输入模型中，用于预测文本片段中缺失的标记序列。

1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

2. 预训练目标函数

$$\max_{\theta}\mathbb{E}_{z\sim Z_m}\left[\sum_{i=1}^{m}\log p_\theta\left(s_{z_i}|x_{\text{corrupt}},s_{z<i}\right)\right]$$

其中，θ 是模型参数，Z_m是m个文本片段的所有可能的排列组合，$p_\theta\left(s_{z_i}|x_{\text{corrupt}},s_{z<i}\right)$是在给定损坏文本 x_{corrupt}和之前预测的片段$s_{z<i}$的条件下，预测当前片段s_{z_i} 的条件概率。

在生成每个片段 s_i 的过程中，该片段的概率可分解为：

$$p_\theta\left(s_i|x_{\text{corrupt}},s_{z<i}\right)=\prod_{j=1}^{l_i}p\left(s_{i,j}|x_{\text{corrupt}},s_{z<i},s_{i,<j}\right)$$

其中，$s_{i,<j}$ 表示在预测当前标记 $s_{i,j}$之前，该片段已预测出的标记序列。

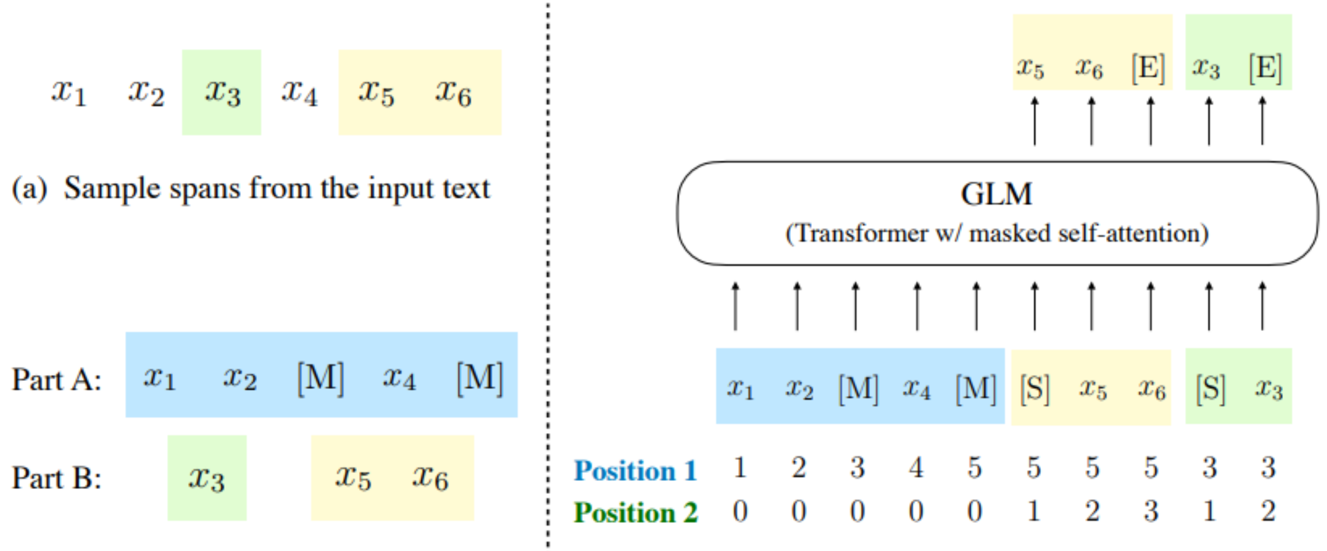
1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

3.二维位置编码

用二维位置编码处理输入x的A部分和B部分的相对位置关系。每个词元被赋予两个位置编码：

- 第一个位置编码表示词元在损坏文本 x_{corrupt} 中的全局位置
- 第二个位置编码表示词元在其所属文本片段（A部分或B部分）的局部位置。

这两个位置编码通过可学习的嵌入表示被投影到两个向量中，随后被添加到对应的输入嵌入中。



■1. 基于 Transformer 的模型架构

1.6 编解码（Encoder-Decoder）大语言模型： GLM模型

具体示例

假设输入文本为“成都是四川省的省会城市”，它由6个词元组成： $\boldsymbol{x} = [成都, 是, 四川省, 的, 省会, 城市]$，其中两个文本片段“四川省”和“省会城市”被采样。

将它们替换为特殊的掩码符号 [MASK] 得到的损坏文本为：

$$x_{\text{corrupt}} = [成都, 是, [MASK], 的, [MASK]]$$

GLM 模型将输入x分为两部分：

A部分（损坏文本）： 成都, 是, $[MASK]$, 的, $[MASK]$

B部分（被掩码的片段）： 四川省, 省会, 城市

B部分进行重排，假设重排后的顺序为“省会城市”和“四川省”，再将A部分和B部分组成序列输入GLM中，并加上起始标志 [S]，即：

$$成都, 是, [MASK], 的, [MASK], [S], 省会, 城市, [S], 四川省$$

其中，用二维位置编码进行位置编码

	成都	是	$[MASK]$	的	$[MASK]$	$[S]$	省会	城市	$[S]$	四川省
位置编码 1:	1	2	3	4	5	5	5	5	3	3
位置编码 2:	0	0	0	0	0	1	2	3	1	2

目录

1. 基于 Transformer 的模型架构

- 编码大语言模型
- 解码大语言模型
- 编解码大语言模型

2. 非 Transformer 的模型架构

- FAT模型
- AFT模型
- RWKV模型

3. 大模型架构配置

- 归一化技术
- 激活函数
- 位置编码
- 注意力与偏置

■2. 非 Transformer 的模型架构

2.1 FAT 模型

Transformer的弊端： 注意力机制时间复杂为$O(N^2)$

考虑输入和输出序列长度都是 n，维度是 d。

对于RNN结构，其隐藏层权重是$\mathbb{R}^{d\times d}$，n 个时间步的总复杂度是$O(nd^2)$；无法并行!

对于自注意力结构，Q、K和V都是$\mathbb{R}^{n\times d}$，总复杂度是$O(n^2d)$；能够并行!

FAT模型重新构造了原始Transformer 模型中的自注意力机制，引入了基于**核函数**的特征映射。

https://arxiv.org/abs/2006.16236

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

**Transformers are RNNs:
Fast Autoregressive Transformers with Linear Attention**

Angelos Katharopoulos [1 2] **Apoorv Vyas** [1 2] **Nikolaos Pappas** [3] **François Fleuret** [2 4 *]

Abstract

Transformers achieve remarkable performance in several tasks but due to their quadratic complexity, with respect to the input's length, they are prohibitively slow for very long sequences. To address this limitation, we express the self-attention as a linear dot-product of kernel feature maps and make use of the associativity property of matrix products to reduce the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, where N is the sequence length. We show that this formulation permits an iterative implementation that dramatically accelerates autoregressive transformers and reveals their relationship to recurrent neural networks. Our *linear transformers* achieve similar performance to vanilla transformers and they are up to 4000x faster on autoregressive prediction of very long sequences.

by the global receptive field of self-attention, which processes contexts of N inputs with a quadratic memory and time complexity $\mathcal{O}(N^2)$. As a result, in practice transformers are slow to train and their context is *limited*. This disrupts temporal coherence and hinders the capturing of long-term dependencies. Dai et al. (2019) addressed the latter by attending to memories from previous contexts albeit at the expense of computational efficiency.

Lately, researchers shifted their attention to approaches that increase the context length without sacrificing efficiency. Towards this end, Child et al. (2019) introduced sparse factorizations of the attention matrix to reduce the self-attention complexity to $\mathcal{O}(N\sqrt{N})$. Kitaev et al. (2020) further reduced the complexity to $\mathcal{O}(N\log N)$ using locality-sensitive hashing. This made scaling to long sequences possible. Even though the aforementioned models can be efficiently trained on large sequences, they do not speed-up autoregressive inference.

■2. 非 Transformer 的模型架构

2.1 FAT 模型：核方法

定义 7.6（核函数） 设 $\mathcal{X}$ 是输入空间（欧氏空间 $\mathbf{R}^n$ 的子集或离散集合），又设 $\mathcal{H}$ 为特征空间（希尔伯特空间），如果存在一个从 $\mathcal{X}$ 到 $\mathcal{H}$ 的映射

$$\phi(x): \mathcal{X} \to \mathcal{H} \tag{7.65}$$

使得对所有 $x, z \in \mathcal{X}$，函数 $K(x,z)$ 满足条件

$$K(x,z) = \phi(x) \cdot \phi(z) \tag{7.66}$$

则称 $K(x,z)$ 为核函数，$\phi(x)$ 为映射函数，式中 $\phi(x) \cdot \phi(z)$ 为 $\phi(x)$ 和 $\phi(z)$ 的内积.

2. 非 Transformer 的模型架构

2.1 FAT 模型

只需要定义一个非负的sim（用于定义Attention函数）。这包括所有的核函数: $\mathbb{R}^{2\times d} \rightarrow \mathbb{R}_+$

传统 Transformer 中的自注意力

$$V' = \operatorname{softmax}\left(\frac{QK^\top}{\sqrt{D}}\right)V.$$

两层嵌套的 for 循环：j和i
计算复杂度为 O(N²)

用下标 i 来表示矩阵的第 i 行

$$V_i' = \frac{\sum_{j=1}^{N} \operatorname{sim}(Q_i, K_j)V_j}{\sum_{j=1}^{N} \operatorname{sim}(Q_i, K_j)}.$$

$$\operatorname{sim}(Q_i, K_j) = \exp\left(\frac{Q_i K_j^\top}{\sqrt{D}}\right)$$

采用核函数

FAT注意力机制

$$\operatorname{sim}(Q_i, K_j) = \phi(Q_i)\phi(K_j)^\top$$

$\phi(x) = \operatorname{ELU}(x) + 1$ exponential linear unit

$$V_i' = \frac{\sum_{j=1}^{N} \phi(Q_i)\phi(K_j)^\top V_j}{\sum_{j=1}^{N} \phi(Q_i)\phi(K_j)^\top}$$

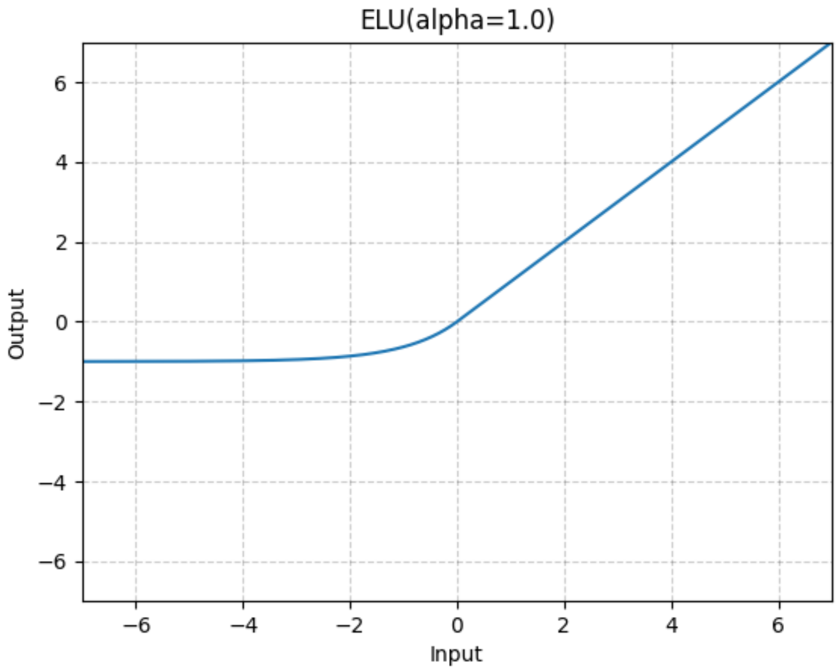
求和项与 i 无关

$$V_i' = \frac{\phi(Q_i)\sum_{j=1}^{N} \phi(K_j)^\top V_j}{\phi(Q_i)\sum_{j=1}^{N} \phi(K_j)^\top}$$

■2. 非 Transformer 的模型架构

2.1 FAT 模型：ELU

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha * (\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$



■2. 非 Transformer 的模型架构

2.1 FAT 模型

在此基础上，解码器只需将式 (6.4) 中的 N 替换为当前词元（第 i 个），公式如下：

$$\boldsymbol{V}_i' = \frac{\sum_{j=1}^{i} \operatorname{sim}(\boldsymbol{Q}_i, \boldsymbol{K}_j)\boldsymbol{V}_j}{\sum_{j=1}^{i} \operatorname{sim}(\boldsymbol{Q}_i, \boldsymbol{K}_j)} \tag{6.10}$$

可以将前面定义的 sim() 函数带入，得到公式如下：

$$\boldsymbol{V}_i' = \frac{\phi(\boldsymbol{Q}_i)\sum_{j=1}^{i} \phi(\boldsymbol{K}_j)^\top \boldsymbol{V}_j}{\phi(\boldsymbol{Q}_i)\sum_{j=1}^{i} \phi(\boldsymbol{K}_j)^\top} \tag{6.11}$$

为了简化公式表达，引入两个新的符号 $\boldsymbol{S}_i$ 和 $\boldsymbol{Z}_i$，它们表达式分别如下：

$$\boldsymbol{S}_i = \sum_{j=1}^{i} \phi(\boldsymbol{K}_j)^\top \boldsymbol{V}_j \tag{6.12}$$

$$\boldsymbol{Z}_i = \sum_{j=1}^{i} \phi(\boldsymbol{K}_j)^\top \tag{6.13}$$

稍作变换后，$\boldsymbol{S}_i$ 和 $\boldsymbol{Z}_i$ 可以写成递归形式，如下：

$$\boldsymbol{S}_i = \phi(\boldsymbol{K}_i)^\top \boldsymbol{V}_i + \boxed{\boldsymbol{S}_{i-1}} \qquad 4)$$

$$\boldsymbol{Z}_i = \phi(\boldsymbol{K}_i)^\top + \boxed{\boldsymbol{Z}_{i-1}} \qquad 5)$$

缓存下来，
降低复杂度

■2. 非 Transformer 的模型架构

2.2 AFT 模型 计算复杂度为$O(N)$

传统 Transformer 中的自注意力

$$V_i' = \frac{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j) V_j}{\sum_{j=1}^{N} \mathrm{sim}(Q_i, K_j)} .$$

FAT注意力机制

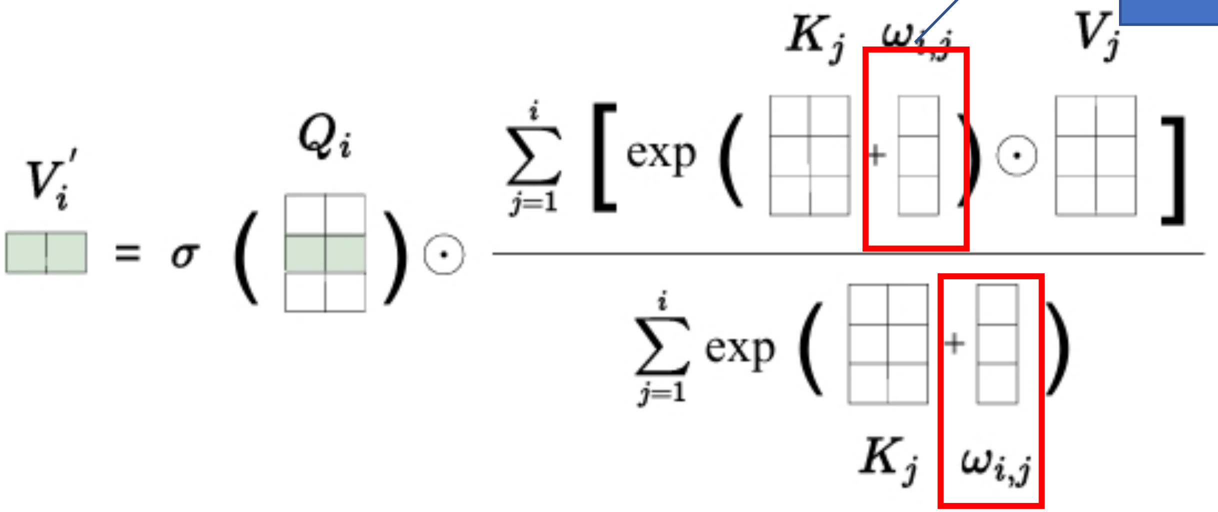
$$V_i' = \frac{\phi(Q_i) \sum_{j=1}^{N} \phi(K_j)^\top V_j}{\phi(Q_i) \sum_{j=1}^{N} \phi(K_j)^\top}$$

AFT的自注意力

$$V_i' = \sigma(Q_i) \odot \frac{\sum_{j=1}^{i} \exp(K_j + \omega_{i,j}) \odot V_j}{\sum_{j=1}^{i} \exp(K_j + \omega_{i,j})}$$

$\omega_{i,j}$是待训练的参数

可训练的位置偏差



https://arxiv.org/abs/2105.14103 An Attention Free Transformer

■2. 非 Transformer 的模型架构

2.3 RWKV 模型

2023 年，彭博[1]提出了一种新的模型架构Receptance Weighted Key Value (RWKV) 。 RWKV模型从 FAT 模型中借鉴了简化自注意力机制的思想,结合了 RNN 和 Transformer 的最佳特性。

https://arxiv.org/abs/2305.13048

RWKV: Reinventing RNNs for the Transformer Era

Bo Peng[1,2*] Eric Alcaide[2,3,4*] Quentin Anthony[2,5*]
Alon Albalak[2,6] Samuel Arcadinho[2,7] Stella Biderman[2,8] Huanqi Cao[9] Xin Cheng[10]
Michael Chung[11] Xingjian Du[1] Matteo Grella[12] Kranthi Kiran GV[2,13] Xuzheng He[2]
Haowen Hou[14] Jiaju Lin[1] Przemysław Kazienko[15] Jan Kocoń[15] Jiaming Kong[16]
Bartłomiej Koptyra[15] Hayden Lau[2] Krishna Sri Ipsit Mantri[17] Ferdinand Mom[18,19]
Atsushi Saito[2,20] Guangyu Song[21] Xiangru Tang[22] Bolun Wang[23] Johan S. Wind[24]
Stanisław Woźniak[15] Ruichong Zhang[9] Zhenyuan Zhang[2] Qihang Zhao[25,26]
Peng Zhou[23] Qinghua Zhou[5] Jian Zhu[27] Rui-Jie Zhu[28,29]

[1]Generative AI Commons [2]EleutherAI [3]U. of Barcelona [4]Charm Therapeutics [5]Ohio State U. [6]U. of C., Santa Barbara
[7]Zendesk [8]Booz Allen Hamilton [9]Tsinghua University [10]Peking University [11]Storyteller.io [12]Crisis24 [13]New York U.
[14]National U. of Singapore [15]Wroclaw U. of Science and Technology [16]Databaker Technology [17]Purdue U. [18]Criteo AI Lab
[19]Epita [20]Nextremer [21]Moves [22]Yale U. [23]RuoxinTech [24]U. of Oslo [25]U. of Science and Technology of China
[26]Kuaishou Technology [27]U. of British Columbia [28]U. of C., Santa Cruz [29]U. of Electronic Science and Technology of China

Abstract

Transformers have revolutionized almost all natural language processing (NLP) tasks but suffer from memory and computational complexity that scales quadratically with sequence length. In contrast, recurrent neural networks (RNNs) exhibit linear scaling in memory and computational requirements but struggle to match the same performance as Transformers due to limitations in parallelization and scalability. We propose a novel model architecture, Receptance Weighted Key Value (RWKV), that combines the efficient parallelizable training of transformers with the efficient inference of RNNs.

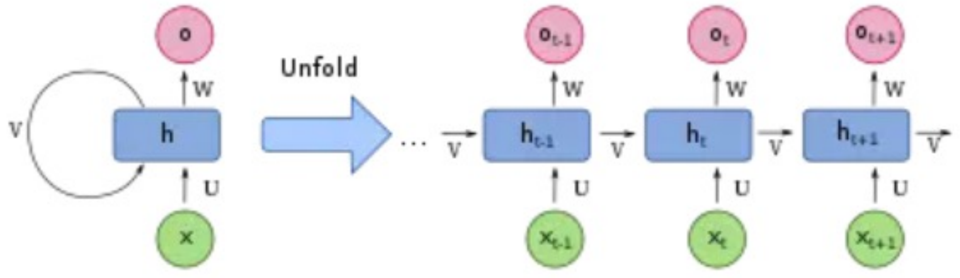
processing tasks such as natural language understanding, conversational AI, time-series analysis, and indirectly sequential formats like images and graphs (Brown et al., 2020; Ismail Fawaz et al., 2019; Wu et al., 2020; Albalak et al., 2022). Predominant among these techniques include RNNs and Transformers (Vaswani et al., 2017), each with specific benefits and drawbacks. RNNs require less memory, particularly for handling long sequences. However, they suffer from the vanishing gradient problem and non-parallelizability in the time dimension during training, limiting their scalability (Hochreiter, 1998; Le and Zuidema, 2016).

1. 彭博是一名个人开发者，毕业于香港大学物理系。 RWKV 模型从设计，到优化，到大规模训练，全部由彭博一人完成。

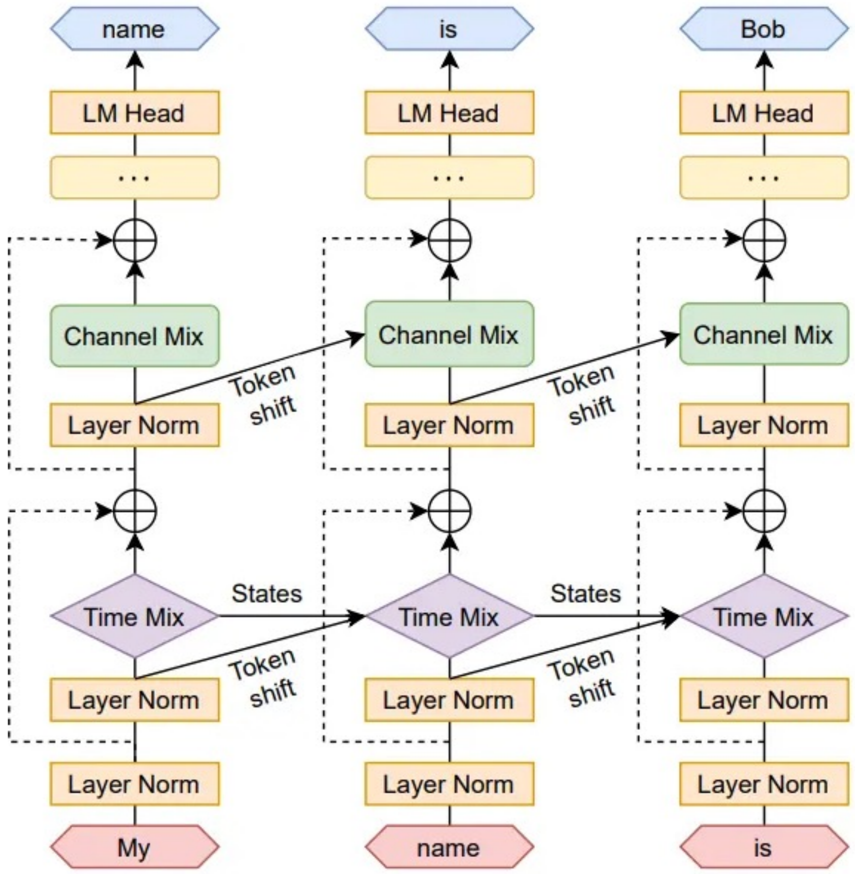
■2. 非 Transformer 的模型架构

2.3 RWKV 模型

•R：Receptance，作为过去信息的接受程度的接受向量
•W：Weight，位置权重衰减向量，可训练的模型参数
•K：键(Key)是类似于传统注意力中 K 的向量
•V：值(Value)是类似于传统注意力中 V 的向量



RNN



RWKV

2. 非 Transformer 的模型架构

2.3 RWKV 模型

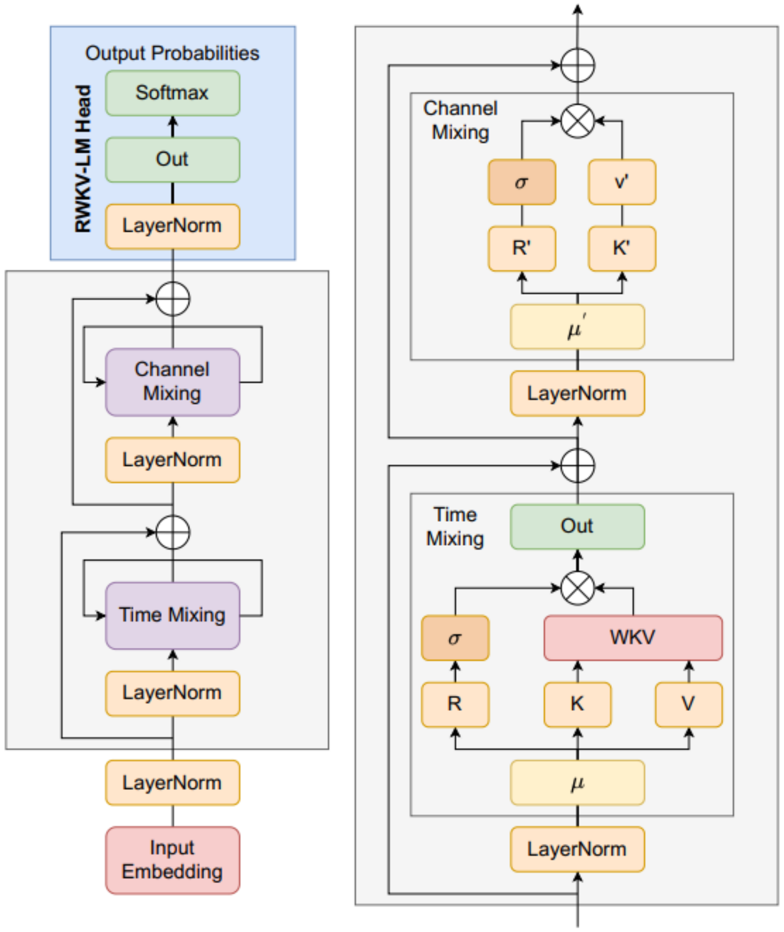


图 6.2: 左侧：RWKV 模型块的内部结构；右侧：RWKV 整体结构。

通道混合模块

$$\boldsymbol{R}'_i = W_r(\mu'_r \odot x_i + (1-\mu'_r) \odot x_{i-1})$$

$$\boldsymbol{K}'_i = W_k(\mu'_k \odot x_i + (1-\mu'_k) \odot x_{i-1})$$

$$\boldsymbol{O}_i = \sigma(\boldsymbol{R}'_i) \odot (W'_v \odot \max(\boldsymbol{K}'_i, 0)^2)$$

时间混合模块

$$\boldsymbol{R}_i = W_r(\mu_r \odot x_i + (1-\mu_r) \odot x_{i-1})$$

$$\boldsymbol{K}_i = W_k(\mu_k \odot x_i + (1-\mu_k) \odot x_{i-1})$$

$$\boldsymbol{V}_i = W_v(\mu_v \odot x_i + (1-\mu_v) \odot x_{i-1})$$

$$\mathrm{wkv}_i = \frac{\sum_{j=1}^{i-1} \exp\left(-(i-1-j)w + k_j\right) \odot v_j + \exp\left(u + k_i\right) \odot v_i}{\sum_{j=1}^{i-1} \exp\left(-(i-1-j)w_{k_j}\right) + \exp\left(u + k_i\right)}$$

$$\boldsymbol{O}_i = W_o \cdot (\sigma(\boldsymbol{V}_i) \odot \mathrm{wkv}_i)$$

目录

■3. 大模型架构配置

3.1 层归一化技术：回顾

$$\mathrm{LayerNorm}\left(X + \mathrm{MultiHeadAttention}(X)\right)$$

层归一化是一种正则化技术，X_i被映射成$(X_i - \mu)/\sigma$

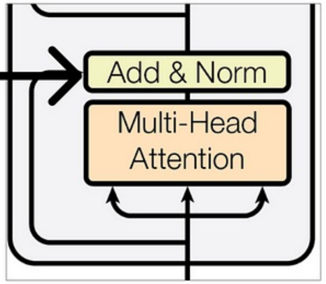
11.63	11.05	8.29	9.44	7.06	8.86
11.87	9.45	7.28	8.46	6.89	8.25
11.75	10.94	7.6	9.1	6.64	9.24
10.34	9.51	7.51	7.47	5.74	7.46
10.6	9.71	7.91	8.16	6.39	8.08
10.41	10.68	8.05	9.23	6.99	8.81

Mean	Std
9.26	1.57
8.56	1.64
9.04	1.76
7.86	1.51
8.37	1.35
8.93	1.28

$$\frac{\text{value} - \text{mean}}{\text{std} + \text{error}} = \frac{11.63 - 9.26}{1.57 + 0.0001}$$

=

1.51	1.14	-0.62	0.11	-1.4	-0.25
2.02	0.54	-0.78	-0.06	-1.02	-0.19
1.54	1.08	-0.82	0.03	-1.36	0.11
1.64	1.09	-0.23	-0.26	-1.4	-0.26
1.65	0.99	-0.34	-0.16	-1.47	-0.21
1.16	1.37	-0.69	0.23	-1.52	-0.09



■3. 大模型架构配置

3.1 层归一化技术

➢ 计算均值

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

➢ 计算方差

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

➢ 归一化

$$x_{\text{norm}} = \frac{x-\mu}{\sqrt{\sigma^2 + \text{Ú}}}$$

➢ 缩放和偏移

$$y = \gamma x_{\text{norm}} + \beta$$

层后归一化

- 自注意力机制或前馈网络之后进行
- 加快神经网络的训练收敛速度

层前归一化

- 自注意力机制或前馈网络之前进行
- 防止梯度爆炸或梯度消失

夹心归一化

- 层前归一化的基础上加额外归一化层
- 有时无法稳定大语言模型的训练

RMSNorm归一化

◆ 计算均方根

$$\text{RMS}(\mathbf{x}) = \sqrt{\frac{1}{H}\sum_{i=1}^{H} x_i^2}$$

◆ 归一化

$$\hat{\mathbf{x}}_i = \frac{x_i}{\text{RMS}(\mathbf{x})}$$

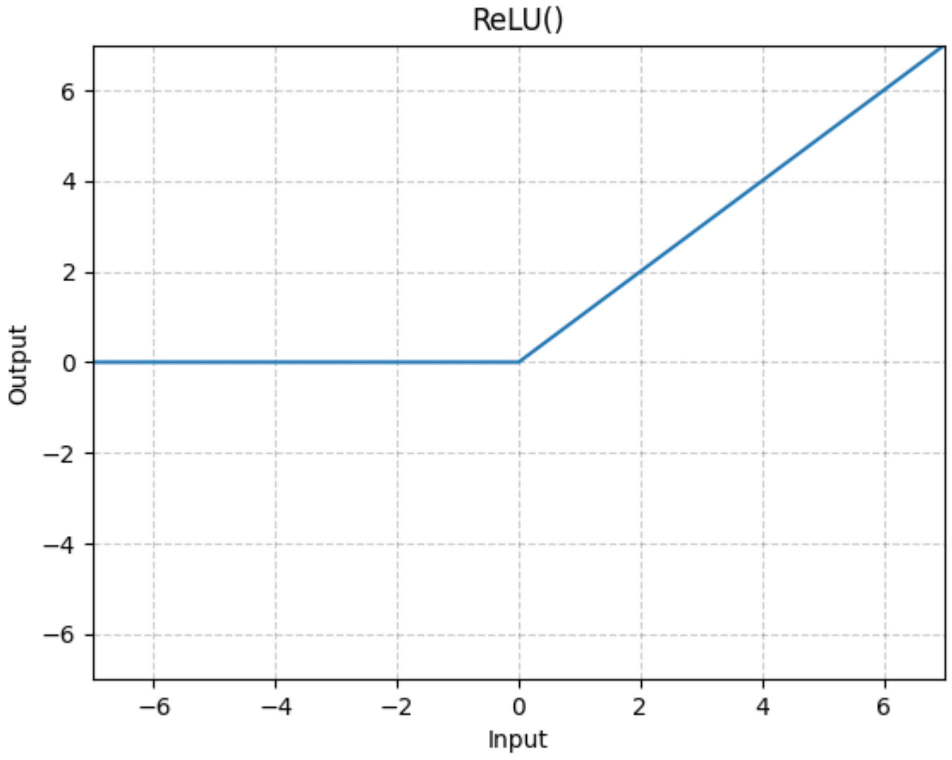
◆ 缩放（γ是可学习的参数）

$$\mathbf{y} = \gamma\hat{\mathbf{x}}$$

3.2 激活函数

原始版本使用ReLU（Rectified Linear Unit）

$$\mathrm{ReLU}(x) = (x)^{+} = \max(0, x)$$



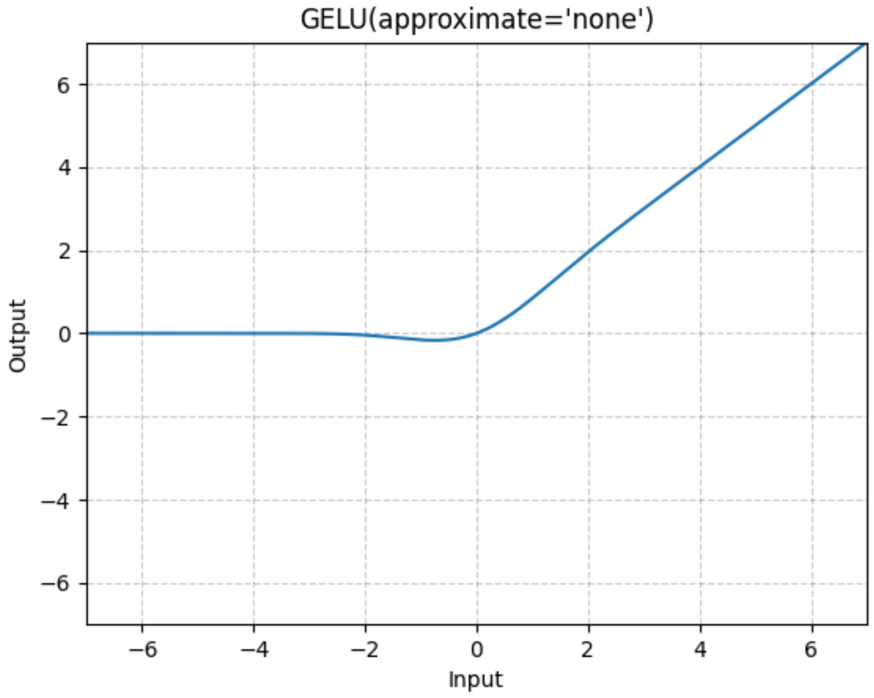
■3. 大模型架构配置

3.2 激活函数：GeLU

现有LLM广泛使用GeLU（Gaussian Error Linear Units）

$$\mathrm{GELU}(x) = x * \Phi(x)$$

where $\Phi(x)$ is the Cumulative Distribution Function for Gaussian Distribution.



GPT2Config

GPT-1和GPT-2使用的是GeLU

class transformers.GPT2Config <source>

```
( vocab_size = 50257, n_positions = 1024, n_embd = 768, n_layer = 12,
n_head = 12, n_inner = None, activation_function = 'gelu_new', resid_pdrop
= 0.1, embd_pdrop = 0.1, attn_pdrop = 0.1, layer_norm_epsilon = 1e-05,
initializer_range = 0.02, summary_type = 'cls_index', summary_use_proj =
True, summary_activation = None, summary_proj_to_labels = True,
summary_first_dropout = 0.1, scale_attn_weights = True, use_cache = True,
bos_token_id = 50256, eos_token_id = 50256,
scale_attn_by_inverse_layer_idx = False, reorder_and_upcast_attn = False,
**kwargs )
```

■3. 大模型架构配置

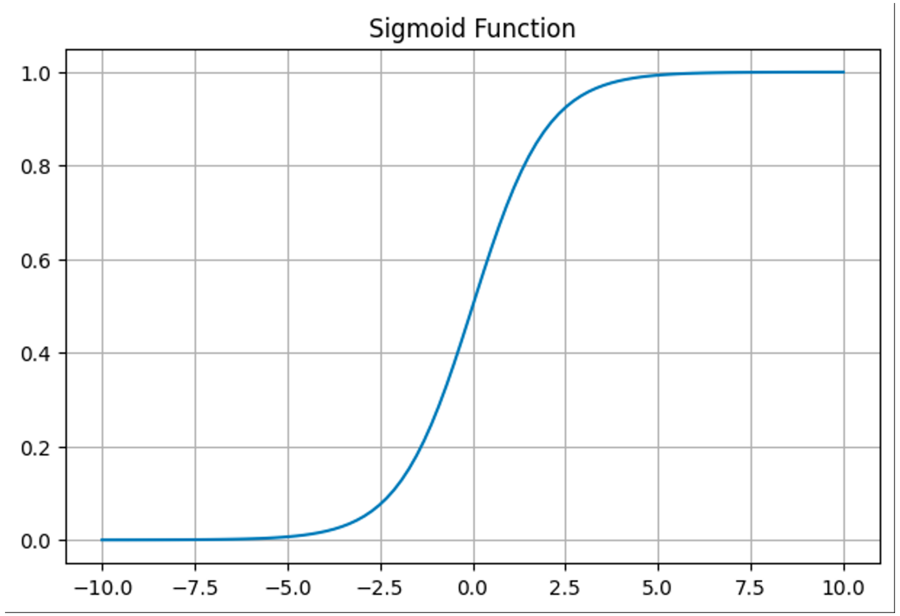
3.2 激活函数：GLU

课后试着绘制GLU的图

GLU（Gated Linear Unit）

$$\mathrm{GLU}(x, y) = x \odot \sigma(y)$$

x和y来自相同输入，经过不同线性变换




```python
class GLULayer(nn.Module):
    def __init__(self, input_dim, output_dim):
        super().__init__()
        self.linear = nn.Linear(input_dim, output_dim * 2)

    def forward(self, x):
        x, y = self.linear(x).chunk(2, dim=-1)
        return x * torch.sigmoid(y)
```

https://arxiv.org/pdf/1612.08083

3. 大模型架构配置

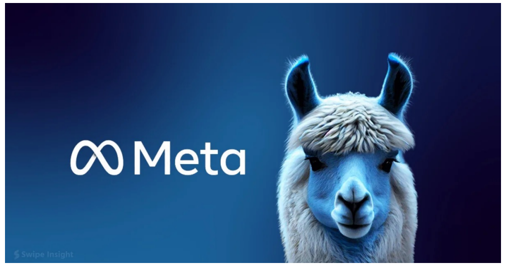
3.2 激活函数： SwiGLU

课后试着绘制SwiGLU的图

- Swish 激活函数：f(x) = x * sigmoid(βx)
- GLU(x, y) = x * sigmoid(y) 相等于 sigmod(xW + c) * (xV + c)

$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \mathrm{Swish}_\beta(xW + b) \otimes (xV + c)$$

Llama中使用的是SwiGLU



https://arxiv.org/pdf/1710.05941v1
https://arxiv.org/pdf/2002.05202

Pre-normalization [GPT3]. To improve the training stability, we normalize the input of each transformer sub-layer, instead of normalizing the output. We use the RMSNorm normalizing function, introduced by Zhang and Sennrich (2019).

SwiGLU activation function [PaLM]. We replace the ReLU non-linearity by the SwiGLU activation function, introduced by Shazeer (2020) to improve the performance. We use a dimension of $\frac{2}{3}4d$ instead of $4d$ as in PaLM.

3.2 激活函数： SwiGLU

课后试着绘制SwiGLU的图

- Swish 激活函数：f(x) = x * sigmoid(βx)
- GLU(x, y) = x * sigmoid(y) 相等于 sigmod(xW + c) * (xV + c)

$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \mathrm{Swish}_\beta(xW + b) \otimes (xV + c)$$

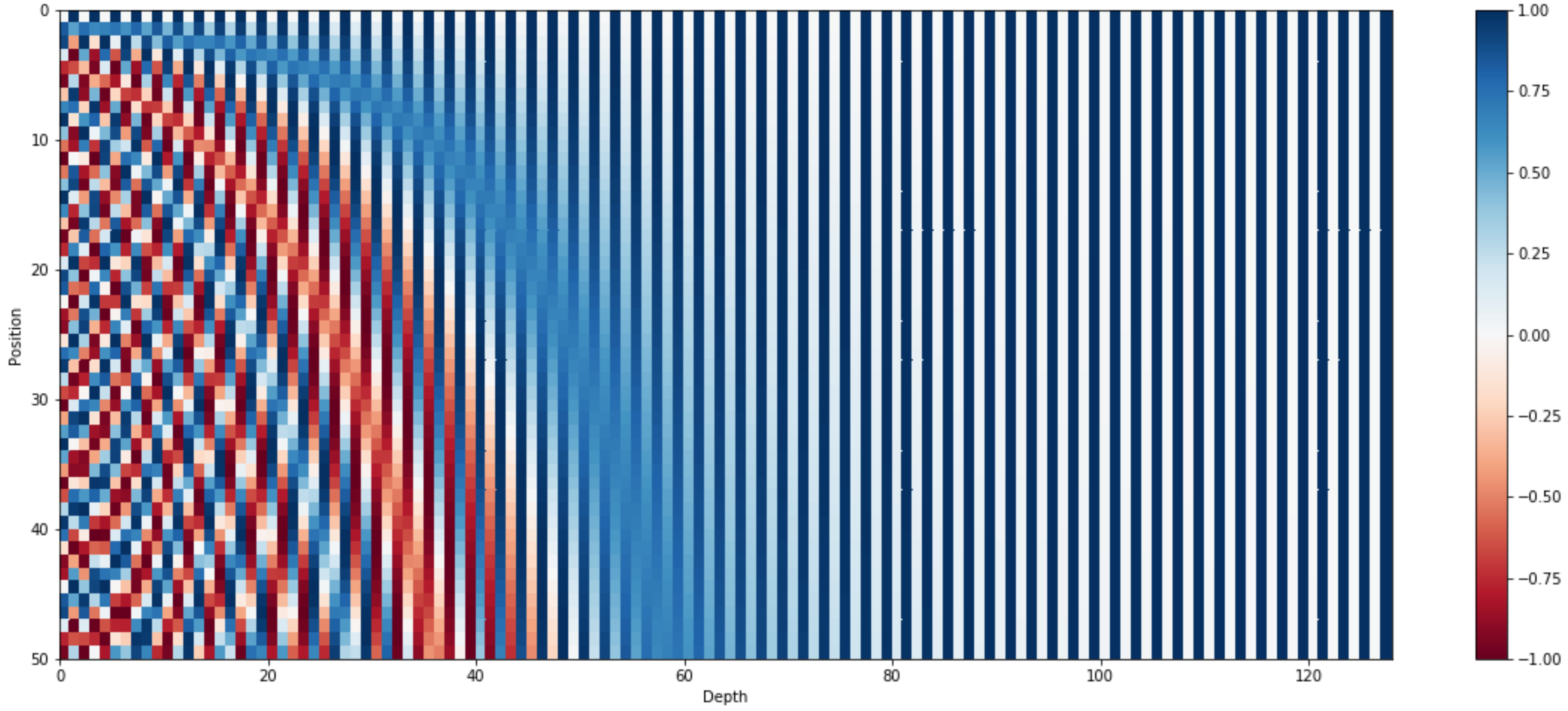
```python
class SwiGLULayer(nn.Module):
    def __init__(self, input_dim, output_dim, beta=1.0):
        super().__init__()
        self.linear = nn.Linear(input_dim, output_dim * 2)
        self.beta = beta

    def forward(self, input):
        x, y = self.linear(input).chunk(2, dim=-1)
        return x * (y * torch.sigmoid(self.beta * y))
```

■3. 大模型架构配置

3.3 正弦波位置编码

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$



■3. 大模型架构配置

3.3 正弦波位置编码

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

> *"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}."*

https://blog.timodenk.com/linear-relationships-in-the-transformers-positional-encoding/

■3. 大模型架构配置

3.3 旋转位置编码

绝对位置编码的两个主要弊端是:

1. **缺乏动态性**:绝对位置编码为每个位置分配一个预定义的固定向量,这意味着无论输入序列的内容如何变化,相同位置的编码向量始终保持不变。这种静态的编码方式限制了模型捕捉位置间动态关系的能力,因为真实的序列数据中,位置之间的关系可能会随着上下文的不同而变化。
2. **处理长距离依赖和长序列的能力有限**:由于绝对位置编码的固定性,它在处理长序列时可能无法有效地捕捉远距离位置之间的依赖关系。随着序列长度的增加,模型可能难以区分不同位置上的细微差别,尤其是在长距离上。这在处理复杂的语言结构或长文本时可能会成为一个问题,因为这些情况下往往需要模型能够理解和利用长距离的上下文信息。

旋转位置嵌入(Rotary Positional Embeddings)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \boxed{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

每个位置都与旋转角度θ关联

层级旋转位置编码:

1. 使每层神经网络从不同视角捕获位置信息
2. 初级层关注近邻位置关系,深层则捕获更远的位置关系
3. 增强了模型对序列数据全局结构和上下文的理解

https://arxiv.org/pdf/2104.09864

■3. 大模型架构配置

3.4 注意力与偏置

注意力机制

- 双向注意力机制 — 结合了正向和反向的上下文信息，能够捕获更全面的语义依赖关系 — BERT
- 单向注意力机制 — 只考虑一个方向上的上下文信息，即只从一个序列到另一个序列计算注意力权重 — GPT

■3. 大模型架构配置

技术	BERT	ALBERT	RoBERTa	GPT-1	GPT-2	GPT-3	GLM
Post-Layer Normalization	√	√	√	√			
Pre-Layer Normalization					√	√	
Sandwich-Layer Normalization							
RMSNorm		√					
SwiGLU							
Rotary Positional Embeddings						√	
双向注意力机制	√	√	√			√	√
单向注意力机制				√	√	√	

■4. 讨论

- **讨论1：** 如果你是一个自然语言处理领域的研究者，你会选择基于Transformer 架构的模型还是非 Transformer 的创新模型来解决特定的语言处理问题?请说明你的理由。

- **讨论2：** 编解码大语言模型看起来比编码大语言模型和解码大语言模型更具优势，请解释为什么目前的主流大模型是解码大语言模型?

谢谢！

第六章　大语言模型架构