# 自然语言处理
# Natural Language Processing

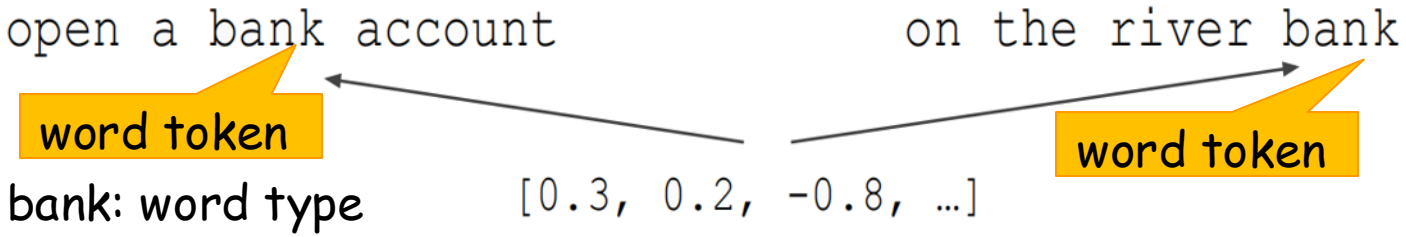
## Chapter 8
## 预训练模型

# 上下文词向量

- ✓ Tag LM
- ✓ ELMo

# 词向量回顾

- **静态词向量 (e.g. word2vec)的问题：**
- 对于一个词语(word type)得到的是固定的词向量表示，在不同的上下文中出现时(作为word token)都只能用该向量进行表示
  - 需要细粒度的词向量表示word token

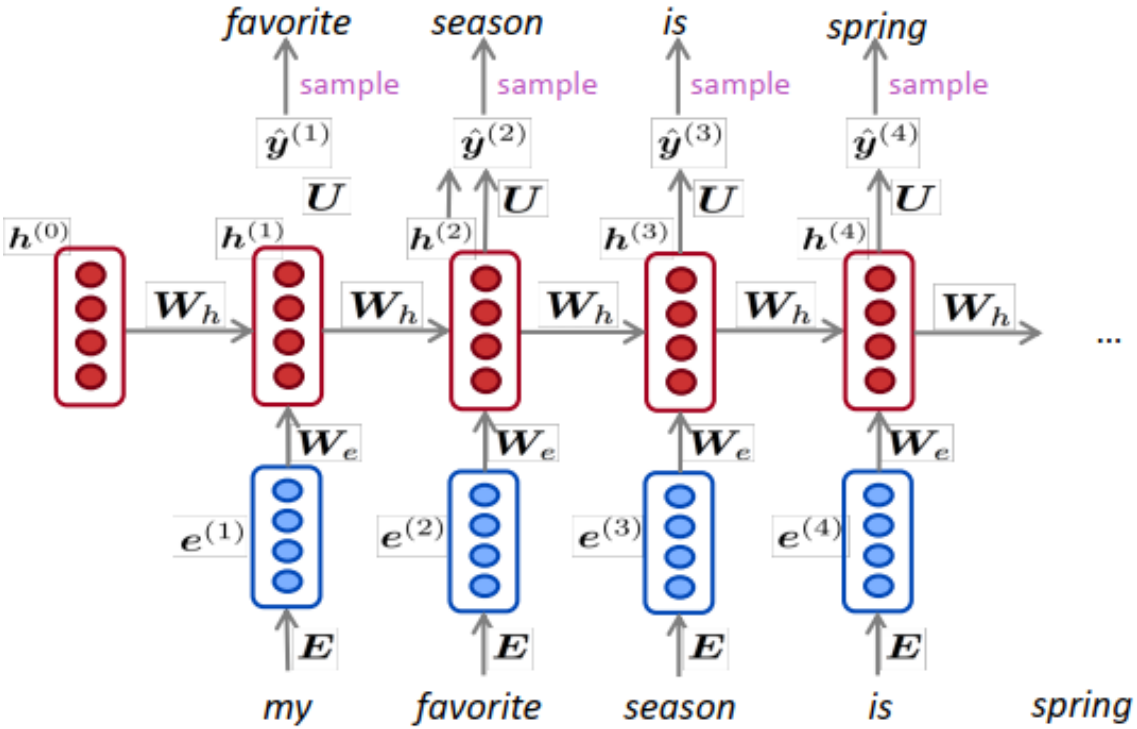



- 每个词语只有一个词向量表示，但词语有不同的方面，不同的含义，不同的句法行为(词性或者语法...).
- E.g. arrive & arrival, 表达意思相关，上下文不同。
- fair & good: 天气好，good & nice：人很好，fair：人很好？

- **解决方案：**考虑上下文的Contextual word embeddings!

# 上下文中的词向量

- 在神经语言模型中，输入词语初始向量，然后经过循环网络层，学习一个隐藏向量。
- 循环网络层根据隐藏向量预测下一个词.
- 当前步骤的隐藏向量可以看成是文本在该位置的上下文相关的表示 → 相对于静态向量表示



# 前驱工作：TagLM模型

- 2017，最初解决命名实体识别 (NER) 任务
- 驱动: 在具体任务上，一般标记数据量相对词向量训练而言较小，因此使用神经模型训练上下文相关的向量，效果受限

- 方案：采用半监督的方法，除了具体任务 (NER)的标注数据，另外使用一些无标记数据，帮助模型训练，即使用语言模型增强sequence tagger

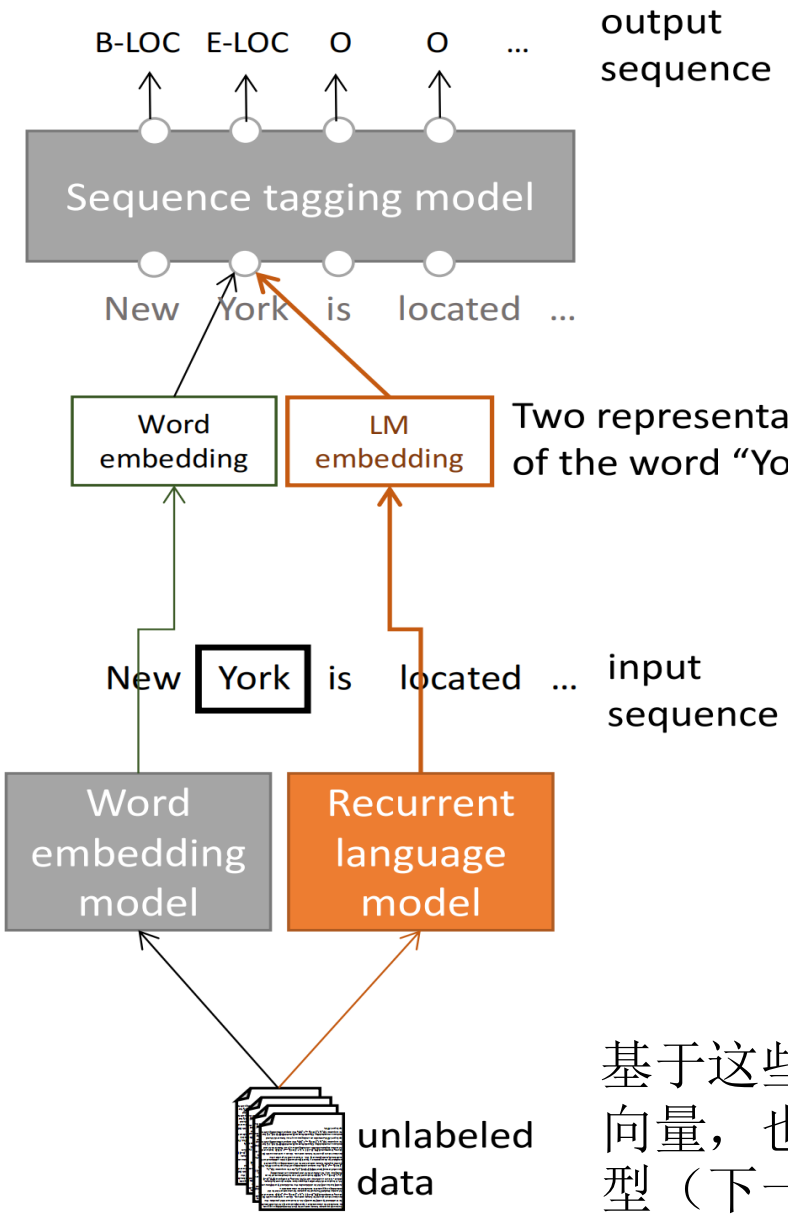
# Tag LM模型

LM: language model 语言模型

**Step 3:** Use both word embeddings and LM embeddings in the sequence tagging model.

**Step 2:** Prepare word embedding and LM embedding for each token in the input sequence.

**Step 1:** Pretrain word embeddings and language model.

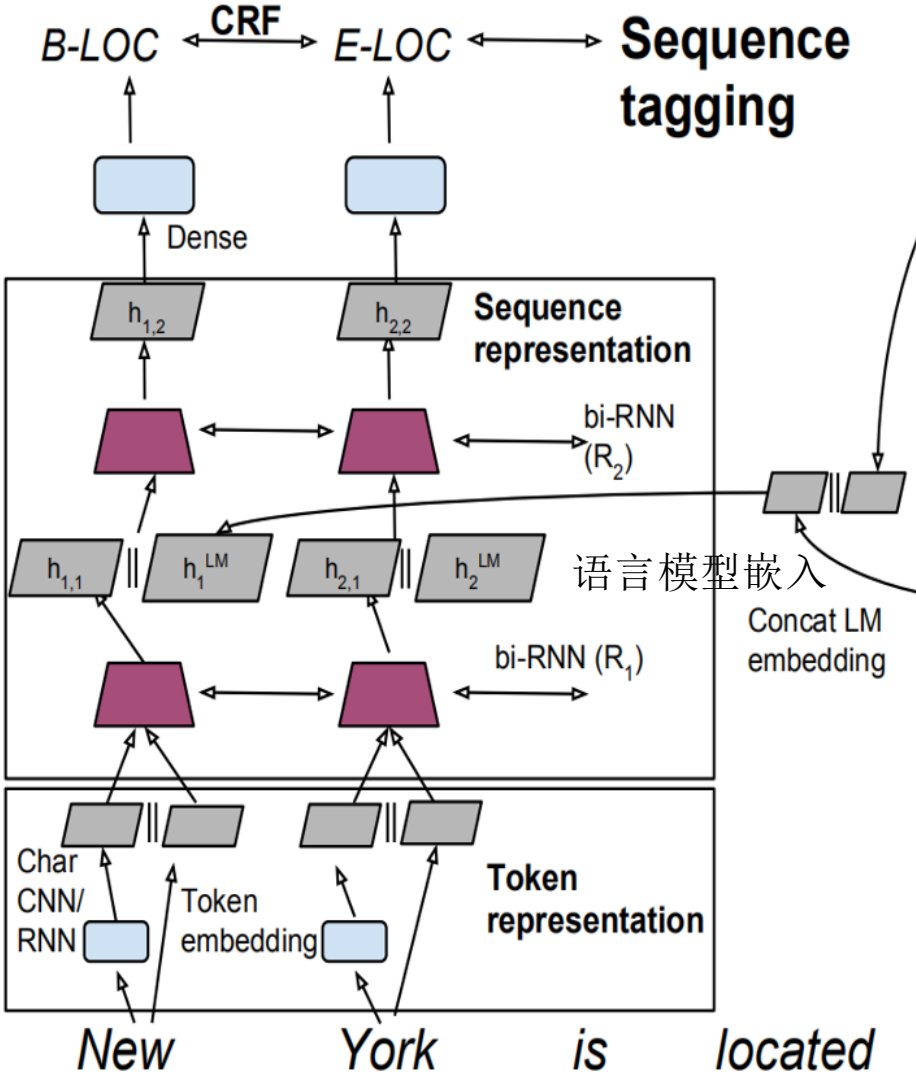



在上下文中训练得到的$h_i$

基于这些数据可以训练静态词向量，也可以训练神经语言模型（下一个词是什么）

# Tag LM模型

该预训练模型提供的表征提高了NER的性能




双向LSTM，独立训练

可以堆叠多层LSTM

监督训练

... 基于**大量无监督数据**训练。不与序列标注器一起训练，没有梯度

# ELMo模型

- Embeddings from Language Models (2018)
- 类似于TagLM的深度双向语言模型
- ELMo是一个突破性的系统，让上下文(contextual)词向量远远甩开传统的词向量，受到广泛的关注和使用
- 传统的词向量算法，如w2v 从一个较小的上下文窗口进行学习，而ELMo使用的是比较大的上下文范围，如一整个句子。

Peters et al. Deep contextualized word representations. NAACL. 2018. PDF

# ELMo模型

- ELMo希望提出一个易于使用的语言模型，同时希望这个语言模型的规模不要过大
- 模型细节:
  - 2层biLSTM，输入神经元=4096，输出神经元=512
    - 底层bilstm能捕获低级的语法词属性，对于词性标注、句法依赖等任务比较有用；高层bilstm能够表示高级的语义信息，适合用于语义角色标注，问答，推理等高级任务。
    - 可以继续堆叠
  - 使用字符CNN来构建词语初始表示，减少参数量
  - 残差连接，即h = f(x) + x
  - 参数绑定

# ELMo模型

- tagLM中，从预训练LM里输入到监督模型的，是LM堆栈的顶层，即最后一个bilstm层；在ELMo里面，使用了所有的层。
- 因此，对于第k个词语，在语言模型中的第j个bilstm层中对应的第k个位置的隐藏向量$h_{k,j}$做了一个加权和：

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

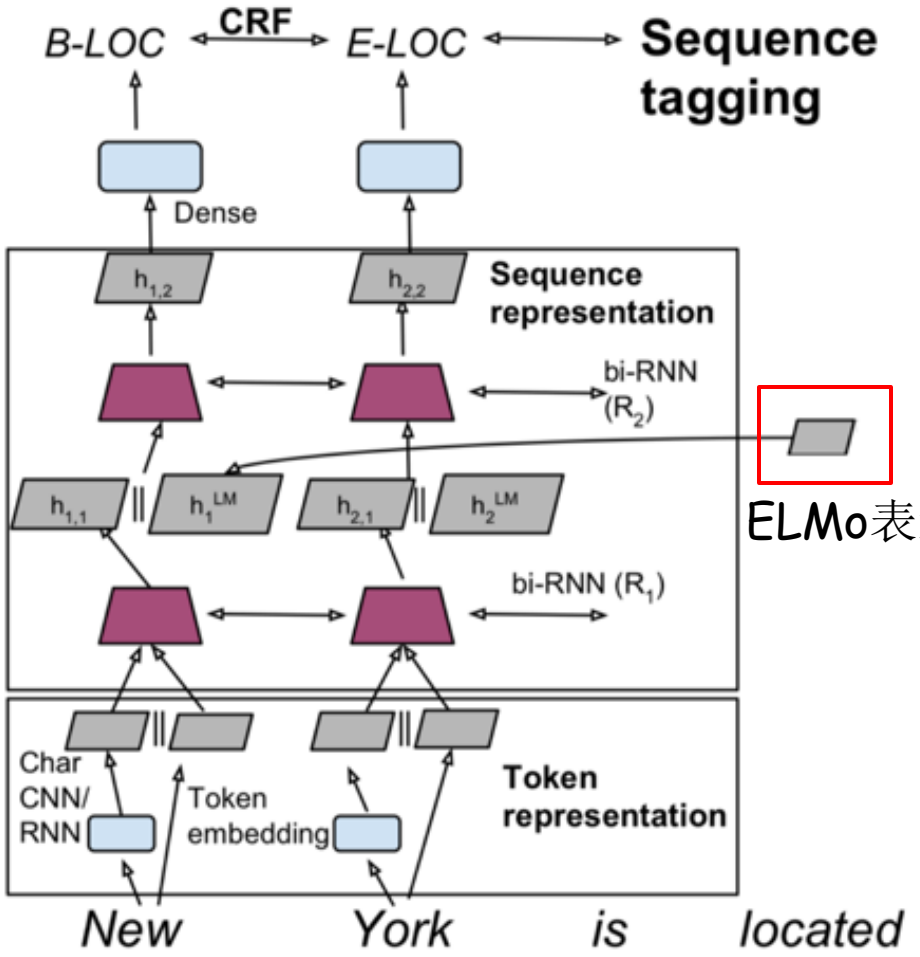
$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

- $\gamma^{task}$ 是全局比例因子，是elmo对于当前任务的一个有用性的度量
- $s_j^{task}$是任务的权重，不同的任务和不同的层之间的关联性不同，和下游任务一起训练的到

# ELMo模型

- 使用中，首先运行预训练好的语言模型，对文本里面每一个词语获得一个表示（$ELMo_k$）
- 进行具体的任务，称为下游任务 (downstream task)
  - ELMo部分的权重进行冻结 (freeze)，即不更新
  - ELMo部分得到的向量与具体任务相关的模型进行拼接
    - 具体细节取决于具体任务
      - ✓ 像tagLM一样，将$ELMo_k$拼接到中间层
      - ✓ 对输出做softmax之前再一次输入$ELMo_k$

# ELMo模型用于序列标注




$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}].$$

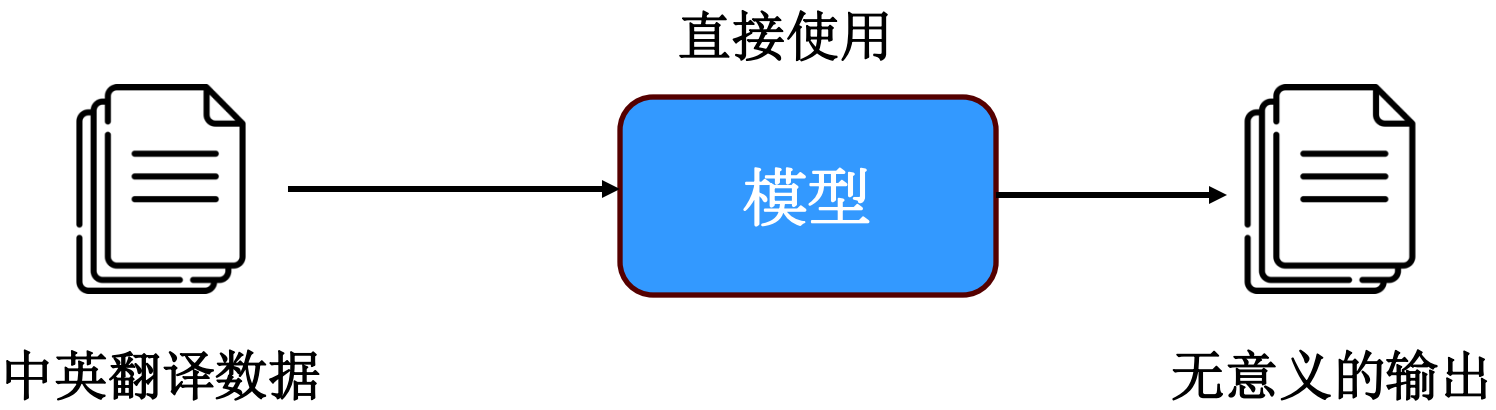
按照TagLM的使用方式，将ELMo用于NER

- ELMo也可以用于其他的nlp任务，e.g. 问答，自然语言推理，语义角色标注，情感分类
- 刷新了当时的SOTA (最好的结果), 而且仅需要把ELMo表示放到一个基础模型中，就可以比SOTA好
- 获得了NAACL最佳论文

# 预训练概念

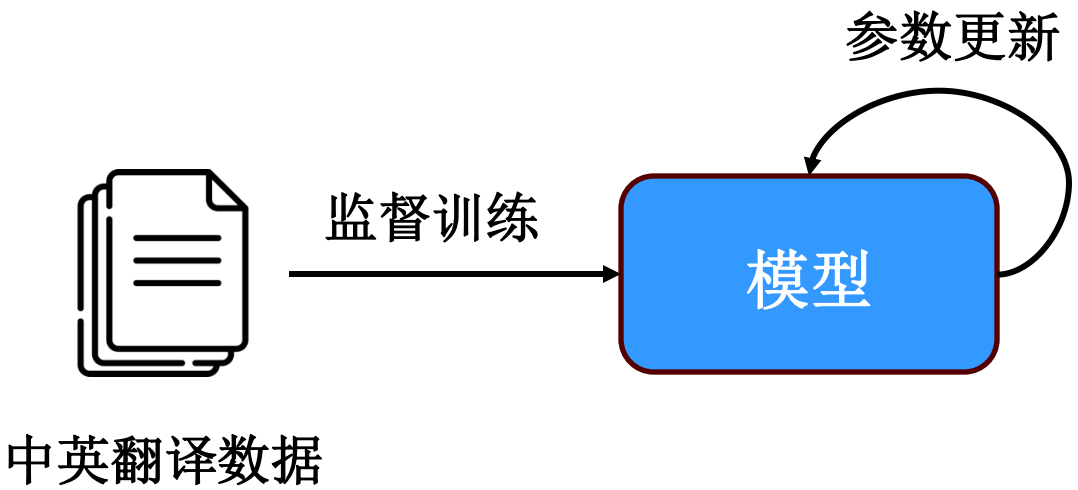
# 训练

- 监督训练 = 模型 + 任务数据 + 参数更新
  - 模型



# 训练

- 监督训练 = 模型 + 任务数据 + 参数更新
  - 模型
  - 数据，监督数据(supervised data)，有标签数据(labeled data)
  - 参数更新，根据真实标签与预测的差异




NLP发展：

第一范式：Fully Supervised Learning (non-neural network)

第二范式：Fully Supervised Learning (neural network)

# 训练

- **监督训练的不足：依赖于标记数据**
- 人工标注的成本高
- 过少的数据会导致过拟合（e.g. 新任务）
  - 死记硬背住有限的训练数据
  - 没有掌握真正解决问题的知识和能力
  - 无法泛化到其它数据、其他任务上

- 如何获得大量的训练数据，提高模型的泛化性能？

# 预训练

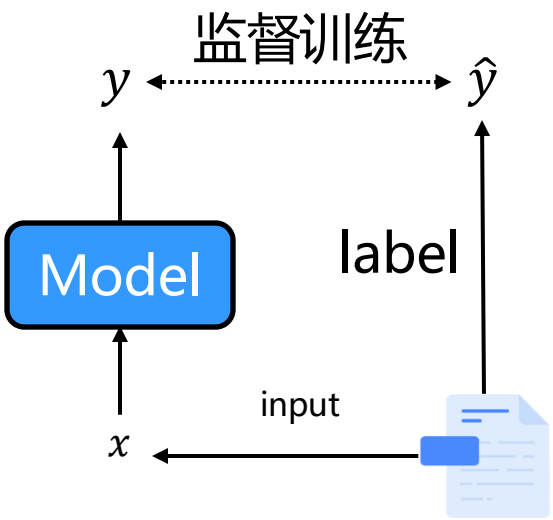
- 观察：人类学习新任务时，需要的数据较少
  - 人类已经具备大量的背景知识，例如常识和语言学知识
  - 这些背景知识与具体的任务无关，是通用的，有助于新任务的解决
- 因此，希望让模型在具体任务之前，经过预训练（预先的训练），拥有背景知识

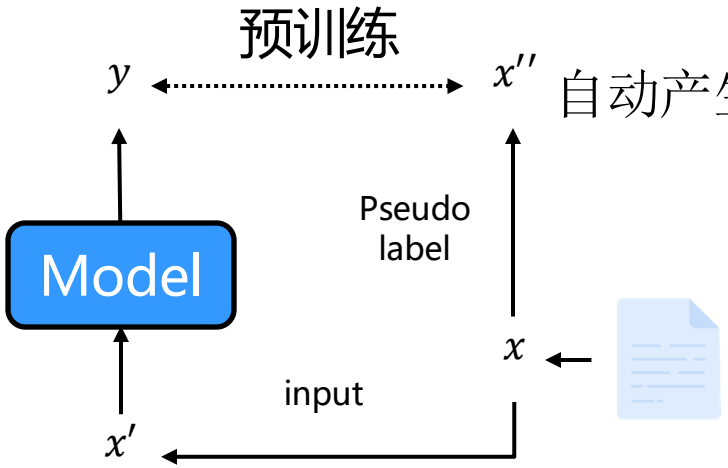
# 预训练

- 收集大量数据
  - 考虑到标记数据的昂贵，采用无标记数据，如百科数据、新闻文本、推特微博等
  - 以推特数据为例，通过爬虫爬取，只有文本，没有诸如“主题”、“语言风格”、“情绪”、“对应外语翻译”、“政治立场”等标签

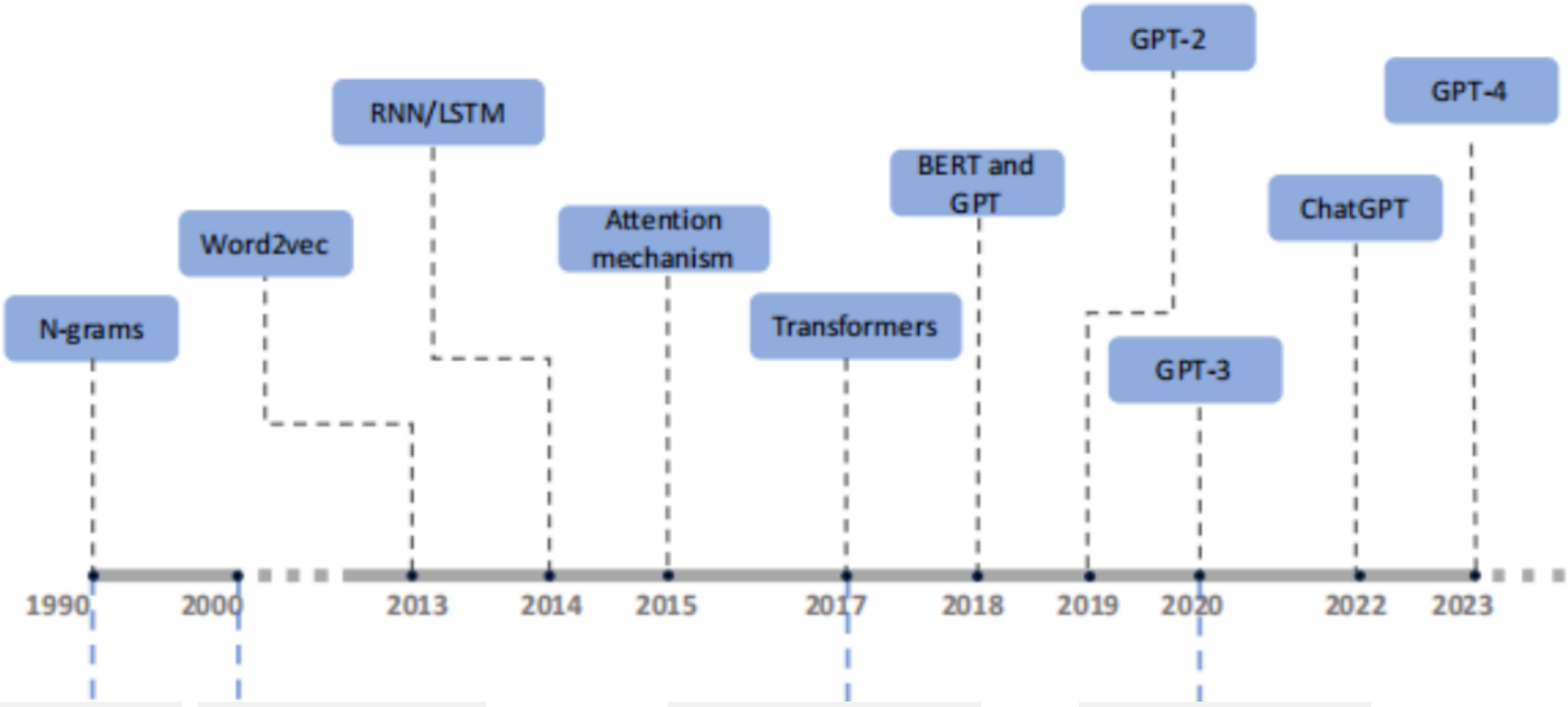
# 预训练

- 设计预训练任务
  - 在大量数据上训练模型，以拥有与具体任务无关的知识
  - 经常使用**自监督学习**，通过文本中的关系或知识，将无标记数据自动打上”标签”，如词语的前/后一个词。然后进行训练优化。
  - 当有标记数据充足时，也可以直接使用监督学习。



# 插播：语言模型的发展




统计语言模型
Statistical language models

神经语言模型
Neural language models

预训练语言模型
Pre-trained language models

大语言模型
Large language models

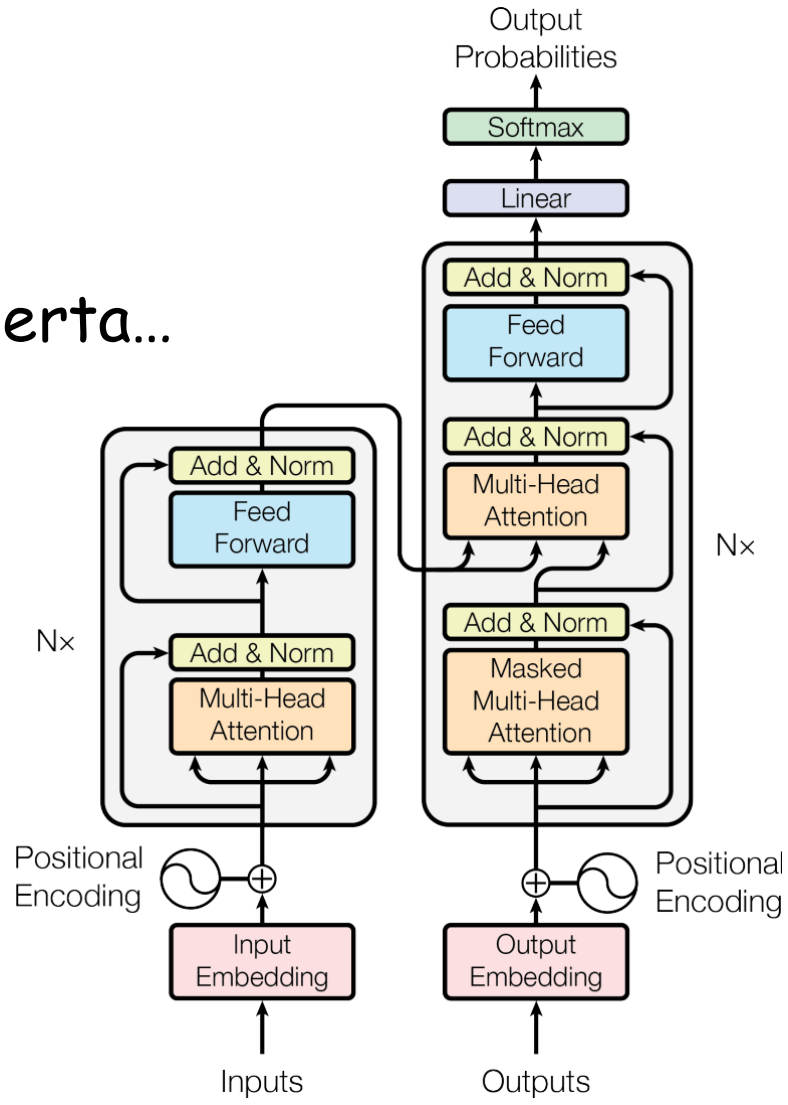
# 主要模型与技术

- ✓ 基于Transformer

# 基于Transformer的预训练模型

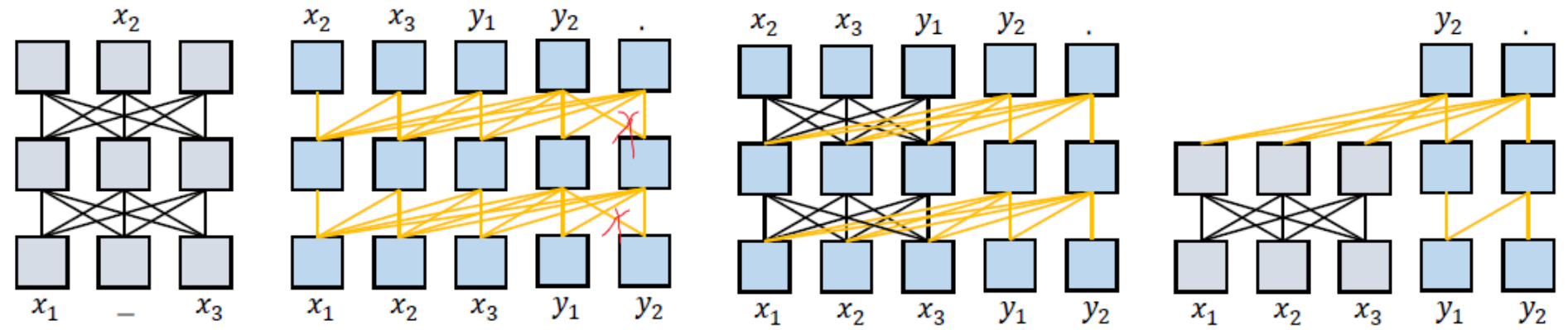
较为简单的分类体系：
根据基于Transformer的结构

- 基于编码器：BERT, AlBert, Roberta…
- 基于解码器：GPT family…
- 基于编解码器：BART, T5, …



# 基于Transformer的预训练模型

对预训练模型中的多种架构进行比对：



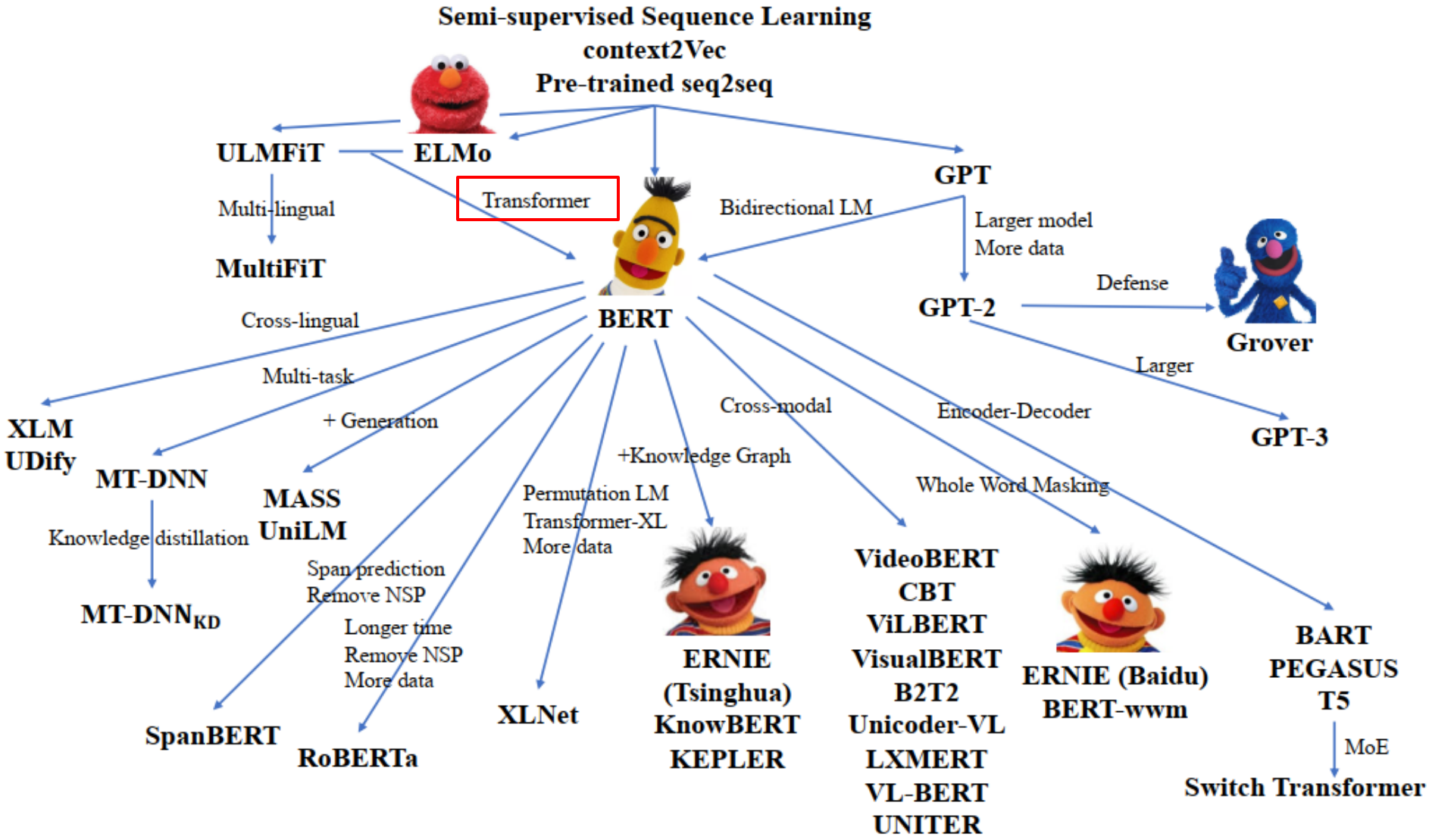

(a) 掩码语言模型　(b) 因果语言模型　(c) 前缀语言模型　(d) 编码器-解码器语言模型

※ masked language model
只有编码器
能看到全部输入

※ Casual language model
只有解码器
只能看到已经产生的部分序列

※ Prefix language model
一部分如 Encoder
能看到全体信息，
一部分如 Decoder
只能看到过去信息

※ 编码器可以看到全部输入，解码器只能看到已经产生的部分序列




Pre-Trained Models: Past, Present and Future, 2021

2022 version

# Bigger Model

2022 PaLM 540B

Switch Transformer 2021 1.6T

MT-NLG 530B 2021

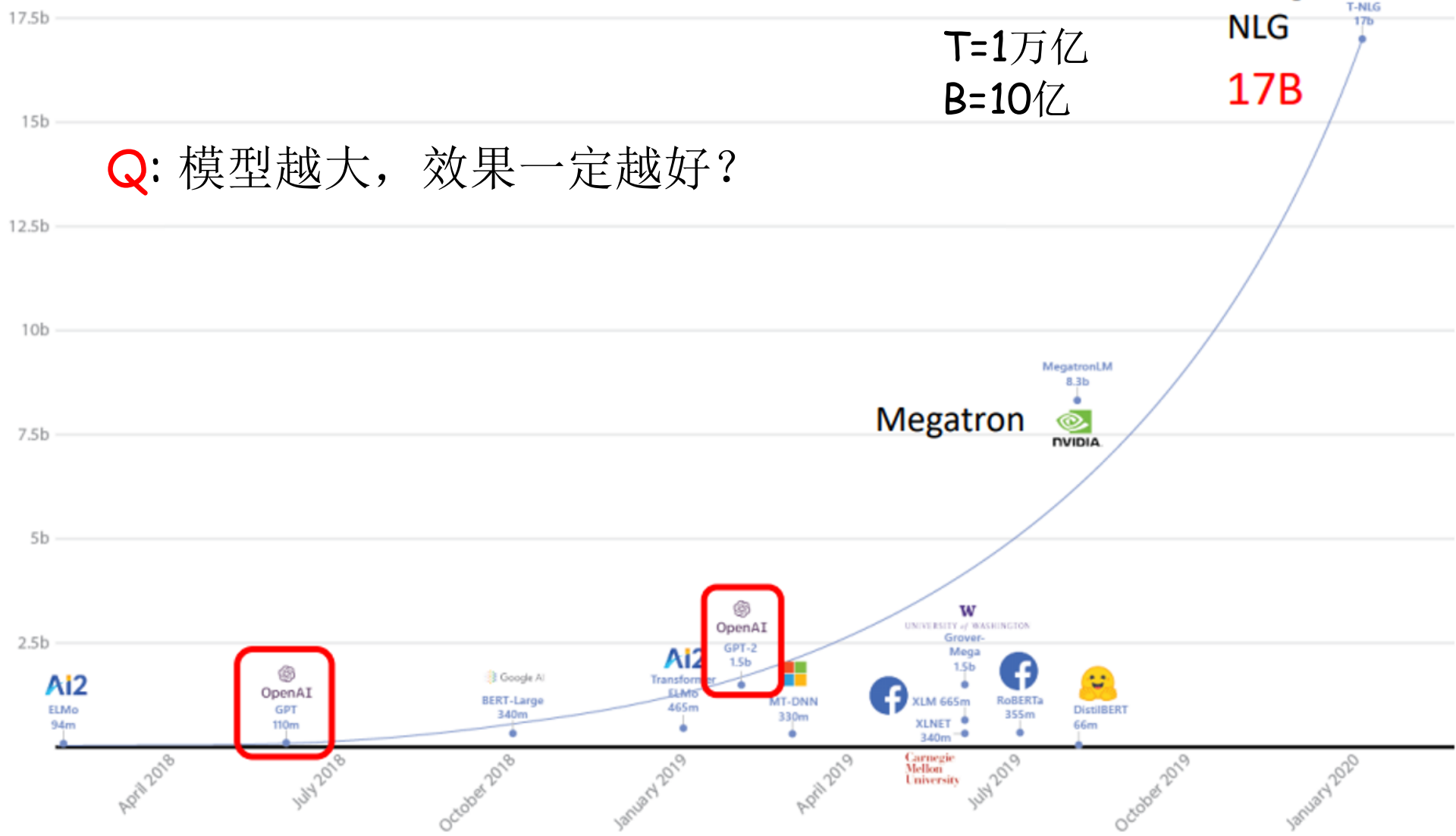
GPT3 175B

Turing NLG 17B

T=1万亿

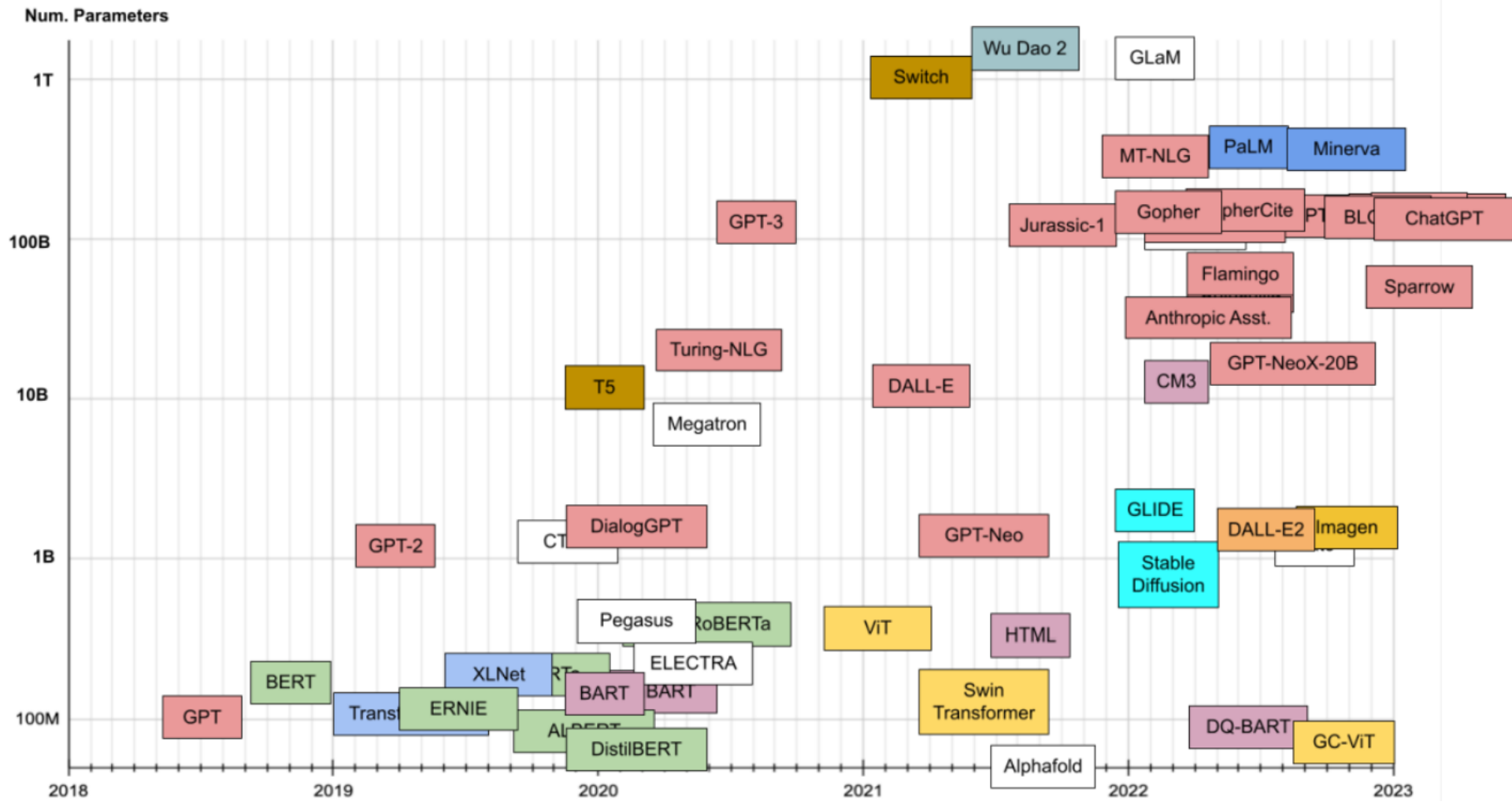
B=10亿

Q: 模型越大，效果一定越好？



# 2023 version

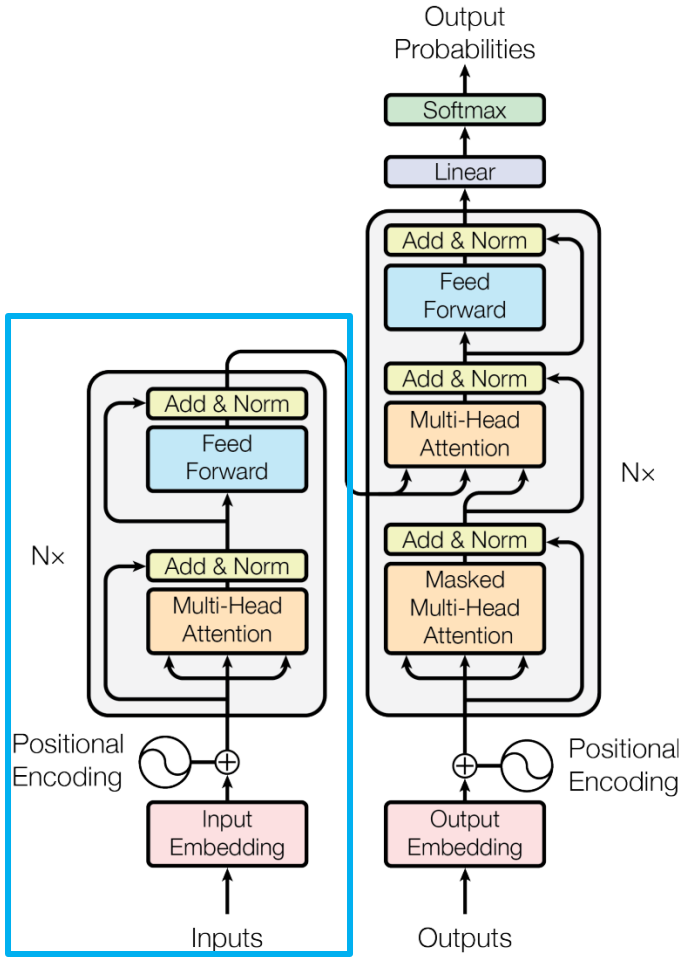
后续：更全面or专注的功能，更多的模态。。。



# ❖ BERT

- 全称 = Bidirectional Encoder Representations from Transformers

- BERT= Transformer编码器端
- 2018年，谷歌提出
- 预训练任务: 掩码语言模型 (Masked Language Modeling, MLM), 下一句预测 (Next Sentence Prediction, NSP)




BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, PDF

# 掩码语言模型

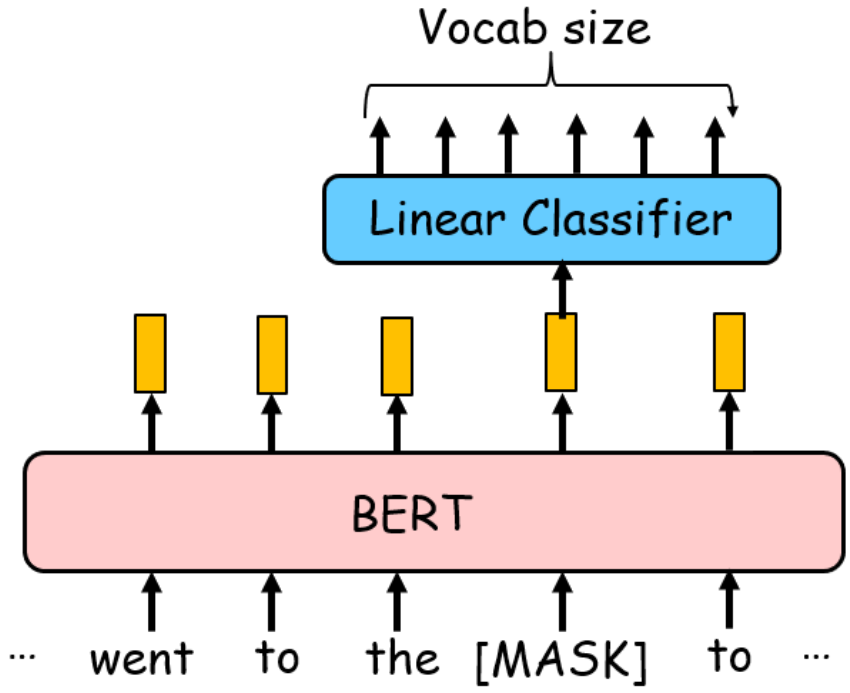
- 目标: 遮蔽 (Mask) 输入文本中 k% 的词语，让模型根据上下文预测被遮蔽的词 [MASK]是什么

```
                                store                    gallon
                                  ↑                        ↑
the man went to the [MASK] to buy a [MASK] of milk
```

取 $k = 15$

- k太小: 数据量少，训练昂贵
- k太大: 遮蔽太多词语，可以用来提供信息的上下文就会很少
- 实验证明此时效果好

类似于CBOW



# 掩码语言模型

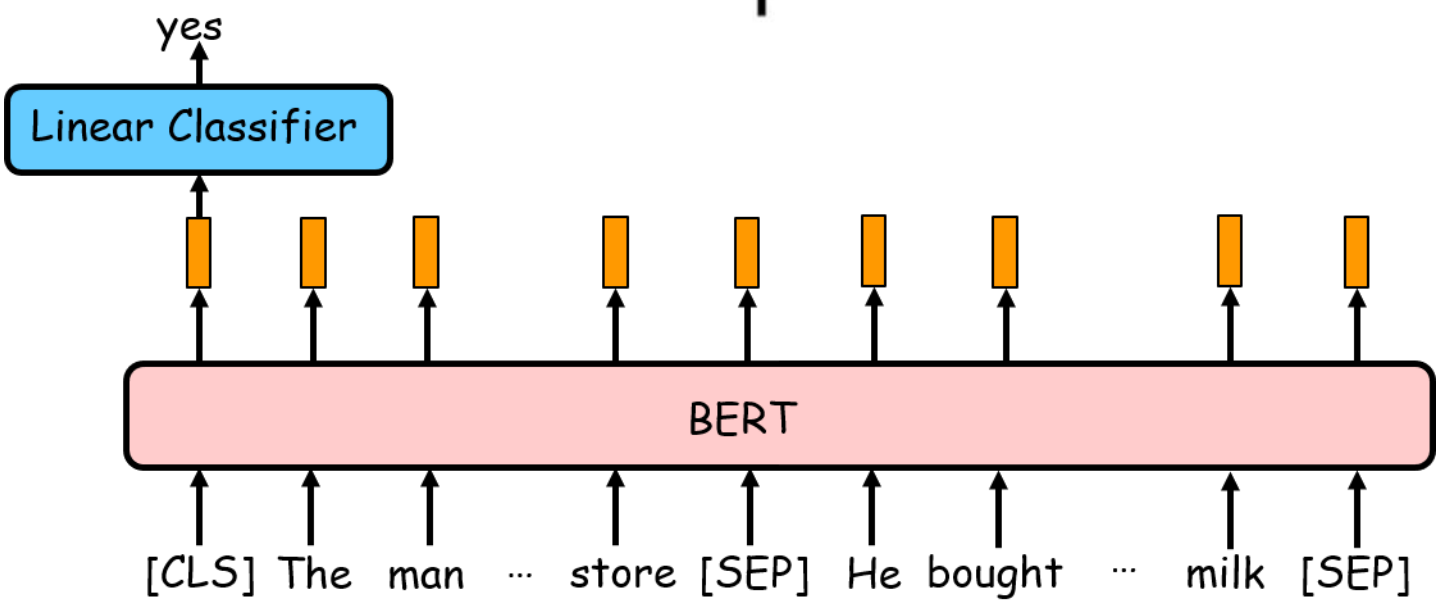
- **问题**: 预训练阶段很多词语替换为[MASK]，但是后续进行具体任务的微调 (fine-tune)时，没有[MASK]标记，存在不一致
- 解决方案: 在MLM中，k%的选中词不是在所有时候都遮蔽。
  - 80% 的情况下, 替换为 [MASK]
    - went to the store → went to the [MASK]
  - 10%的情况下, 替换为 随机词语
    - went to the store → went to the running
  - 10%的情况下, 保持不变
    - went to the store → went to the store

# 下一句预测

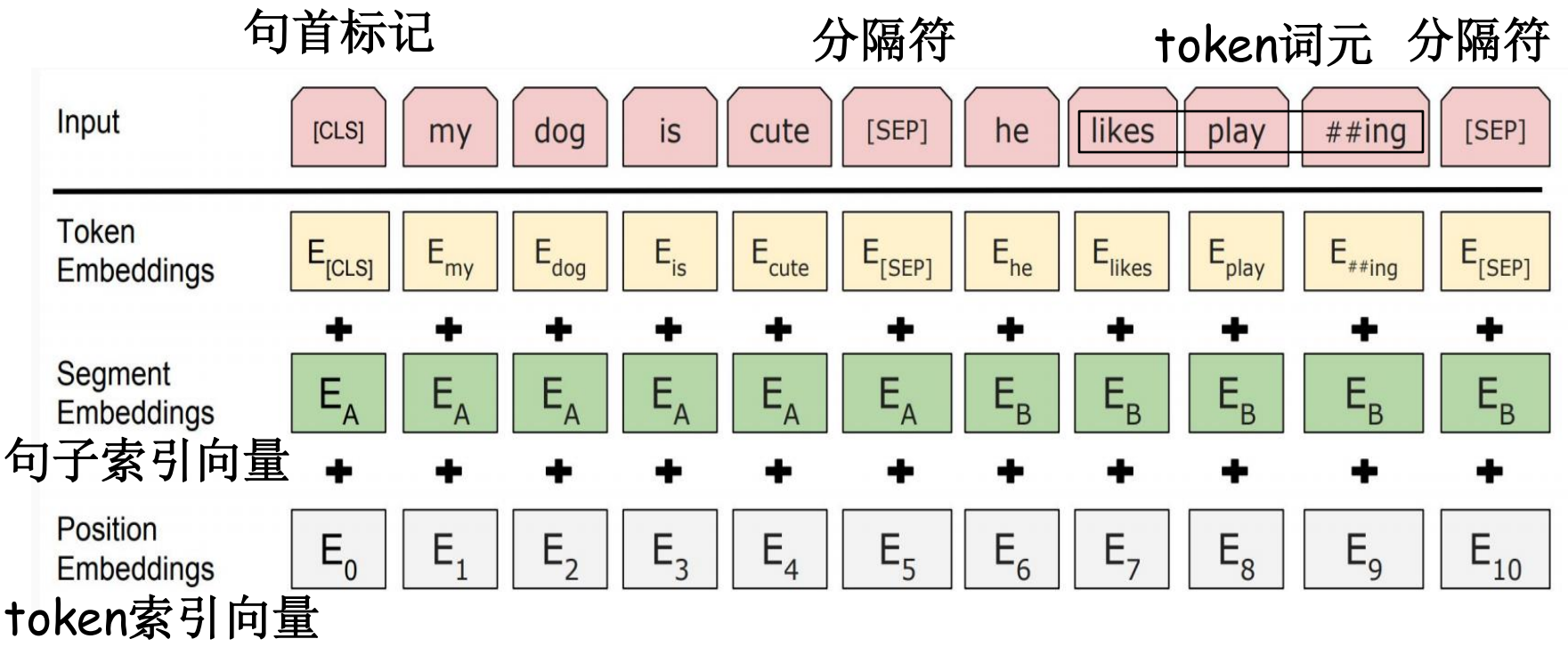
- **目标：**判断句子A和句子B是否存在“A是B的下一句”的顺序关系（二分类任务），希望模型学习到句子之间的上下文关系。

**Sentence A** = The man went to the store.
**Sentence B** = He bought a gallon of milk.
**Label** = IsNextSentence

**Sentence A** = The man went to the store.
**Sentence B** = Penguins are flightless.
**Label** = NotNextSentence



# 输入表示




- BERT中的token是词元 (word piece)，英文token词表规模约3万。中文token是字，词表规模约4千。
- token embedding是每一个token的向量，如One-hot
- token 初始表示是3个向量之和 → 编码器

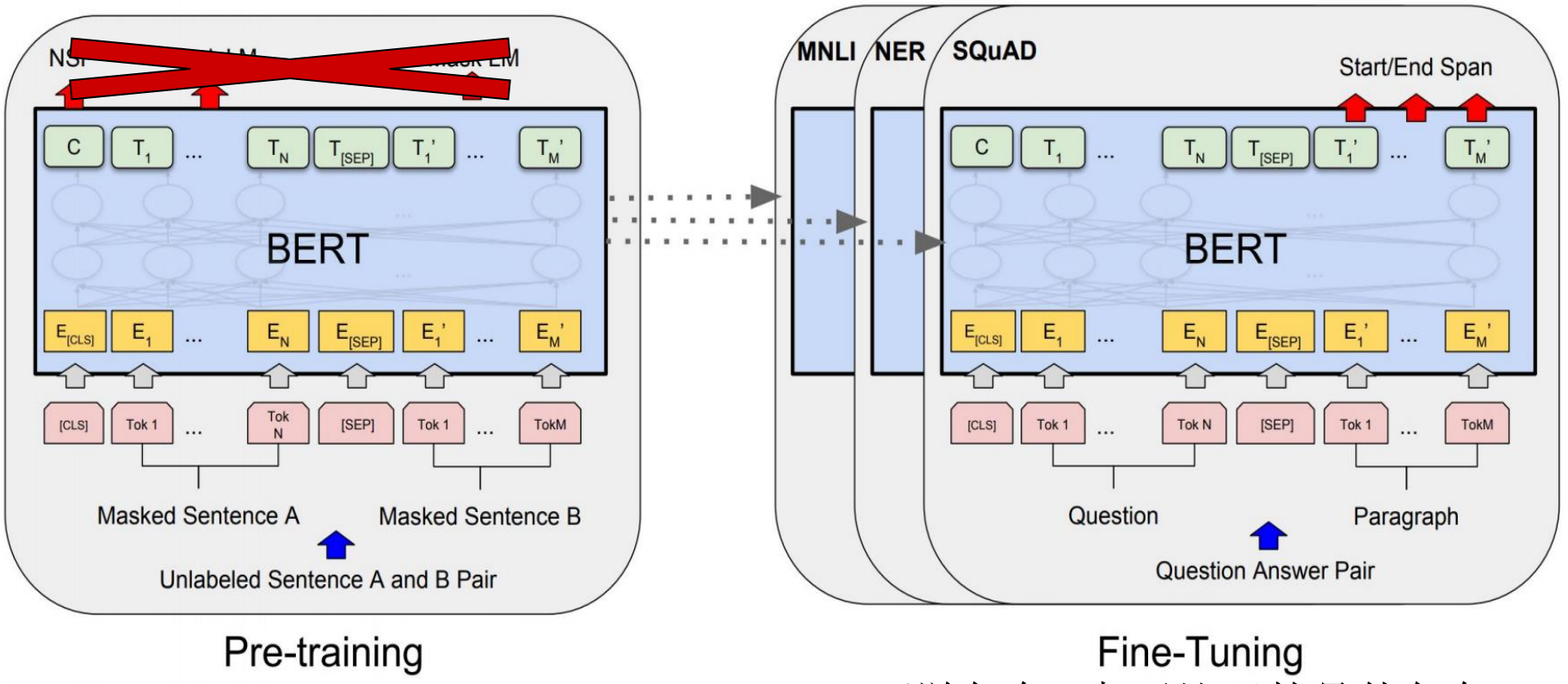
# 模型细节

- 预训练数据: 维基百科 (2.5B words)+BookCorpus (800M words)
- 优化器: AdamW, 1e-4 learning rate, linear decay

- 2个版本 (谷歌的初始官方版本):
  - ✓ BERT-Base: 12-layer, 768-hidden, 12-head
  - ✓ BERT-Large: 24-layer, 1024-hidden, 16-head (参数量340M)
- 设备：4x4 或 8x8 TPU (4天)

# 微调

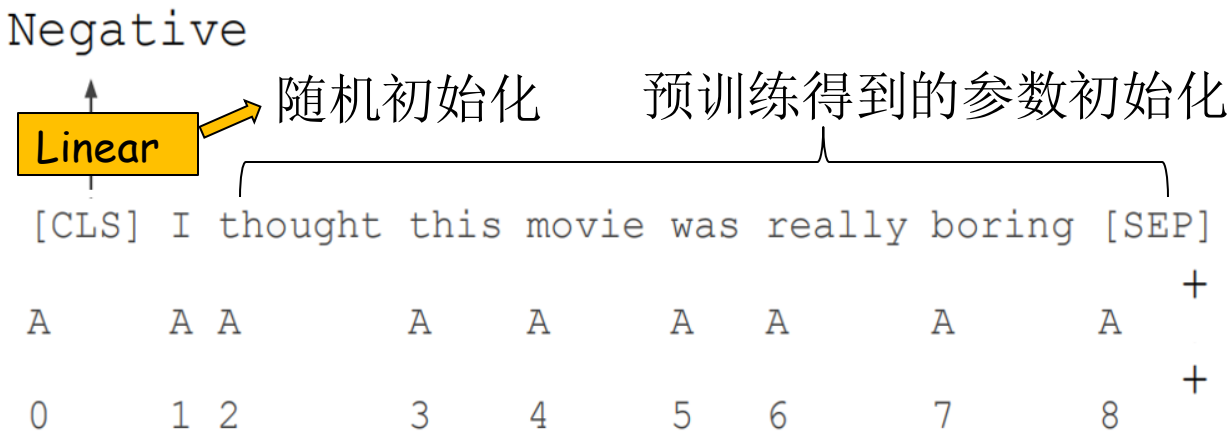
- 解决具体任务时，删除BERT原先的MLM和NSP预测层，在模型后面加上具体任务的预测层，在该任务的标记数据上进行训练 → 微调 (fine-tuning)




下游任务：真正处理的具体任务，标记数据量较少

# 微调

- **情感分类**。使用[CLS]向量作为文本整体的表示，以此预测类别。



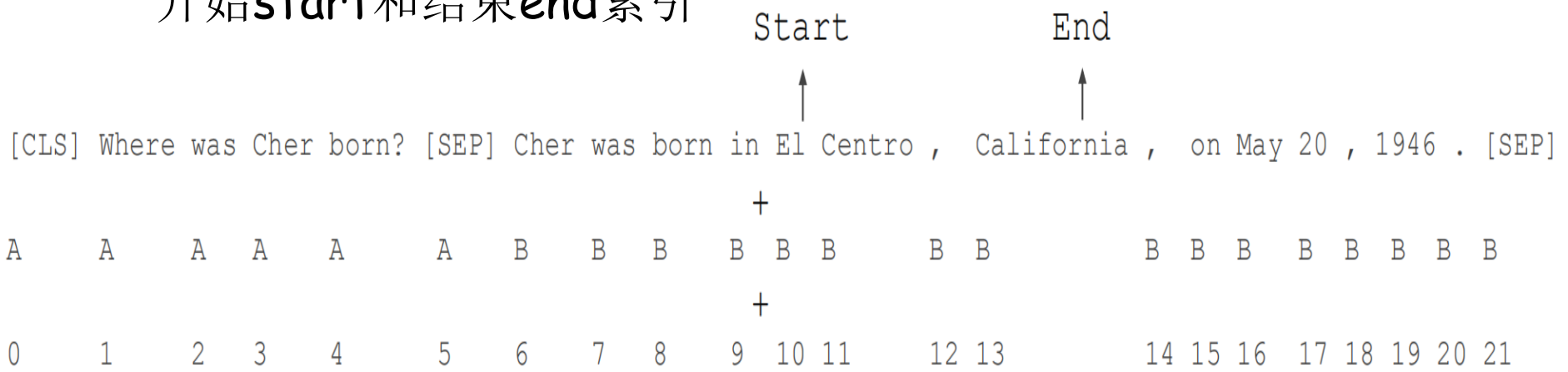

BERT+分类层：完整的情感分类器

BERT部分参数可以固定，也可以一起更新

选择[CLS]的一种解释：BERT使用的是self-attention，虽然理论上每一个位置i的词语都会考虑其他位置的词，但还是会偏向当前位置i，所以为了公平，就取这个人为添加的向量作为代表。实际上CLS其实就是classification的缩写，一般使用[CLS]向量专门用来代表整个句子的embedding。

# 微调

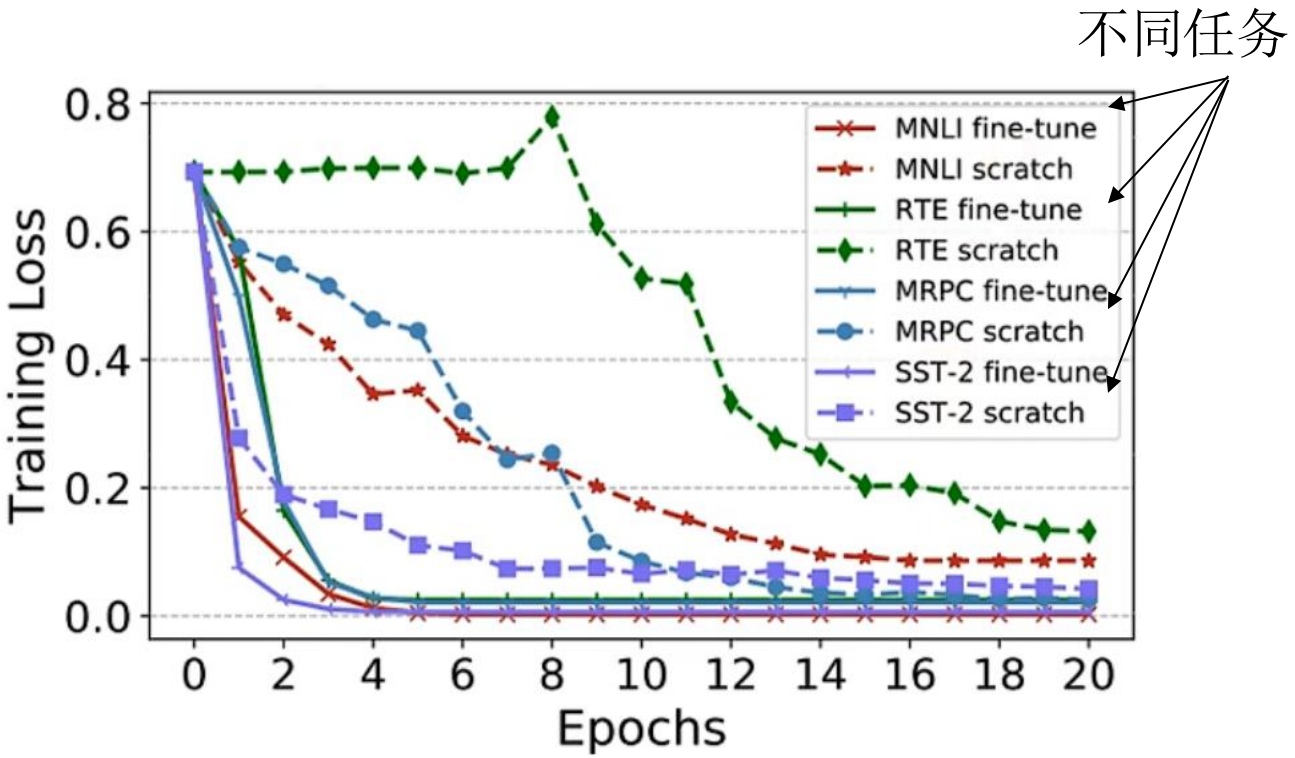
- SQuAD 问答: 答案文本是输入文本中的一部分，可以预测开始start和结束end索引




在BERT后面加预测层，对于start和end分别设置一个向量，与B上每一个位置的表示计算注意力权重，预测start概率最大、end概率最大的词。答案文本即为start和end之间的文本。

# 预训练带来的提升

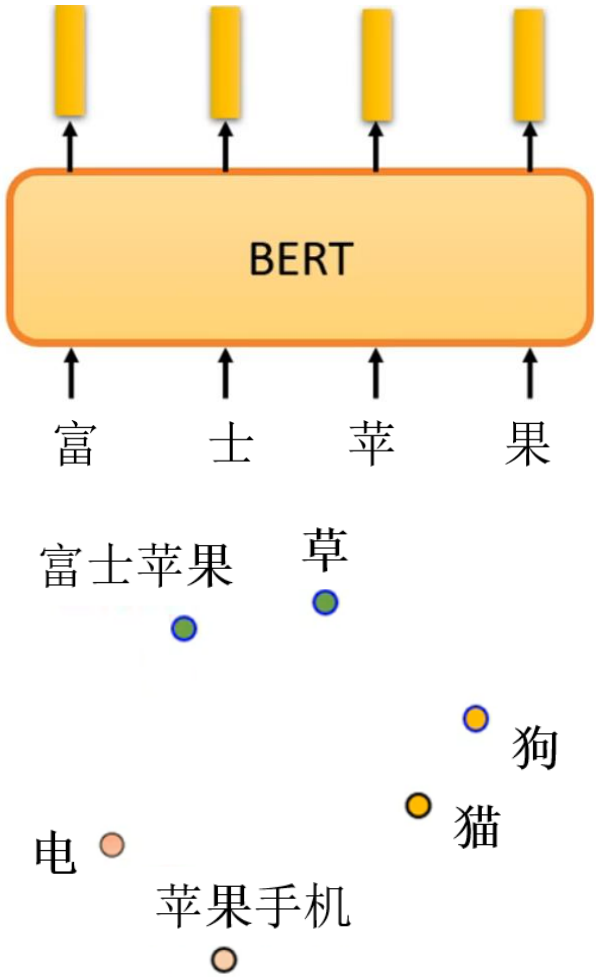
- 预训练 VS 随机初始化参数

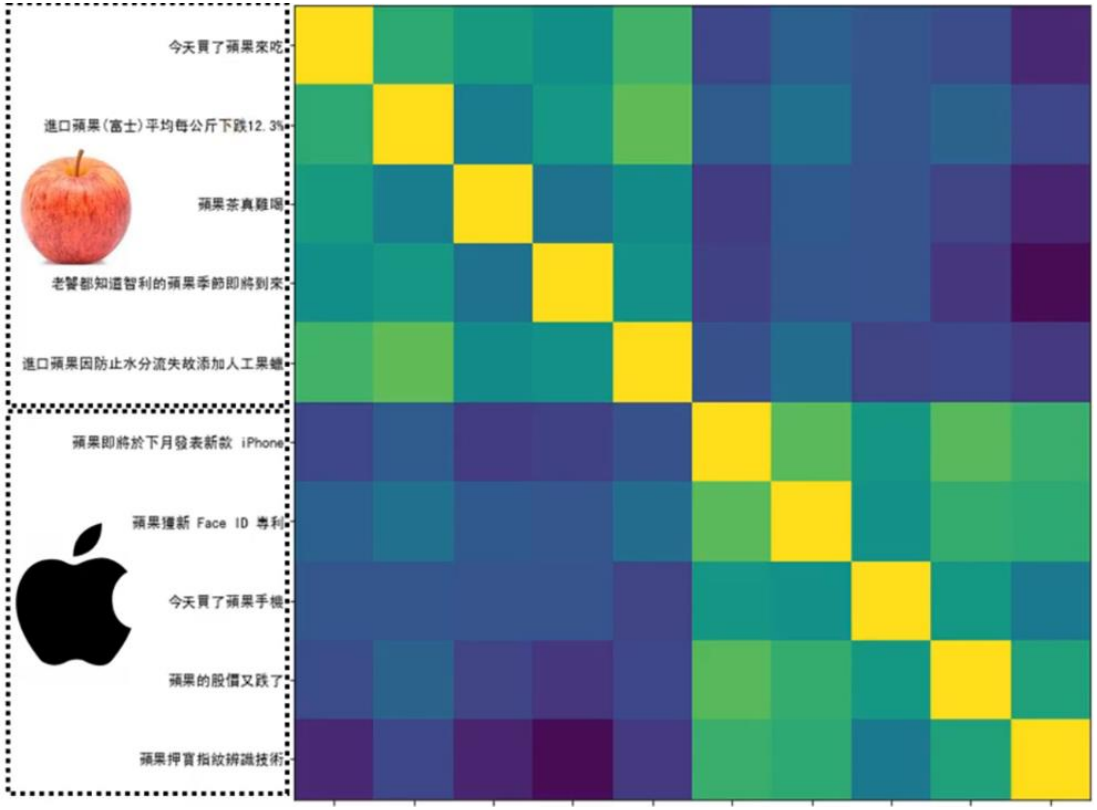



https://arxiv.org/pdf/1908.05620.pdf

# Bert嵌入

- BERT训练得到的是上下文相关的动态token嵌入
- 富士苹果 vs 苹果手机




How to get such embedding?
One easy way: bert-as-service

# 开源模型

- TensorFlow: https://github.com/google-research/bert
- PyTorch: https://github.com/huggingface/pytorch-pretrained-BERT

预训练模型使用注意：区分语言版本, e.g. 中文版、多语言版

NLP发展：

第三范式：Pre-train, fine-tune

# ❖ RoBERTa模型

- 2019，参数量和Bert齐平
- 基于Bert 进行了如下修改:
  - 更大的batch size ，训练数据更多，训练时间更长
  - 动态掩码机制 (未用于中文版本)。将数据输入时才masking，并将训练数据复制多份，一条数据可以进行不同的masking，划分进不同的epoch，充分利用数据。
  - 删除下一句预测(NSP)任务，认为太简单，对模型序列无益
  - 文本编码采用一种在word-level和character-level之间的表示

| | MNLI | QNLI | QQP | RTE | SST | MRPC | CoLA | STS | WNLI | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| *Single-task single models on dev* | | | | | | | | | | |
| $\text{BERT}_{\text{LARGE}}$ | 86.6/- | 92.3 | 91.3 | 70.4 | 93.2 | 88.0 | 60.6 | 90.0 | - | - |
| $\text{XLNet}_{\text{LARGE}}$ | 89.8/- | 93.9 | 91.8 | 83.8 | 95.6 | 89.2 | 63.6 | 91.8 | - | - |
| RoBERTa | **90.2/90.2** | **94.7** | **92.2** | **86.6** | **96.4** | **90.9** | **68.0** | **92.4** | **91.3** | - |

Yinhan Liu et al. Roberta: A robustly optimized BERT pretraining approach, 2019. PDF

# ❖ ALBERT模型

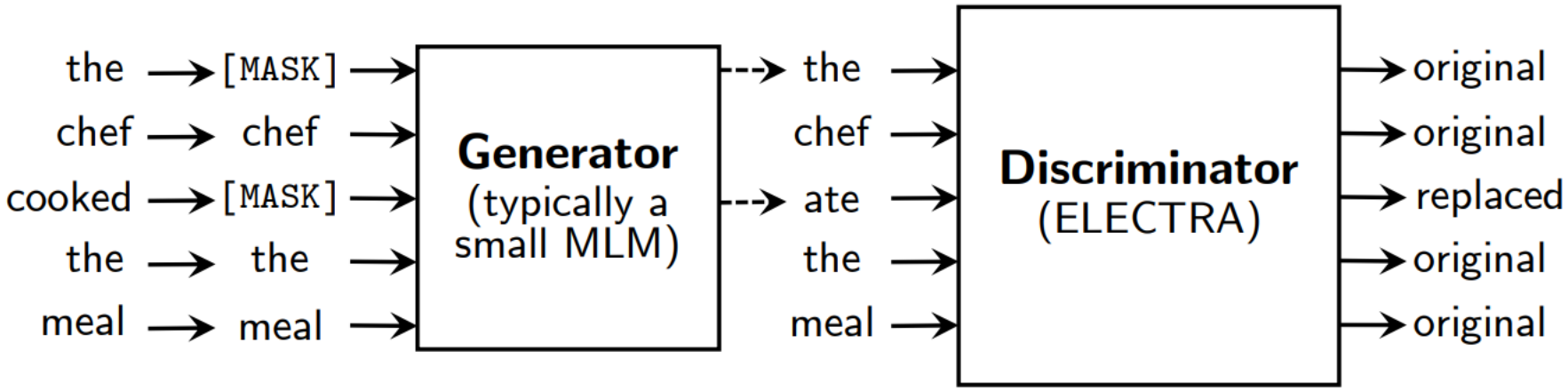
- ALBERT = A Lite BERT，轻量版BERT，2019年提出
- 对词嵌入参数进行因式分解，降低词嵌入的维度
- 跨层参数共享，直接减少参数量
- 句子顺序预测 (Sentence-order prediction, SOP) 取代NSP，增加任务难度

| Models | MNLI | QNLI | QQP | RTE | SST | MRPC | CoLA | STS | WNLI | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| *Single-task single models on dev* | | | | | | | | | | |
| BERT-large | 86.6 | 92.3 | 91.3 | 70.4 | 93.2 | 88.0 | 60.6 | 90.0 | - | - |
| XLNet-large | 89.8 | 93.9 | 91.8 | 83.8 | 95.6 | 89.2 | 63.6 | 91.8 | - | - |
| RoBERTa-large | 90.2 | 94.7 | **92.2** | 86.6 | 96.4 | **90.9** | 68.0 | 92.4 | - | - |
| ALBERT (1M) | 90.4 | 95.2 | 92.0 | 88.1 | 96.8 | 90.2 | 68.7 | 92.7 | - | - |
| ALBERT (1.5M) | **90.8** | **95.3** | **92.2** | **89.2** | **96.9** | **90.9** | **71.4** | **93.0** | - | - |

Lan et al. ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS, PDF

# ❖ ELECTRA模型

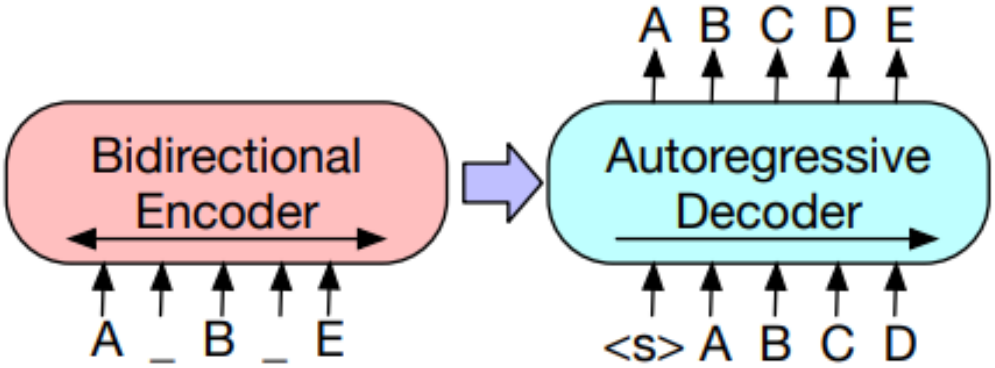
- Objective: replaced token detection (RTD)：判断token是替换过的还是原来的。模型会从所有输入标记中学习，而不仅仅是mask部分，使计算更有效率
- 生成器和判别器共同训练，但判别器的梯度不会回流到生成器。




Clark et al. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. 2019. PDF

# ❖ BART

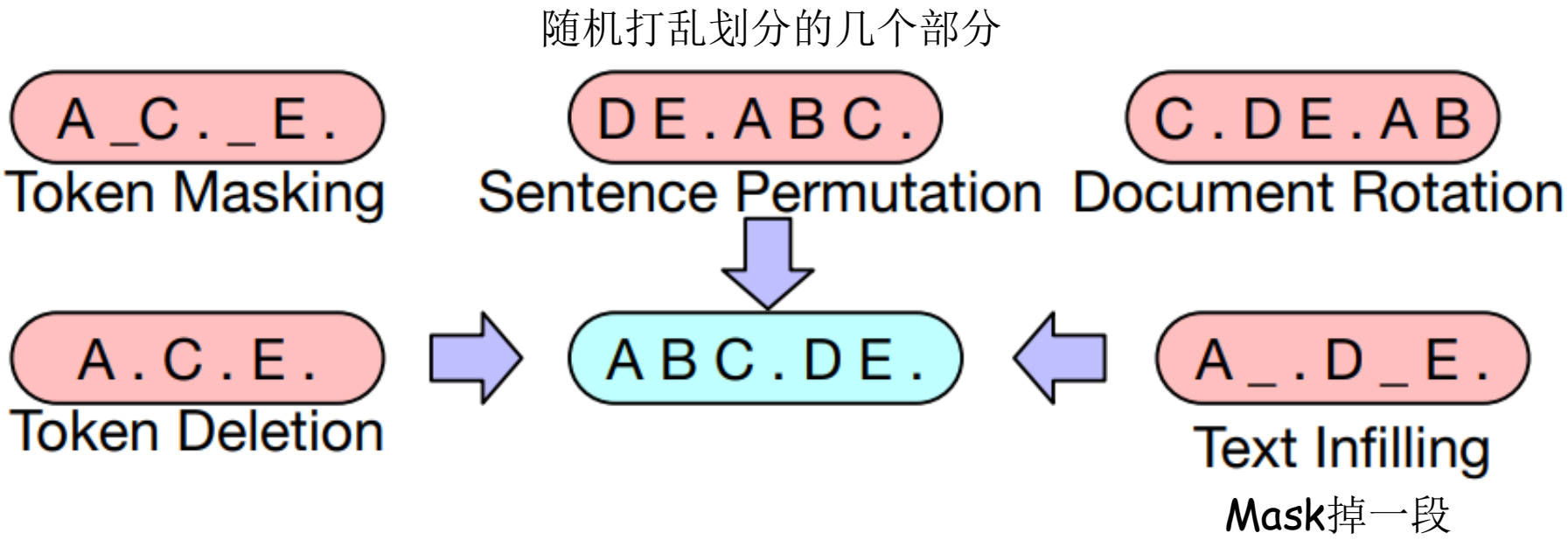
- Bidirectional and Auto-Regressive Transformers
- 2019.10 由 Meta 提出
- 具备完整的编码器和解码器，比BERT更适合做生成任务，比纯decoder多了双向上下文语境信息




Lewis et al. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. 2020. PDF

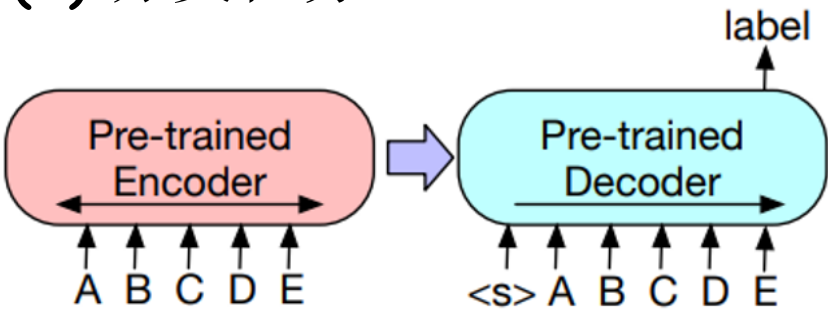
# BART：预训练

- 任务：将文档进行破坏，通过encoder、decoder进行还原，最小化重构损失 (reconstruction loss)
- BART采用多种文档破坏方式 (多种token noise)



# BART：微调

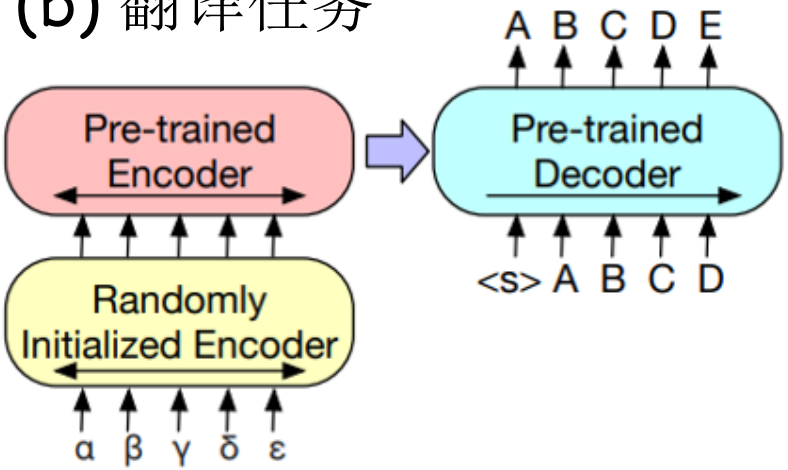
(a) 分类任务




多对一分类：将文本同时输入encoder端和decoder端，取decoder最后一个token对应的隐向量预测label

SQuAD 问答：取decoder每一个token对应的隐向量，分别做类别预测

(b) 翻译任务



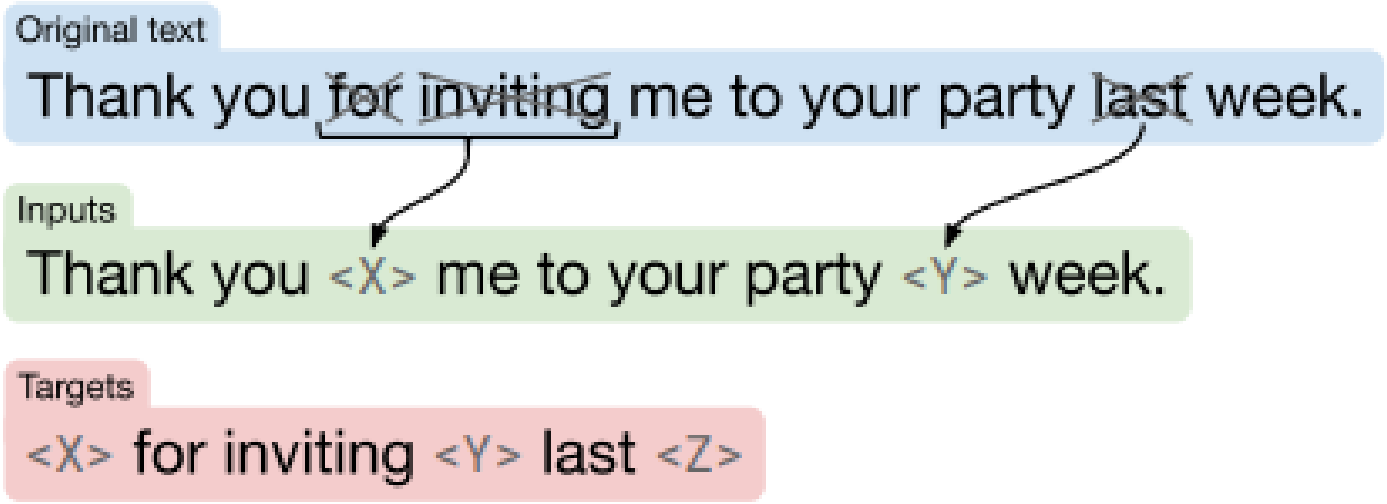

将BART encoder端的embedding层替换成一个随机初始的encoder

Step1：只更新 随机初始的encoder + BART 位置编码 + BART encoder第一层的self-attention

Step2：更新全部参数，但是只训练很少的几轮

# ❖ T5

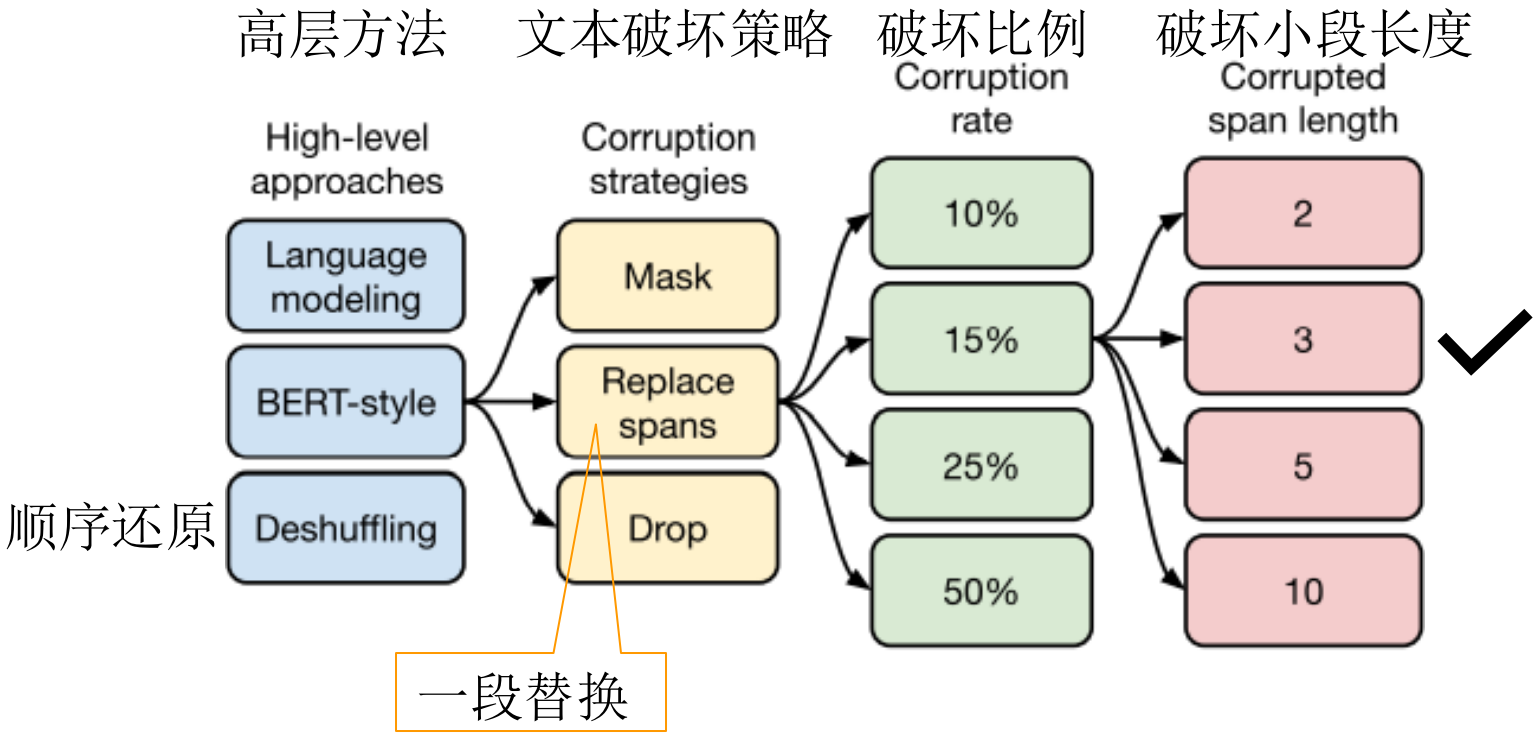
- Text-to-Text Transfer Transformer
- 2019.10 由 Google 提出
- Encoder-decoder结构
- 预训练任务: 掩码语言模型
  - 可能mask多个连续token
  - 输出被mask的序列，而非完整序列




Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2020. PDF

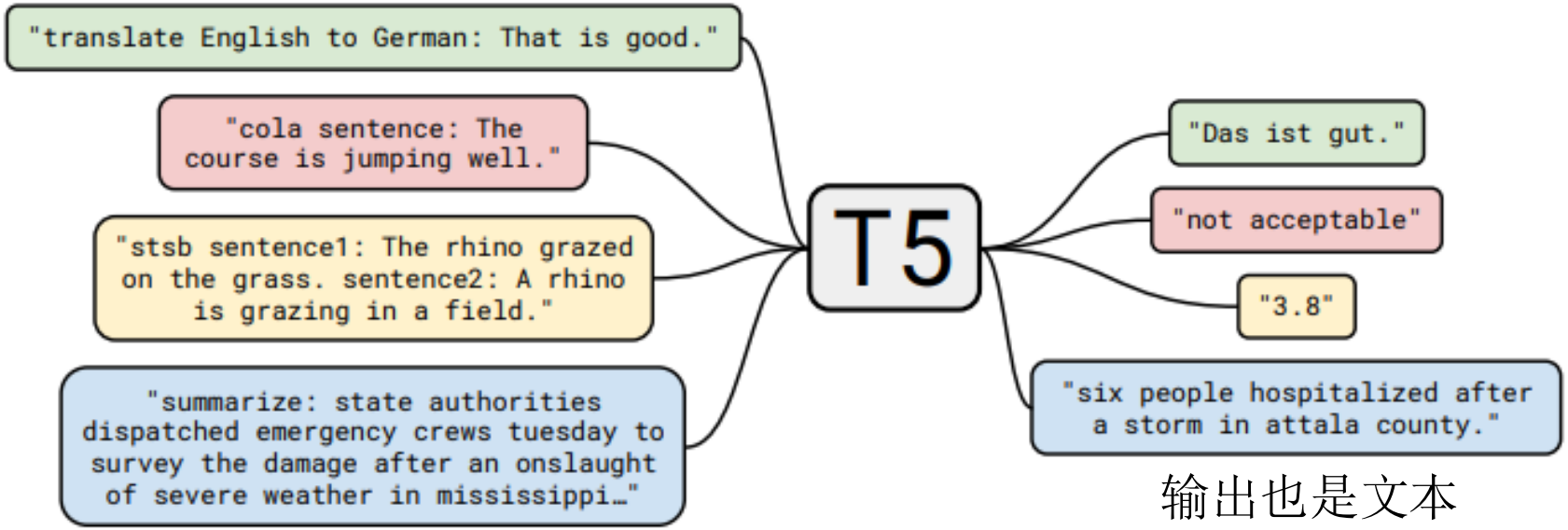
# T5

- 为了确定T5的结构，作者对预训练目标做了一个大范围的探索， 总共从四方面来进行比较



# T5：微调

- 把所有的NLP问题归结为“text-to-text”（文本到文本）的生成式任务
- 对于每个任务，使用文本作为模型的输入，训练模型生成目标文本，因此能够在多个任务上使用相同的模型、损失函数和超参数




输入是文本

输出也是文本