博学谷
www.boxuegu.com
黑马程序员线上品牌

LLM主要类别架构介绍

一样的教育，不一样的品质

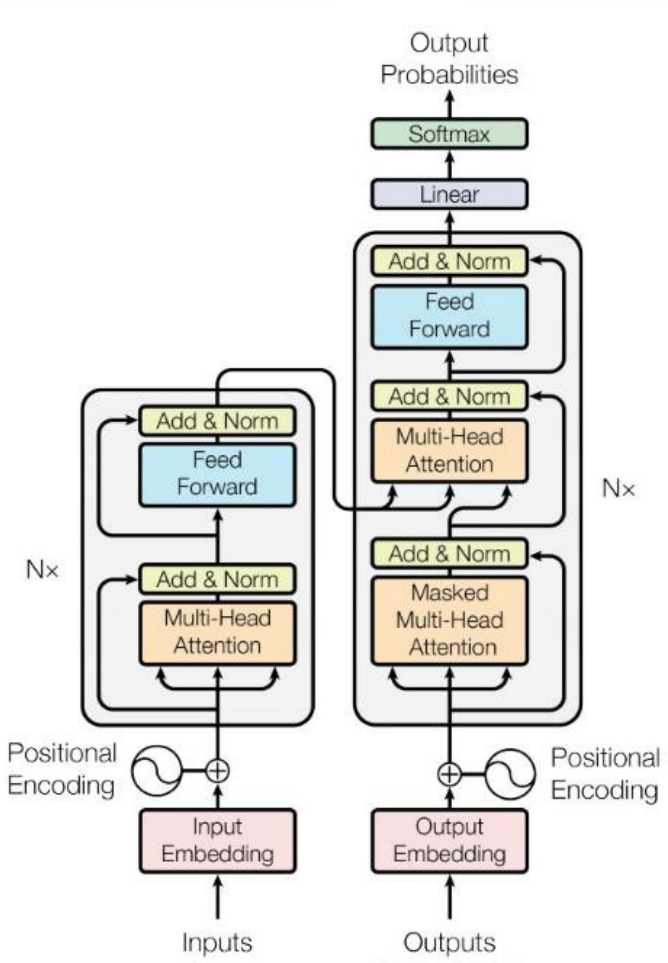
目录

Contents

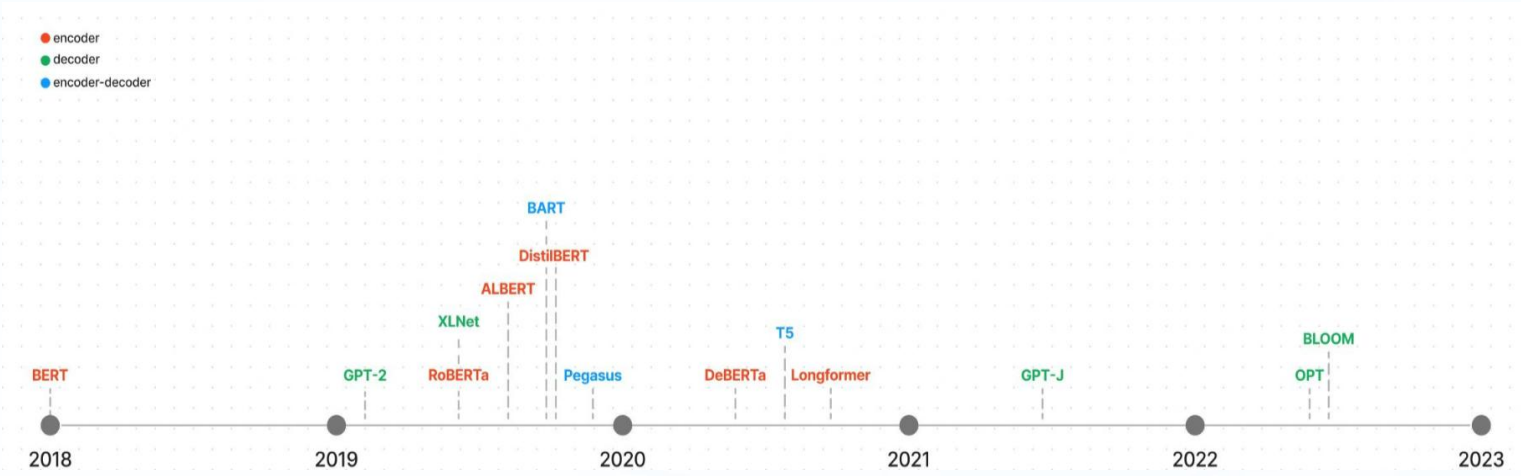
01

LLM主要类别

LLM主要类别



LLM本身基于transformer架构.
自2017年，attention is all you need诞生起，原始的transformer模型为不同领域的模型提供了灵感和启发.
基于原始的Transformer框架，衍生出了一系列模型，一些模型仅仅使用encoder或decoder，有些模型同时使用encoder+decoder.



LLM分类一般分为三种：自编码模型（encoder）、自回归模型(decoder)和序列到序列模型(encoder-decoder).

02

自编码模型

AutoEncoder model，AE

代表模型 BERT

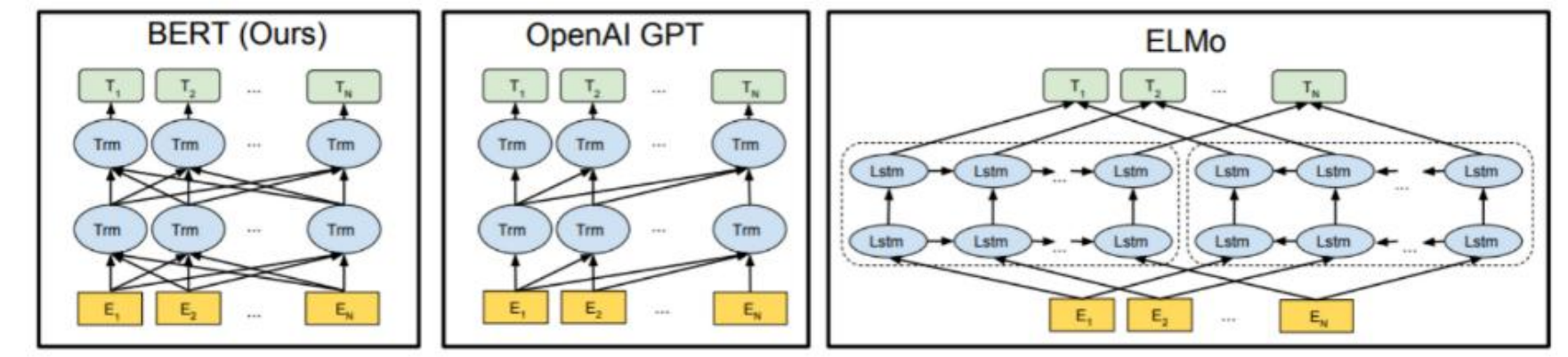
- AE模型，代表作BERT，其特点为：Encoder-Only。
- 基本原理：是在输入中随机MASK掉一部分单词，根据上下文预测这个词。
- AE模型通常用于内容理解任务，比如自然语言理解（NLU）中的分类任务：情感分析、提取式问答。

BERT是2018年10月由Google AI研究院提出的一种预训练模型.

- BERT的全称是Bidirectional Encoder Representation from Transformers.
- BERT在机器阅读理解顶级水平测试SQuAD1.1中表现出惊人的成绩: 全部两个衡量指标上全面超越人类, 并且在11种不同NLP测试中创出SOTA表现. 包括将GLUE基准推高至80.4% (绝对改进7.6%), MultiNLI准确度达到86.7% (绝对改进5.6%). 成为NLP发展史上的里程碑式的模型成就.

BERT的架构

总体架构：如下图所示，最左边的就是BERT的架构图，一个典型的双向编码模型.

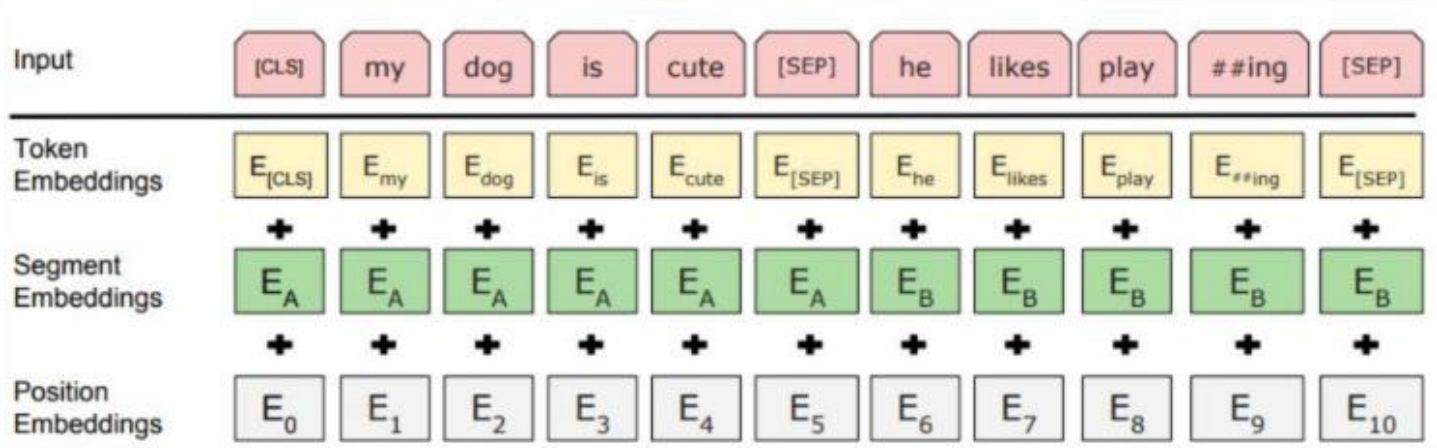


从上面的架构图中可以看到，宏观上BERT分三个主要模块：

- 最底层黄色标记的Embedding模块.
- 中间层蓝色标记的Transformer模块.
- 最上层绿色标记的预微调模块.

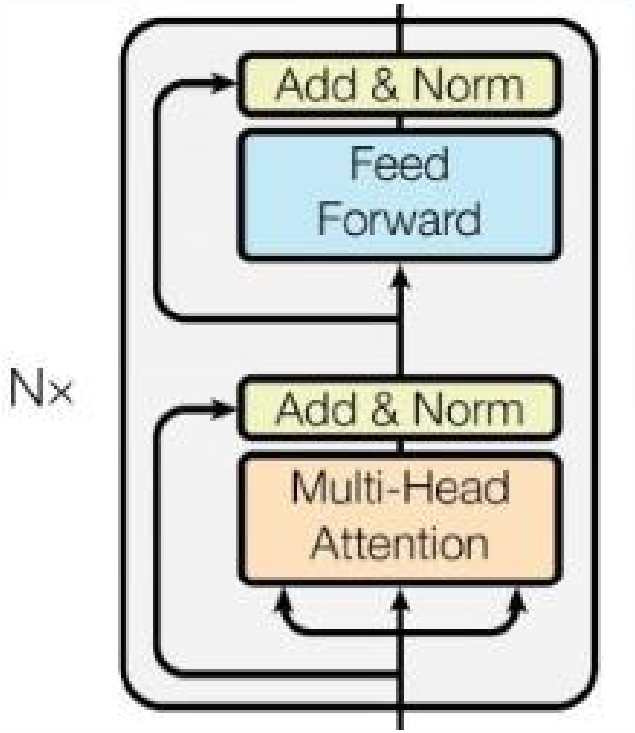
Embedding模块

BERT中的该模块是由三种Embedding共同组成而成，如下图



- Token Embeddings：词嵌入张量，第一个单词是CLS标志，可以用于之后的分类任务.
- Segment Embeddings：句子分段嵌入张量，是为了服务后续的两个句子为输入的预训练任务.
- Position Embeddings：位置编码张量.
- 整个Embedding模块的输出张量就是这3个张量的直接加和结果.

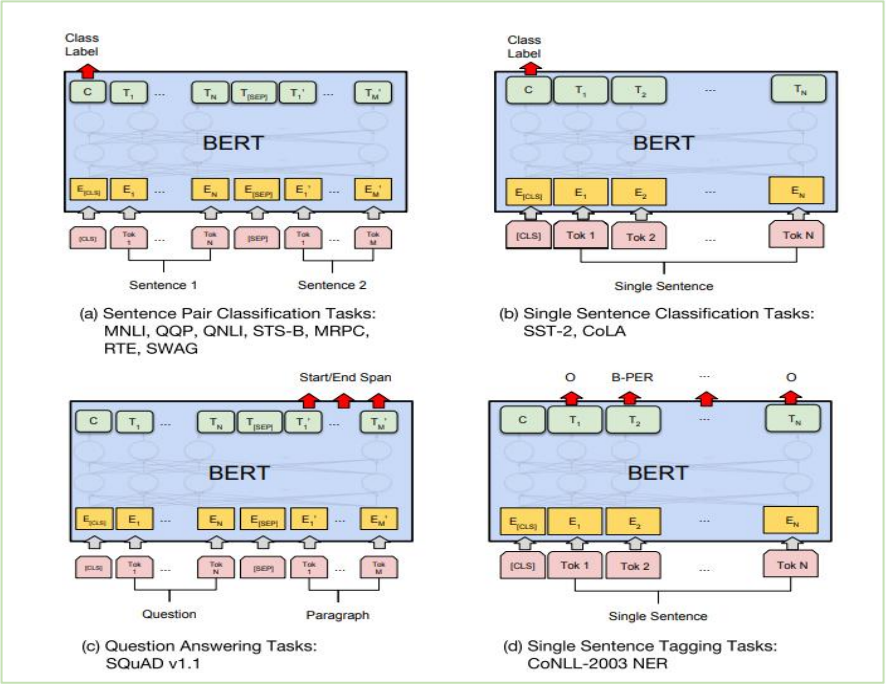
双向Transformer模块



- BERT中只使用了经典Transformer架构中的Encoder部分
- 完全舍弃了Decoder部分
- 两大预训练任务也集中体现在训练Transformer模块中

预微调模块

- 经过中间层Transformer的处理后，BERT的最后一层根据任务的不同需求而做不同的调整即可.



- 比如对于sequence-level的分类任务，BERT直接取第一个[CLS] token 的final hidden state，再加一层全连接层后进行softmax来预测最终的标签.

BERT的预训练任务

Masked LM（带mask的语言模型训练）

在原始训练文本中，随机的抽取15%的token作为参与MASK任务的对象.

- 80%的概率下，用[MASK]标记替换该token.
- 在10%的概率下，用一个随机的单词替换token.
- 在10%的概率下，保持该token不变.

Next Sentence Prediction（下一句话预测任务）

输入句子对(A，B)，模型来预测句子B是不是句子A的真实的下一句话.

- 所有参与任务训练的语句都被选中作为句子A.
- 其中50%的B是原始文本中真实跟随A的下一句话.(标记为IsNext，代表正样本).
- 其中50%的B是原始文本中随机抽取的一句话.(标记为NotNext，代表负样本).

数据集与BERT模型的特点

数据集如下:

BooksCorpus (800M words) + English Wikipedia (2,500M words)

模型的一些关键参数为:

参数	取值
transformer 层数	12
特征维度	768
transformer head数	12
总参数量	1.15亿

AE模型总结

优点

- BERT使用双向transformer，在语言理解相关的任务中表现很好.

缺点

- 输入噪声：预训练-微调存在差异.
- 更适合NLU任务，不适合用NLG任务.

思考总结

Thinking summary

1. 什么是自编码模型？

答案：是在输入中随机MASK掉一部分单词，根据上下文预测这个词。

2. BERT模型的核心架构？

答案：transformer的Encoder模块.

3.BERT的预训练任务？

答案：MLM 和 NSP

03

自回归模型

Autoregressive model, AR

代表模型GPT

AR模型，代表作GPT，其特点为：Decoder-Only。

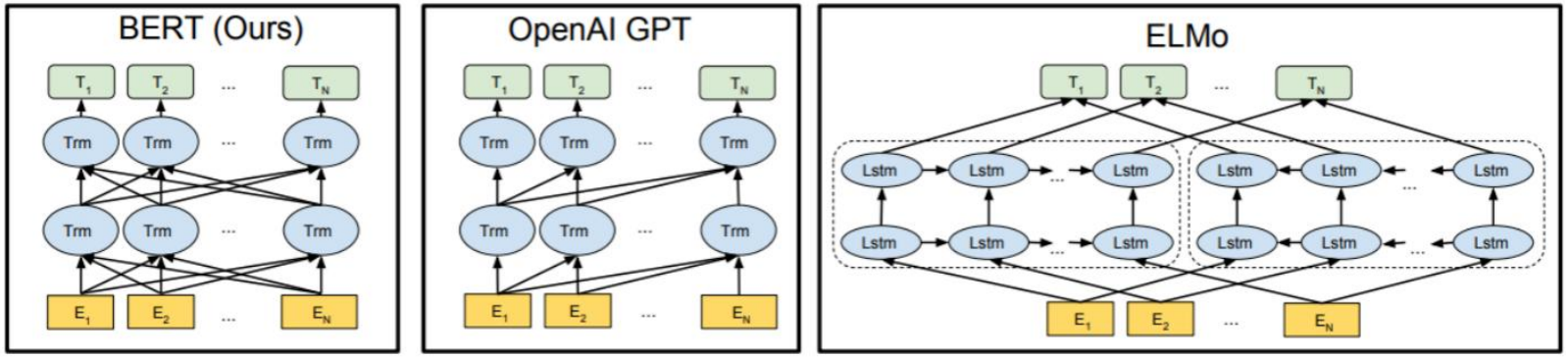
基本原理：从左往右学习的模型，只能利用上文或者下文的信息.

AR模型通常用于生成式任务，在长文本的生成能力很强，比如自然语言生成（NLG）领域的任务：摘要、翻译或抽象问答。

- 2018年6月，OpenAI公司发表了论文“Improving Language Understanding by Generative Pre-training”《用生成式预训练提高模型的语言理解力》，推出了具有1.17亿个参数的GPT（Generative Pre-training，生成式预训练）模型.

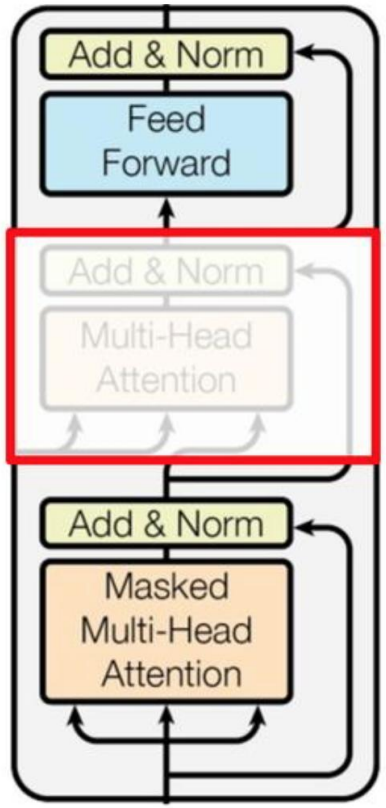
GPT模型架构

看三个语言模型的对比架构图，中间的就是GPT

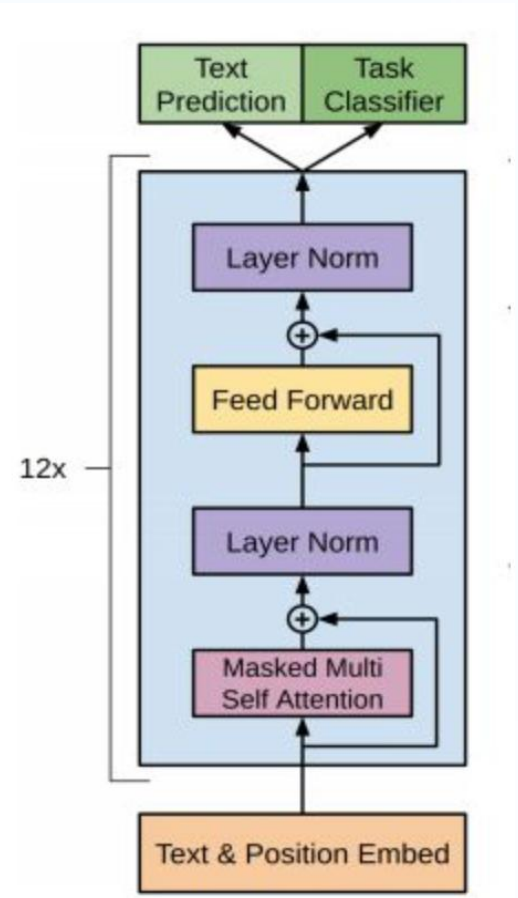


- 从上图可以很清楚的看到GPT采用的是单向Transformer模型，例如给定一个句子[u1，u2，...，un]，GPT在预测单词ui的时候只会利用[u1，u2，...，u(i-1)]的信息，而BERT会同时利用上下文的信息[u1，u2，...，u(i-1)，u(i+1)，...，un].

GPT模型架构



- GPT采用了Transformer的Decoder模块. 但是GPT的Decoder Block和经典Transformer Decoder Block还有所不同
- GPT中取消了第二个encoder-decoder attention子层, 只保留Masked Multi-Head Attention层,和Feed Forward层.
- 注意: 对比于经典的Transformer架构, 解码器模块采用了6个Decoder Block; GPT的架构中采用了12个Decoder Block.



GPT训练过程

GPT的训练包括两阶段过程：1.无监督的预训练语言模型；2.有监督的下游任务fine-tunning.

无监督的预训练语言模型

- 给定句子U = [u_1, u_2, ..., u_n]，GPT训练语言模型时的目标是最大化下面的似然函数：

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, \cdots, u_{i-1}; \Theta)$$

- 上述公式具体来说是要预测每个词u_i的概率，这个概率是基于它前面 u_{i-k} 到 u_{i-1} 个词，以及模型 Θ。这里的 k 表示上文的窗口大小，理论上来讲 k 取的越大，模型所能获取的上文信息越充足，模型的能力越强。

- GPT是一个单向语言模型,模型对输入U 进行特征嵌入得到 transformer 第一层的输h0，再经过多层 transformer 特征编码，使用最后一层的输出即可得到当前预测的概率分布，计算过程如下：

$$h_0 = UW_e + W_p$$

Wp代表单词的位置编码，形状是[max_seq_len, embedding_dim]，We的形状是[vocab_size, embedding_dim].

GPT训练过程

- 得到输入张量h0后，要将h0传入GPT的Decoder Block中，依次得到ht：

$$h_t = transformer\_block(h_{l-1}) \quad l \in [1,t]$$

- 最后通过得到的ht来预测下一个单词：

$$P(u) = softmax(h_t W_e^T)$$

有监督的下游任务fine-tunning

- GPT预训练后，会针对具体的下游任务对模型进行微调. 微调采用的是有监督学习，训练样本包括单词序列[x_1，x_2，...，x_n]和label y. GPT微调的目标任务是根据单词序列[x_1，x_2，...，x_n]预测标签y.

$$P(y|x^1,\cdots,x^m) = softmax(h_l^m W_y)$$

其中W_y表示预测输出的矩阵参数，微调任务的目标是最大化下面的函数：

$$L_2 = \sum_{(x,y)} \log P(y|x^1,\cdots,x^m)$$

综合两个阶段的目标任务函数，可知GPT的最终优化函数为：

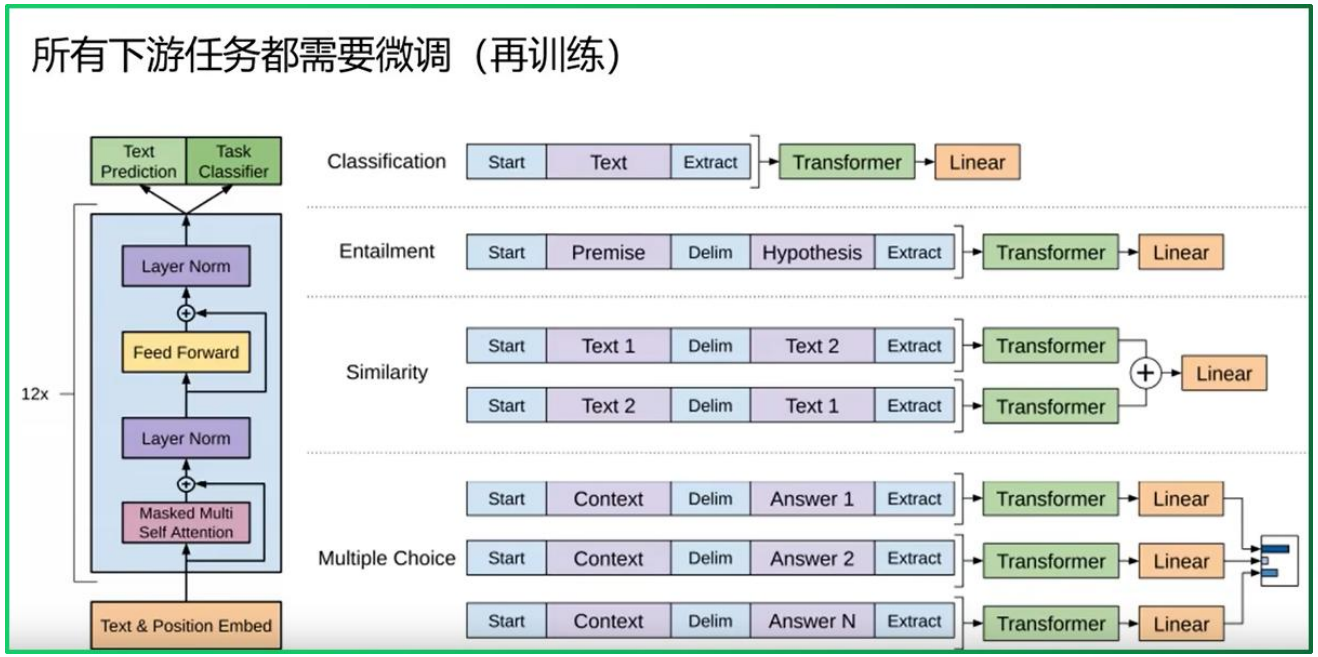
$$L_3 = L_2 + \lambda L_1$$

GPT训练过程

整体训练过程架构图

根据下游任务适配的过程分两步：1、根据任务定义不同输入，2、对不同任务增加不同的分类层.

具体定义可以参见下图:



GPT数据集

GPT使用了BooksCorpus数据集，文本大小约 5 GB，包含 7400w+ 的句子。这个数据集由 7000 本独立的、不同风格类型的书籍组成，选择该部分数据集的原因：

01

书籍文本包含大量高质量长句，保证模型学习长距离信息依赖。

02

书籍未开源公布，所以很难在下游数据集上见到，更能验证模型的泛化能力.

GPT模型的特点

模型的一些关键参数为：

参数	取值
transformer 层数	12
特征维度	768
transformer head 数	12
总参数量	1.17亿

优点

- GPT在9个任务上的表现sota
- 更易于并行化

缺点

- GPT 语言模型是单向的.
- 针对不同的任务，需要不同的数据集进行模型微调，相对比较麻烦

AR模型总结

优点

- AR模型擅长生成式任务.
- AR模型生成数据较容易.

缺点

- AR模型不能完全捕捉token的内在联系.

思考总结

Thinking summary

1. 什么是自回归模型？

答案：从左往右学习的模型，只能利用上文或者下文的信息.

2.GPT模型的核心架构？

答案：transformer的Decoder模块(去除中间的第二个子层).

3.GPT的预训练任务？

答案：无监督的预训练 和 有监督任务的微调

04

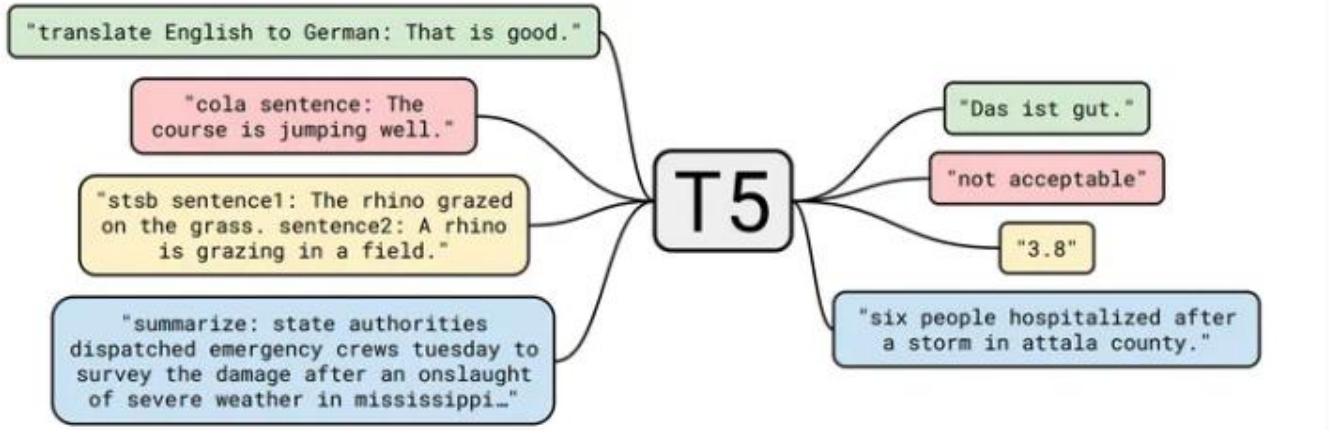
序列到序列模型

Sequence to Sequence Model

代表模型T5

encoder-decoder模型同时使用编码器和解码器. 它将每个task视作序列到序列的转换/生成（比如，文本到文本，文本到图像或者图像到文本的多模态任务）.Encoder-decoder模型通常用于需要内容理解和生成的任务，比如机器翻译.

T5 由谷歌提出，该模型的目的为构建任务统一框架：将所有NLP任务都视为文本转换任务.



通过这样的方式就能将 NLP 任务都转换成 Text-to-Text 形式，也就可以用同样的模型，同样的损失函数，同样的训练过程，同样的解码过程来完成所有 NLP 任务.

T5模型架构与训练过程

T5模型结构与原始的Transformer基本一致,除了做了以下几点改动：

- 采用了一种简化版的Layer Normalization，去除了Layer Norm 的bias；将Layer Norm放在残差连接外面.
- 位置编码：T5使用了一种简化版的相对位置编码，即每个位置编码都是一个标量，被加到 logits 上用于计算注意力权重。各层共享位置编码，但是在同一层内，不同的注意力头的位置编码都是独立学习的.

自监督预训练

T5 在预训练阶段采用了类似于 BERT 和 GPT 的大规模自监督学习策略，但与这两个模型的设计不同的是，T5 使用了“文本到文本”的格式来处理任务。
预训练任务包括两种类型：Causal Language Modeling（因果语言建模）和 填空任务（Masked Language Modeling）

多任务微调

除了使用大规模数据进行无监督预训练，T5模型还可以利用不同任务的标注数据进行有监督的多任务预训练，例如SQuAD问答和机器翻译等任务.

T5数据集与模型的特点

T5数据集如下：

作者对公开爬取的网页数据集Common Crawl进行了过滤，去掉一些重复的、低质量的，看着像代码的文本等，并且最后只保留英文文本，得到数据集C4：the Colossal Clean Crawled Corpus.

模型的一些关键参数为：

参数	取值
transformer 层数	24
特征维度	768
transformer head 数	12
总参数量	2.2亿

encoder-decoder模型总结

优点

- 处理多种NLP任务，具有良好的可扩展性；
- 相比GPT-2、GPT3等模型，T5参数量相对较少，训练速度更快.

缺点

- 训练时间较长、需要更大的算力.
- 模型的可解释性不足.

思考总结

Thinking summary

1. 什么是序列到序列模型？

答案：同时使用编码器和解码器，它将每个task视作序列到序列的转换/生成

2.T5模型的核心架构？

答案：transformer架构

3.T5的预训练任务？

答案：采用了类似于 BERT 和 GPT 的大规模自监督学习策略，预训练任务包括两种类型：Causal Language Modeling（因果语言建模）和 填空任务（Masked Language Modeling）

05

大模型主流模型架构

Decoder-only

Decoder-only

- LLM之所以主要都用Decoder-only架构，除了训练效率和工程实现上的优势外，在理论上是因为Encoder的双向注意力会存在低秩问题，这可能会削弱模型表达能力，就生成任务而言，引入双向注意力并无实质好处.

- 而Encoder-Decoder架构之所以能够在某些场景下表现更好，大概只是因为它多了一倍参数。所以，在同等参数量、同等推理成本下，Decoder-only架构就是最优选择了。

思考总结

Thinking summary

1. LLM主要类别架构？

答案：Encoder-Only、Decoder-Only、Encoder-Decoder

2. 自编码模型的基本原理？

答案：是在输入中随机MASK掉一部分单词，根据上下文预测这个词.

3. 自回归模型的基本原理？

答案：从左往右学习的模型，只能利用上文或者下文的信息.

4. 序列到序列模型的基本原理？

答案：同时使用编码器和解码器. 它将每个task视作序列到序列的转换/生成.

博学谷
www.boxuegu.com
黑马程序员线上品牌

扫码关注博学谷微信公众号