昇思MindSpore技术公开课
大模型专题

GPT

昇思 MindSpore

目录

GPT Introduction

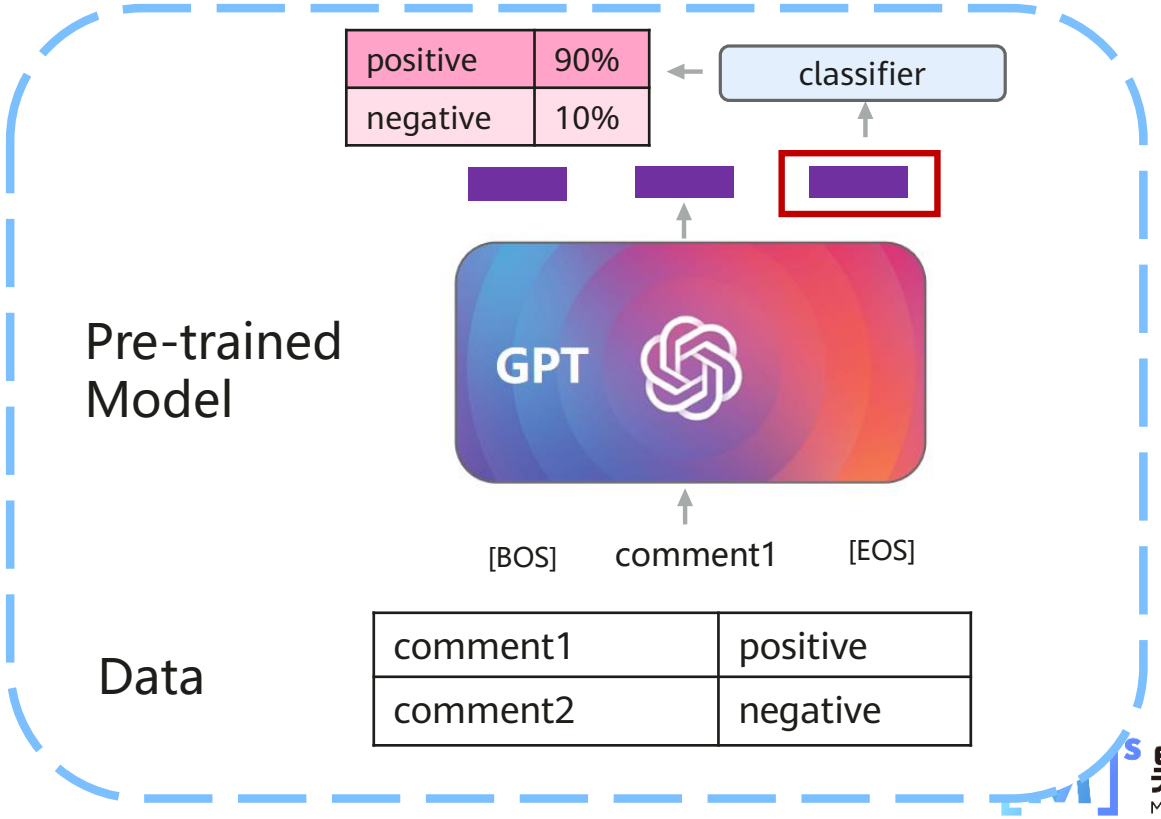
和BERT类似，GPT-1同样采取pre-train + fine-tune的思路：先基于大量未标注语料数据进行预训练，后基于少量标注数据进行微调。但GPT-1在预训练任务思路和模型结构上与BERT有所差别。

1. **Unsupervised Pre-training** on a large corpus of text data

The model is trained based on **a standard language modeling objective**, enabling it to learn intricate patterns, grammar, and language semantics and to generate coherent and contextually relevant text.

Model	GPT
Data	BooksCorpus dataset
Objective	maximize the likelihood using language modeling

2. **Supervised Fine-tuning** on a sepecific task using labelled data



GPT Input

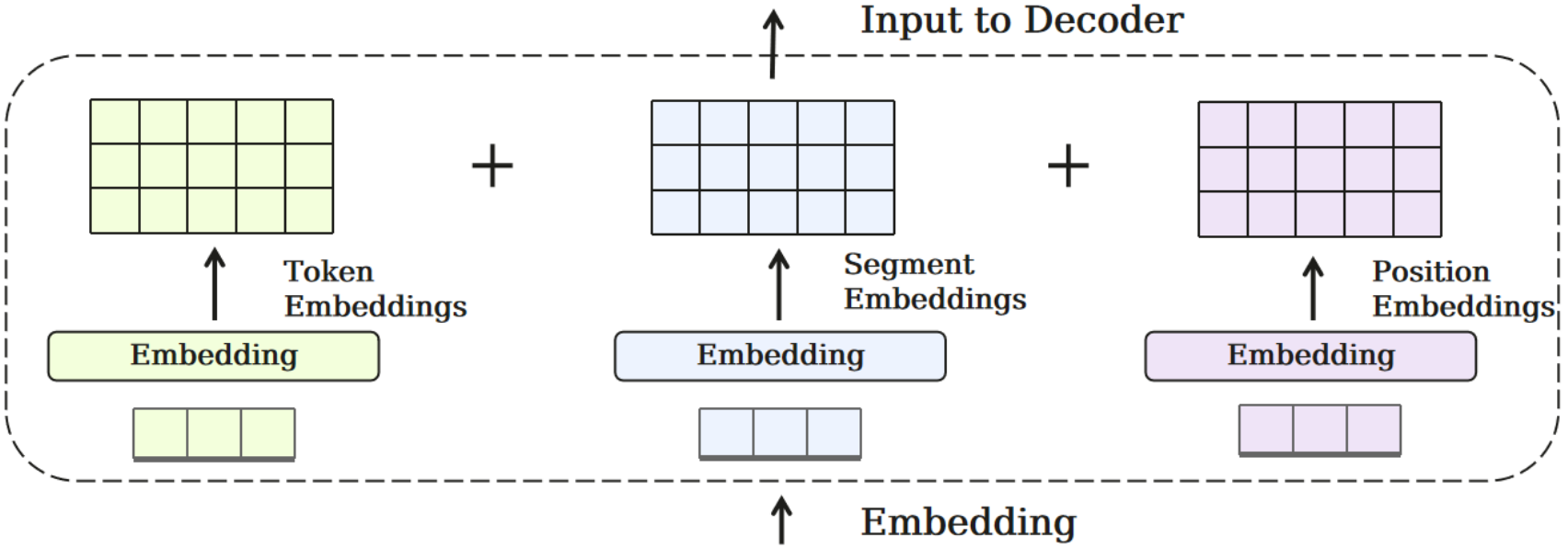
GPT-1的输入同样为句子或句子对组成，并添加special tokens。

- [BOS]：表示句子的开始，（论文中给出的token表示为[START]），添加到序列最前；
- [EOS]：表示序列的结束，（论文中的给出的[EXTRACT]），添加到序列最后，在进行分类任务时，会将该special token对应的输出接入输出层；我们也可以理解为该token可以学习到整个句子的语义信息；
- [SEP]：用于间隔句子对中的两个句子；

token type ids	0	0	0	0	0	0	0	0	0	0
position ids	0	1	2	3	4	5	6	7	8	9
token ids	40478	1150	3912	1061	16	5033	10533	2498	17817	40479
model input	[BOS]	help	prince	may	u	ko	transfer	huge	inheritance	[EOS]

GPT Embedding

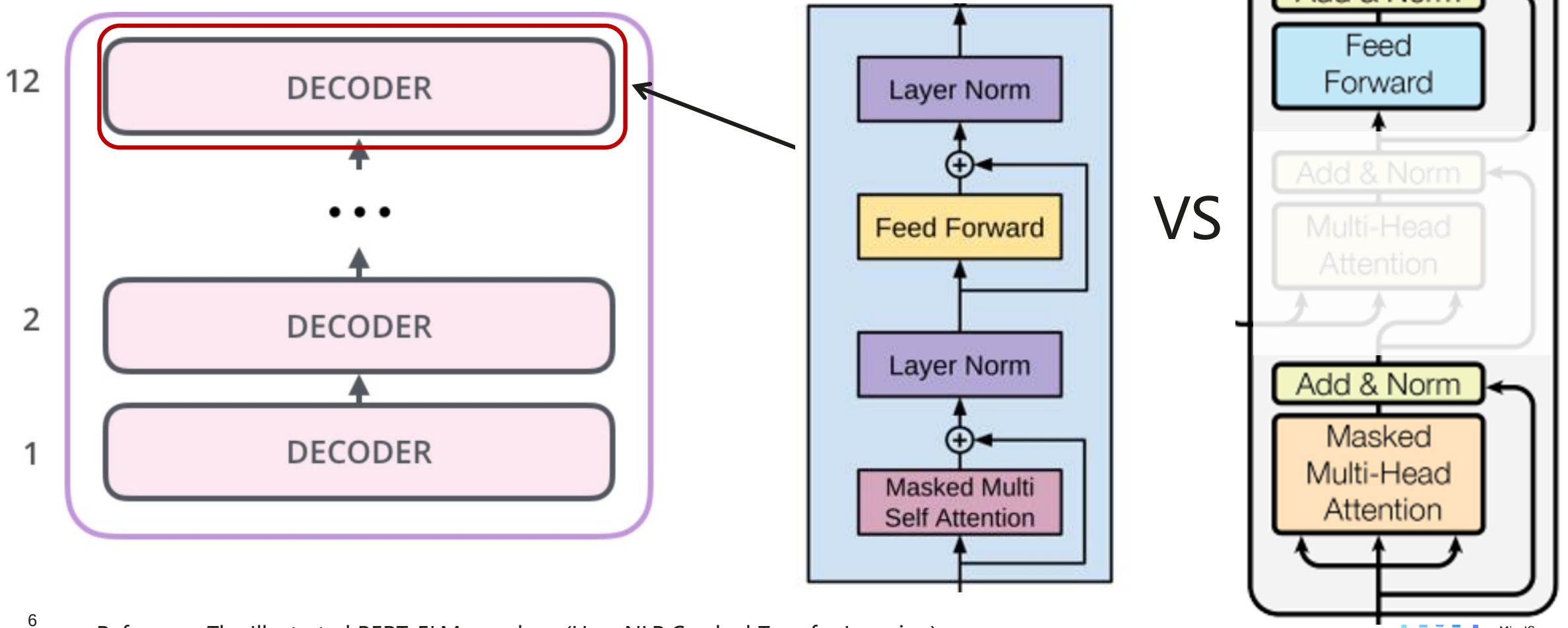
GPT Embedding 同样分为三类：token Embedding、Position Embedding、Segment Embedding



token type ids	0	0	0	0	0	0	0	0	0	0
position ids	0	1	2	3	4	5	6	7	8	9
token ids	40478	1150	3912	1061	16	5033	10533	2498	17817	40479
model input	[BOS]	help	prince	may	u	ko	transfer	huge	inheritance	[EOS]

GPT Model Architecture

GPT由Decoder Layer堆叠而成，Decoder Layer组成与Transformer Decoder Layer相似，只不过没有了计算encoder输出和decoder输入之间注意力分数的multi-head attention。

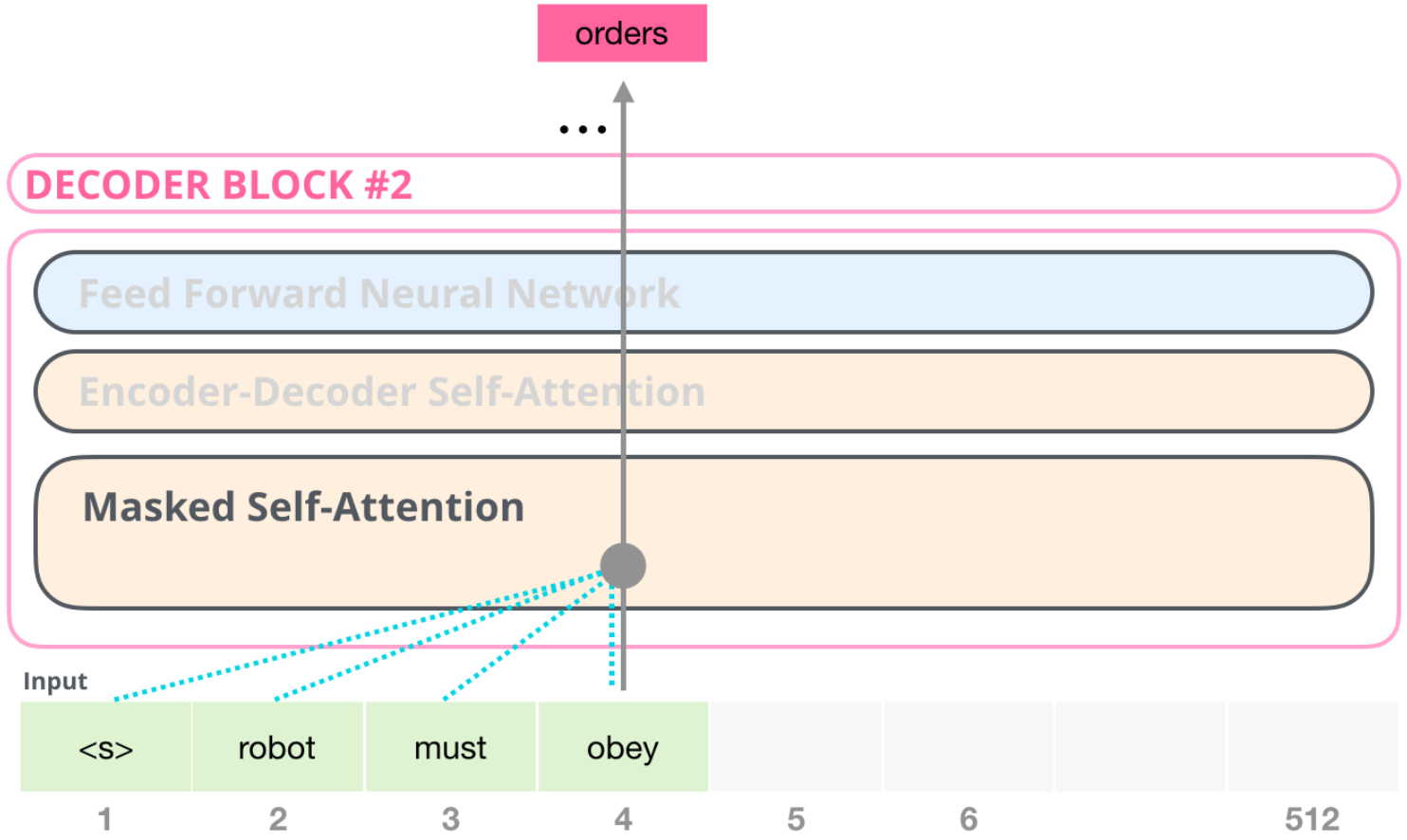


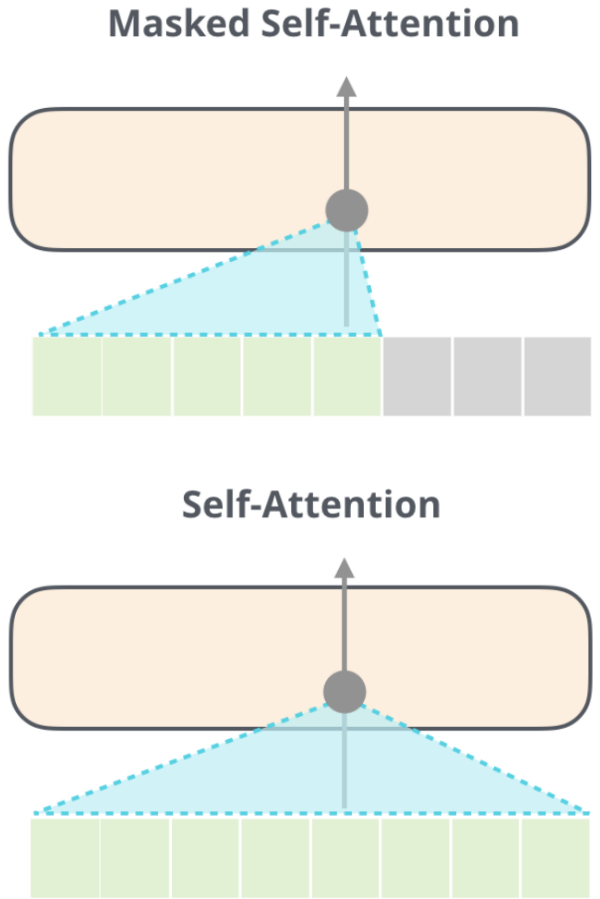
Reference: The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)

GPT Model Architecture

【来浅浅复习一下】Decoder layer中的masked self-attention

在解码器的自注意层中，遮盖了未来时刻的token —— **文本生成**是一个**单向**的过程，即我们无法预见未来





[Reference] http://jalammar.github.io/illustrated-gpt2/

昇思 MindSpore

GPT Model Specification

- 12 layers
- Hidden size of 768
- Vocabulary of 40,478 tokens was used.
- Batch size of 64
- Context window of 512 tokens were used.

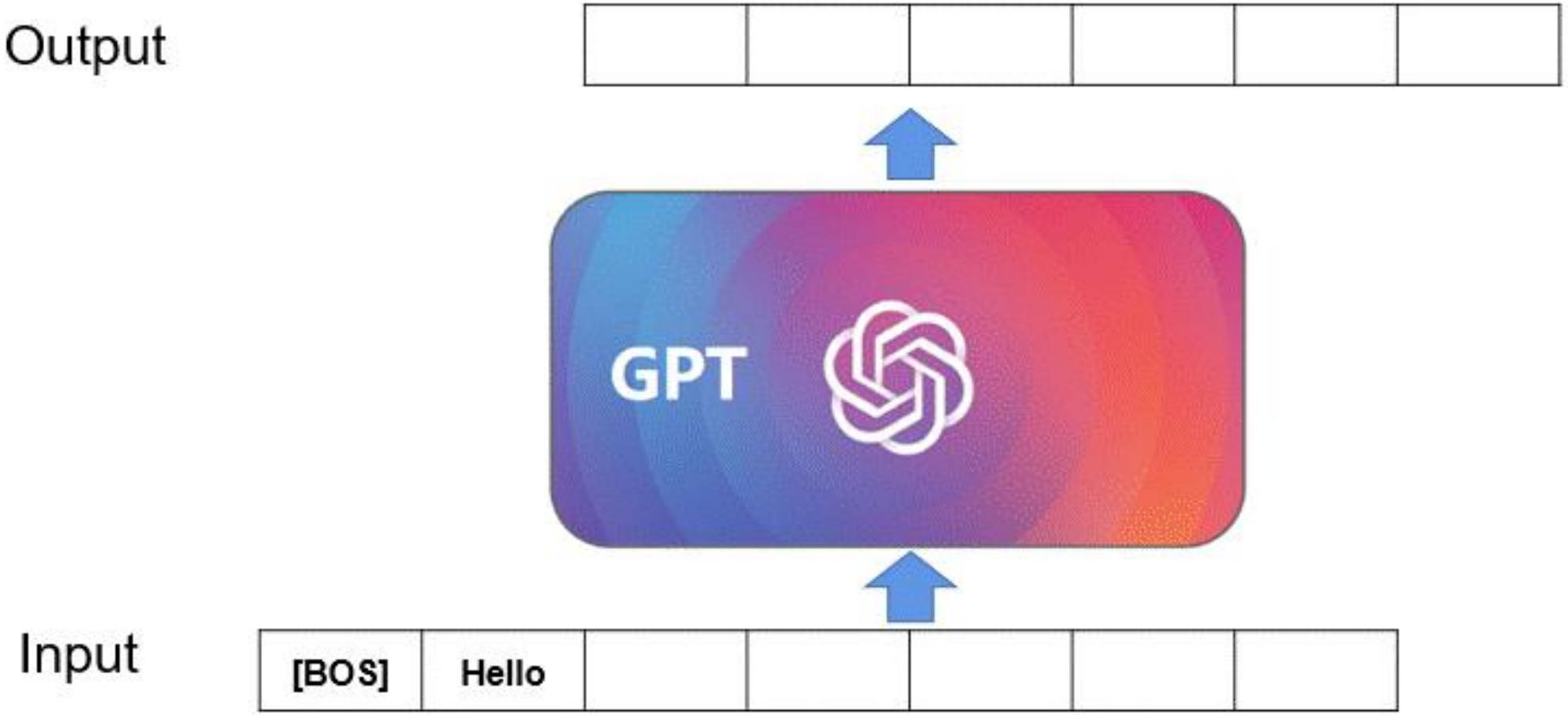
```
class GPTConfig(PreTrainedConfig):
    r"""
    GPT config
    """
    model_type = "gpt"
    attribute_map = {
        "max_position_embeddings": "n_positions",
        "hidden_size": "n_embd",
        "num_attention_heads": "n_head",
        "num_hidden_layers": "n_layer",
    }

    pretrained_config_archive_map = CONFIG_ARCHIVE_MAP

    def __init__(
        self,
        vocab_size=40478,
        n_positions=512,
        n_embd=768,
        hidden_size=768,
        n_layer=12,
        n_head=12,
        afn="gelu_new",
        resid_pdrop=0.1,
        embd_pdrop=0.1,
        attn_pdrop=0.1,
        layer_norm_epsilon=1e-5,
        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,
        **kwargs
    ):
```


[Reference] improving Language Understanding by Generative Pre-Training

~~GPT Output~~

GPT —— Generative(?) Pre-trained Transformer

相比于BERT注重文本理解，GPT侧重的是语句生成，即基于之前的文本序列，预测下一个词语。



~~GPT Output~~

GPT —— Generative(?) Pre-trained Transformer

因此，GPT很适合应用于**生成式（NLG）**的下游任务中，如：

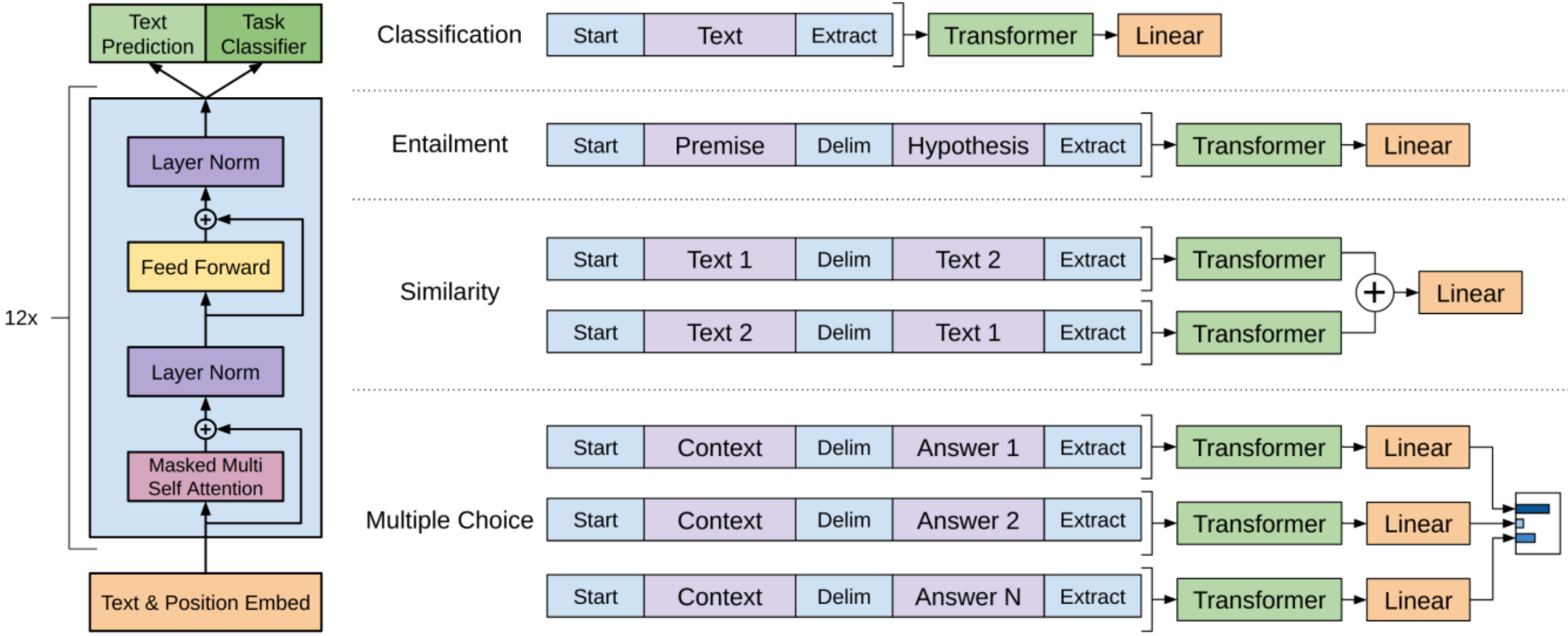
- 文本摘要（Summerization）
- 机器翻译（Machine Translation）
- 问答（Question Answering）
- …

如果是理解式的任务，该怎么办？

~~GPT Output~~

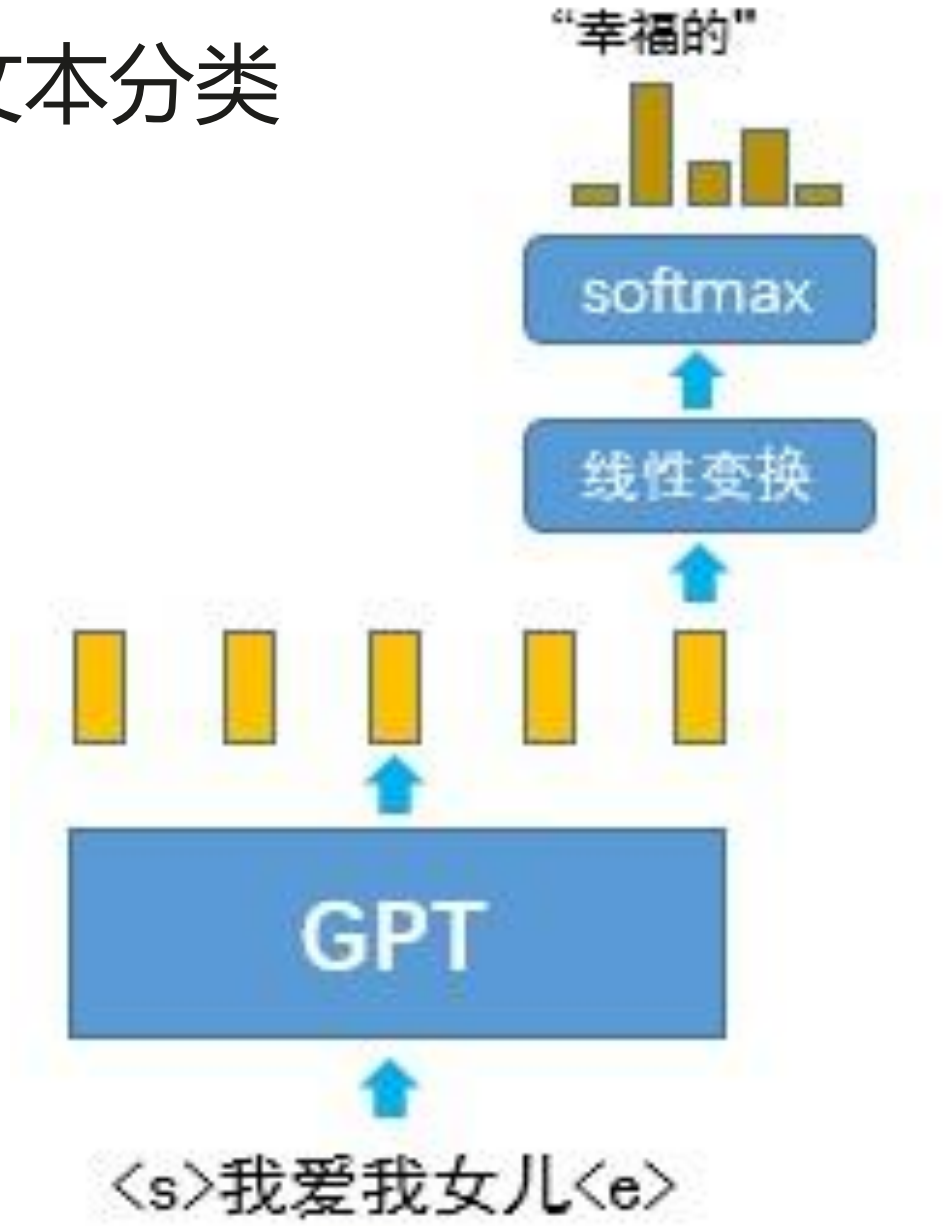
GPT —— Generative(?) Pre-trained Transformer

GPT按照生成式的逻辑统一了下游任务的应用模板，使用最后一个token（[EOS] or [EXTRACT]）对应的隐向量，输入额外的输出层中，进行分类标签预测。

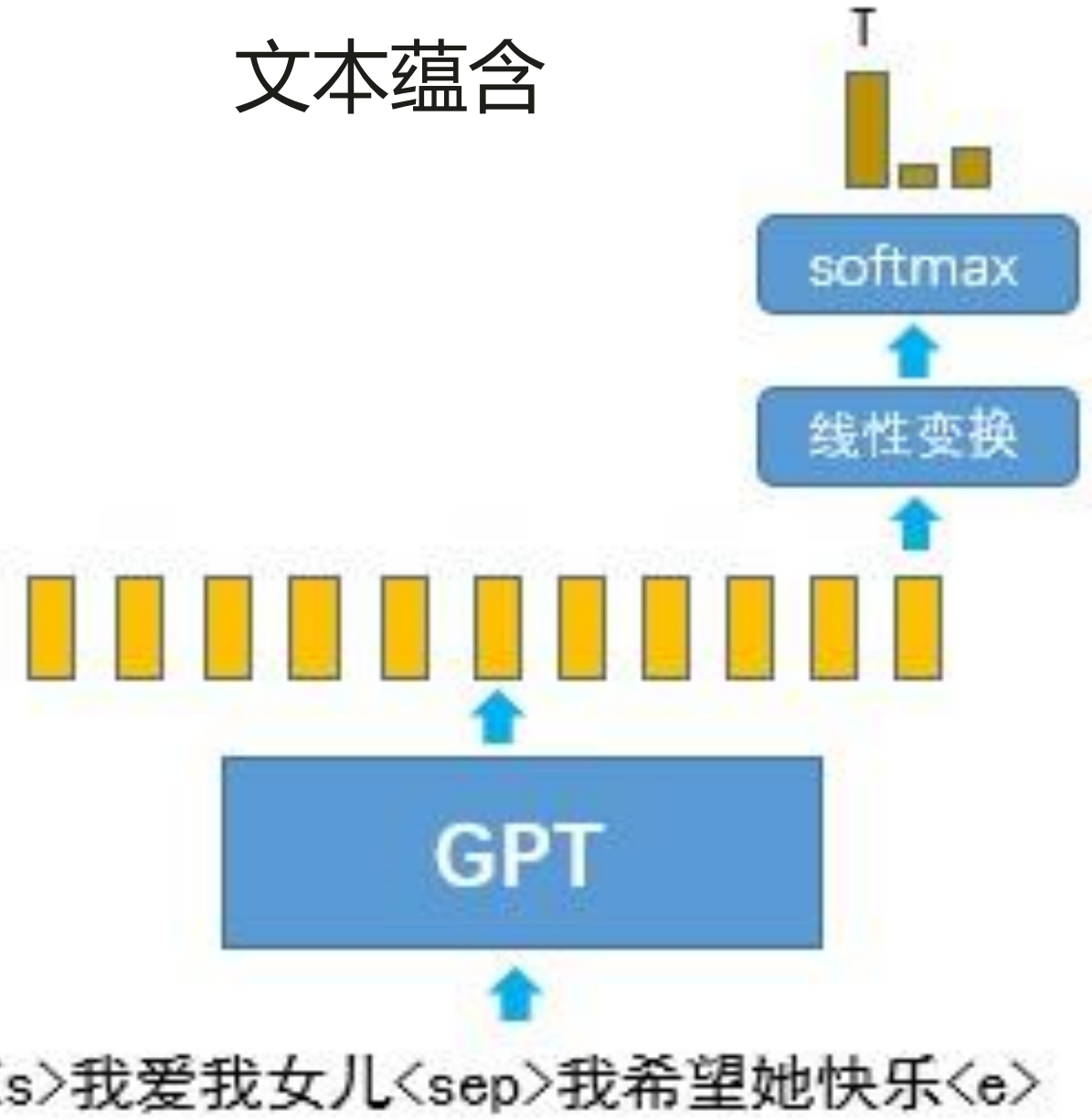


[Reference] improving Language Understanding by Generative Pre-Training

文本分类

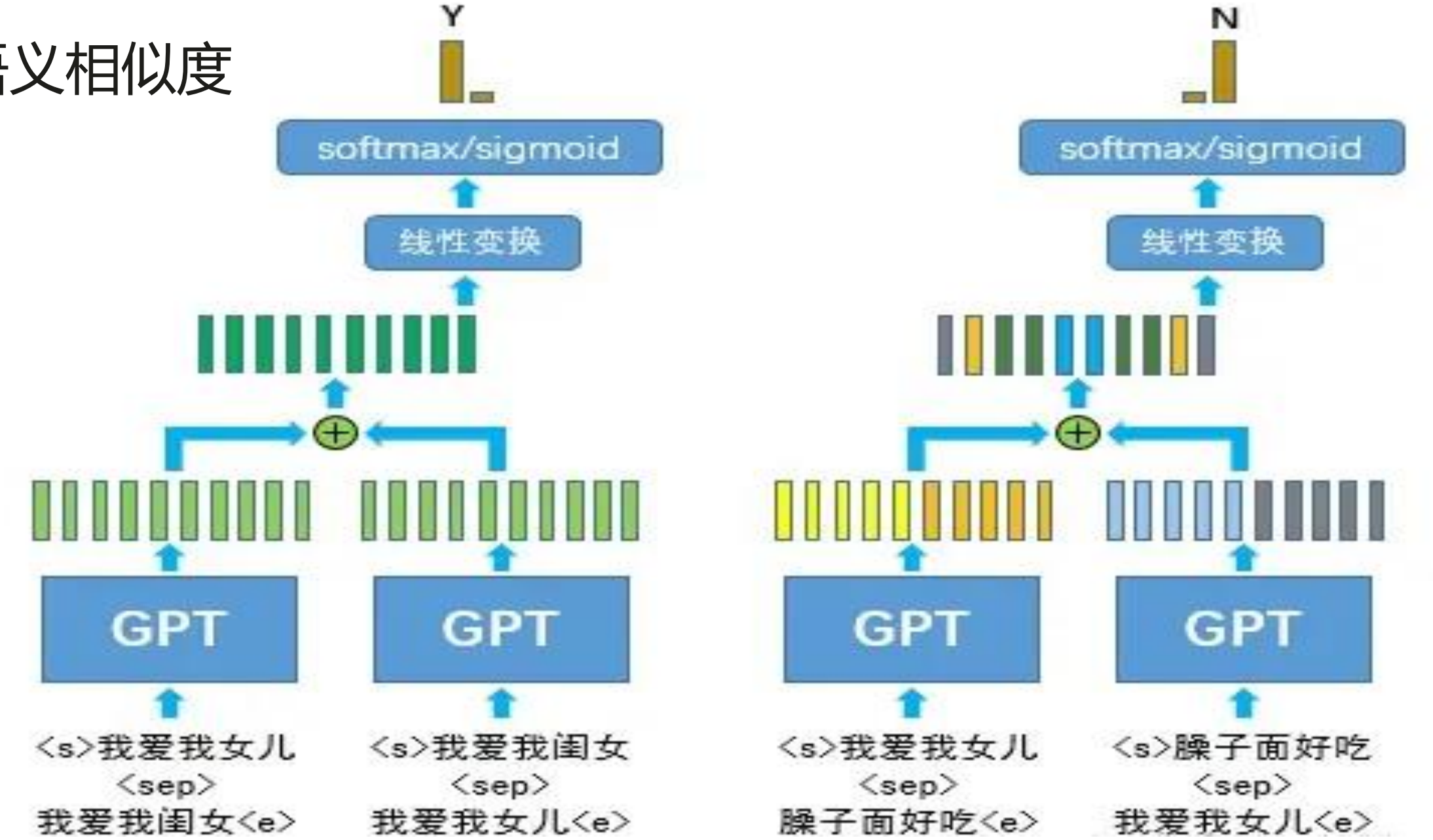


文本蕴含



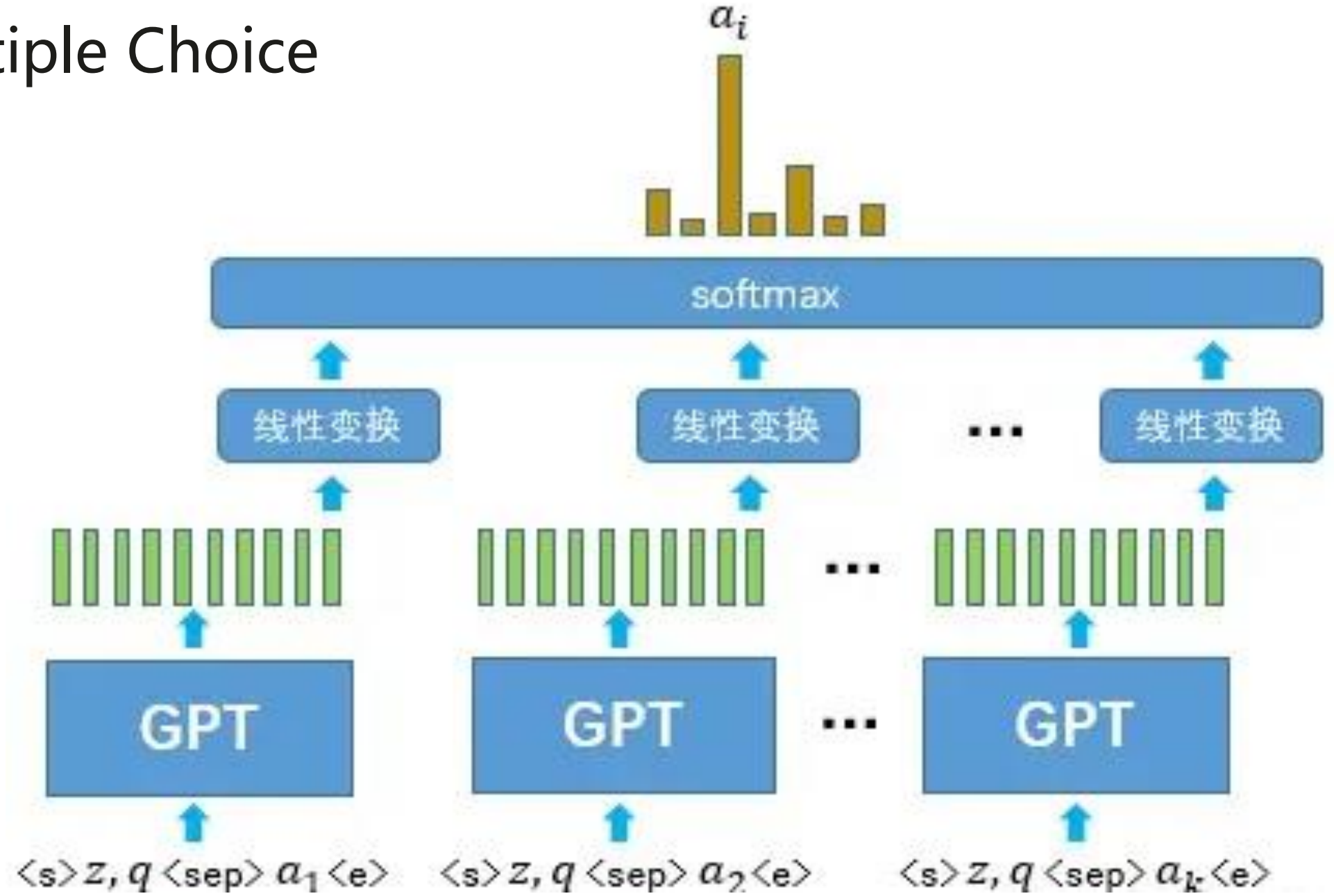
Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

语义相似度



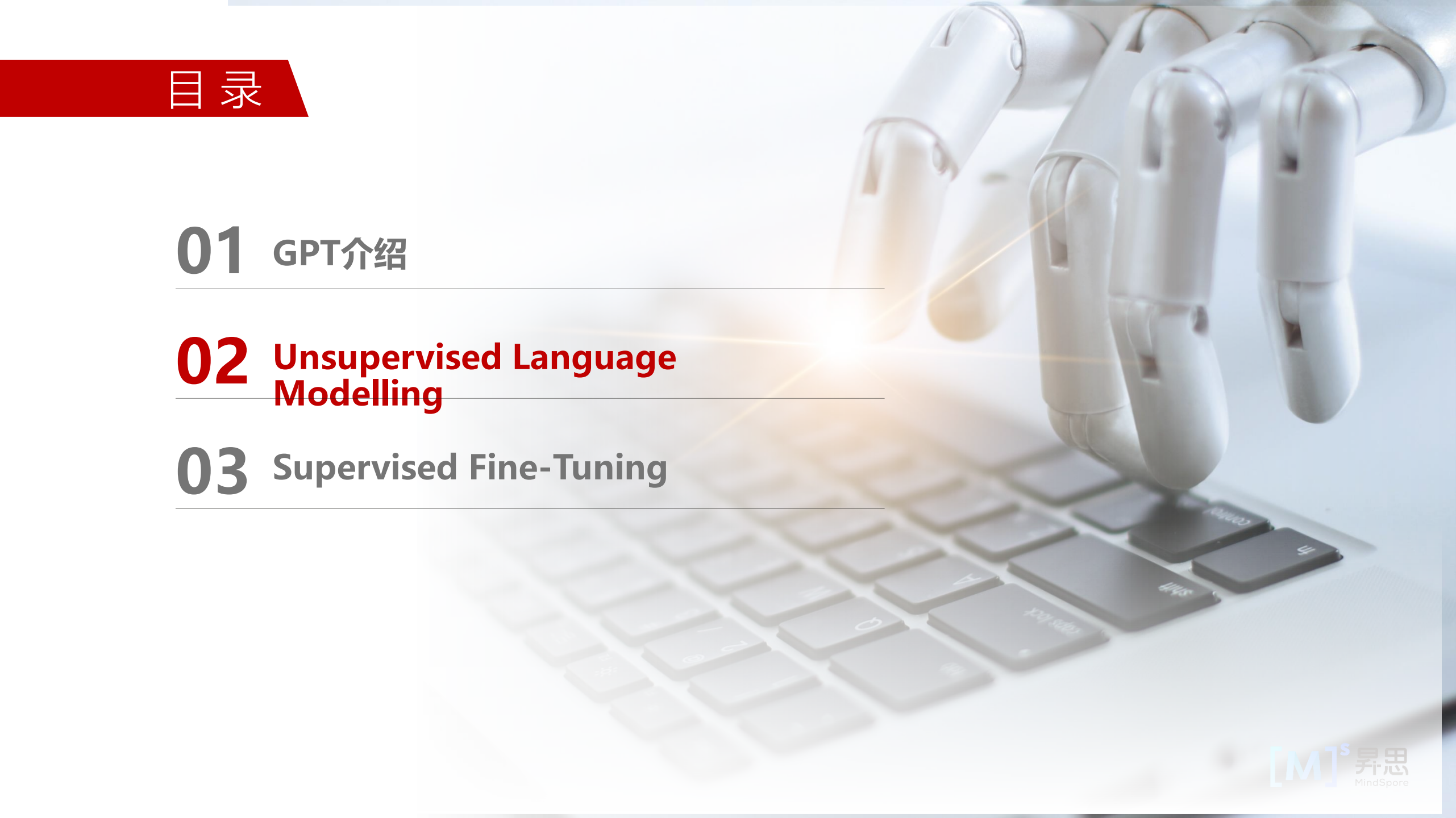
Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

Multiple Choice



Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

目录



Language Modeling Objective

假设我们有一组由n个词汇组成的序列：

$$\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$$

我们需要判断这个序列组成的句子是不是在说人话。通过计算整个序列的概率P(U)，我们认为概率越高，序列组成的句子越符合我们日常说话的方式。

$$p(\mathcal{U}) = p(u_1, u_2, \cdots, u_n)$$

p（“人哪有不发疯的，硬撑罢了”）> p（“狗玻甜由刘恐貌品哟”）

Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

Language Modeling Objective

按照链式法则（Chain Rule），我们可以把概率计算进一步转换成：

$$
\begin{aligned}
p(\mathcal{U}) &= p(u_1, u_2, \cdots, u_n) \\
&= p(u_1)p(u_2 \mid u_1)p(u_3 \mid u_1, u_2) \cdots p(u_n \mid u_1, \cdots, u_{n-1}) \\
&= \prod_{i=1}^{n} p(u_i \mid u_1, \cdots, u_{i-1})
\end{aligned}
$$

是不是很眼熟？

u_i: 当前被预测的下一个单词

$u_1 \ldots u_{i-1}$：前序内容，模型基于前序词汇进行下一单词预测

i：窗口长度（context window length），即需要往前看多少个单词

Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

Language Modeling Objective

在模型训练中，我们的目标是最大化P(U)

—— 让生成的句子看起来更像人话：

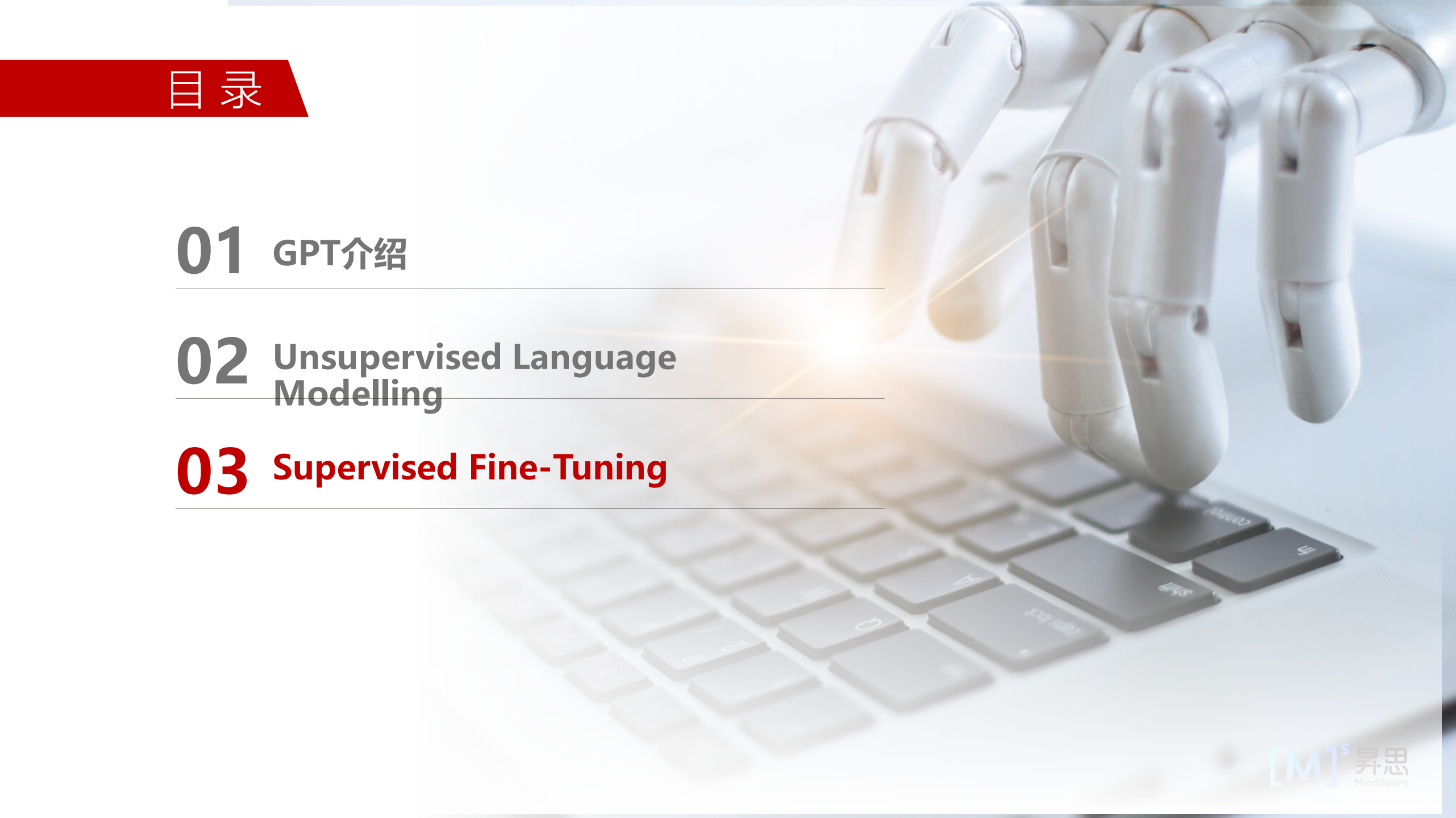
$$\theta^* = \arg\min_{\theta} -\sum_{i=1}^{n} \log p(u_i \mid u_{i-k}, \cdots, u_{i-1}; \theta)$$

- θ：模型参数
- k：窗口长度（受内存、算力等约束，模型没办法看到前序所有单词）

注：该训练无需进行任何标注，为无监督训练（Unsupervised Training）

Reference：[知乎]“追星”Transformer（二）：基于Transformer的预训练模型GPT by 铁心核桃

目录



Supervised Fine-tuning

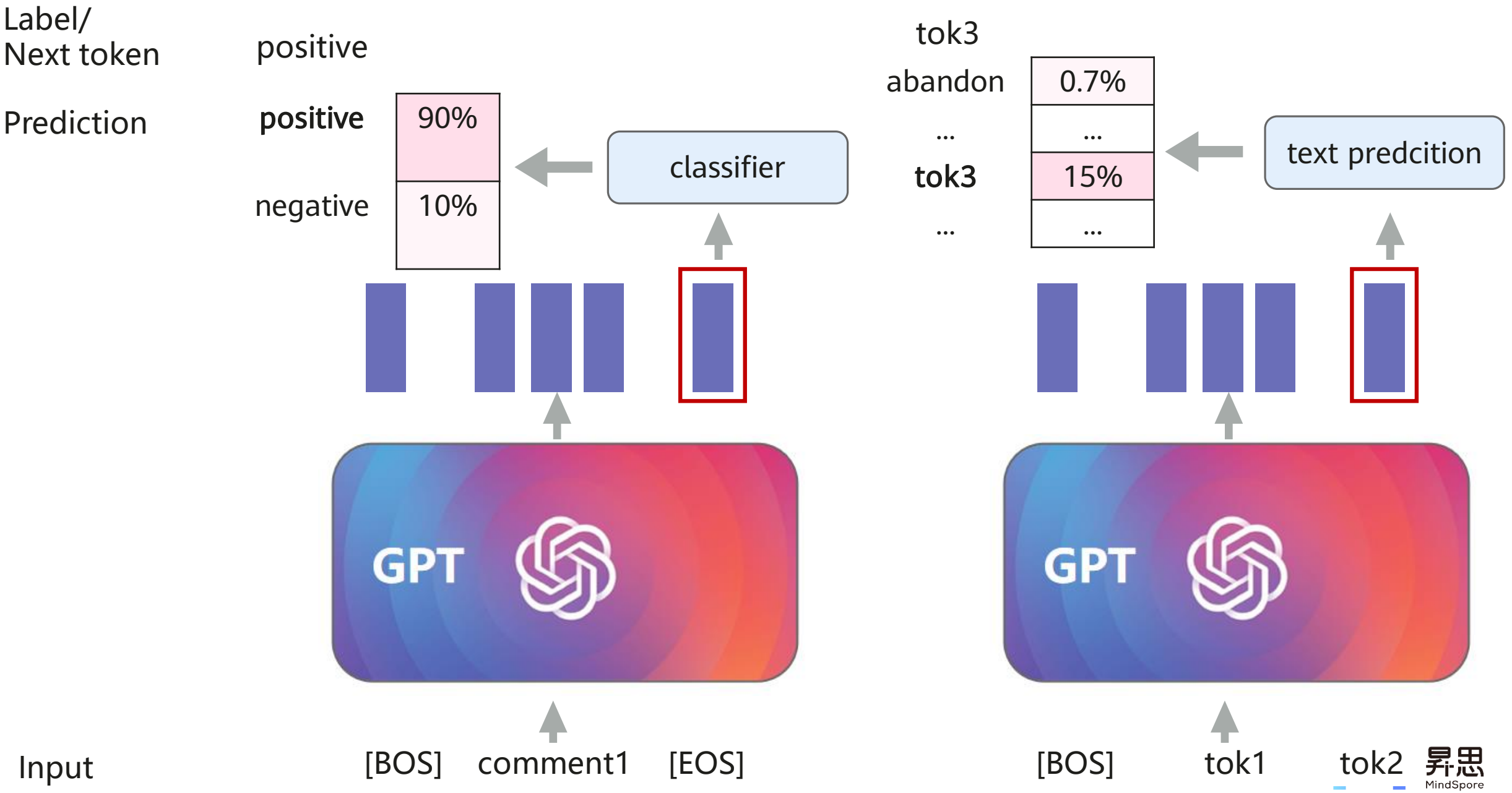
针对下游任务的微调中，往往需要考虑两方面：

1. 预测出的标签是否正确?

2. 预训练中language modeling评测标准的loss是否较低?

（论文中的解释是，为了加快模型的收敛，增强模型的泛化性，所以在下游任务微调中也需要考虑language modelling的损失，当然λ可以设为0）

$$L_3(C) = L_2(C) + \lambda * L_1(C)$$


Label/
Next token
Prediction
positive
positive
90%
negative
10%
classifier
GPT
Input
[BOS]
comment1
[EOS]
tok3
abandon
0.7%
...
...
tok3
15%
...
...
text predcition
GPT
[BOS]
tok1
tok2
昇思
MindSpore

Thank you

昇思 MindSpore