提示学习（Prompting）篇

来自： AiGC面试宝典

宁静致远

2023年09月18日 20:58

一、为什么需要 提示学习（Prompting）？

在面对特定的下游任务时，如果进行 **Full FineTuning**（即对预训练模型中的所有参数都进行微调），太过低效；而如果采用固定预训练模型的某些层，只微调接近下游任务的那几层参数，又难以达到较好的效果。

二、什么是 提示学习（Prompting）？

Prompt 提供上下文和任务相关信息，以帮助模型更好地理解要求，并生成正确的输出。

> 实例一：问答任务中，prompt 可能包含问题或话题的描述，以帮助模型生成正确的答案
> 实例二：在情感分析任务中，让模型做情感分类任务的做法通常是在句子前面加入前缀“该句子的情感是”即可，通过这种方式 将情感分类任务转换为一个“填空”任务，在训练过程中，BERT 可以学习到这个前缀与句子情感之间的关联。例如，它可以学习到“该句子的情感是积极的”和“该句子的情感是消极的”之间的差异。

三、提示学习（Prompting） 有什么优点？

提示学习（Prompting）旨在**通过最小化微调参数的数量和计算复杂度，来提高预训练模型在新任务上的性能，从而缓解大型预训练模型的训练成本**。这样一来，即使计算资源受限，也可以利用预训练模型的知识来迅速适应新任务，实现高效的迁移学习。

四、提示学习（Prompting）有哪些方法，能不能稍微介绍一下它们间？

4.1 前缀微调（Prefix-tining）篇

4.1.1 为什么需要 前缀微调（Prefix-tining）？

1. 人工设计离散的 Prompts 缺点：
   i. **Prompts 的变化对模型最终的性能特别敏感，**加一个词、少一个词或者变动位置都会造成比较大的变化
2. 自动化搜索离散的 Prompts 缺点：
   i. **成本也比较高**
3. **离散化的 token 搜索出来的结果可能并不是最优的；**
4. **传统的微调范式利用预训练模型去对不同的下游任务进行微调，对每个任务都要保存一份微调后的模型权重，**一方面微调整个模型耗时长；另一方面也会占很多存储空间

4.1.2 前缀微调（Prefix-tining）思路是什么？

- step 1 **Prefix 构建。在输入 token 之前构造一段任务相关的 virtual tokens 作为 Prefix**；
- step 2 **训练时只更新 Prefix 部分的参数，**而 Transformer 中的其他部分参数固定；
- step 3 **在 Prefix 层前面加了 MLP 结构**(相当于将 Prefix 分解为更小维度的 Input 与 MLP 的组合后输出的结果)，训练完成后，只保留 Prefix 的参数；（用于 防止直接更新 Prefix 的参数导致训练不稳定的情况）

4.1.3 前缀微调（Prefix-tining）的优点是什么？

1. 前缀微调（Prefix-tining） vs 人工设计离散的 Prompts 无法更新参数：**前缀微调（Prefix-tining）可以学习的“隐式”的 Prompts**；
2. 基于前缀的架构**可以在一个批次中处理来自多个用户/任务的样本**，这是其他轻量级微调方法所不能做到的；
3. vs full fine-tuning：full fine-tuning 更新所有参数，**Prefix Tuning 只更新 Prefix 部分的参数**；

4.1.4 前缀微调（Prefix-tining）的缺点是什么？

1. **占用序列长度**。有一定的额外计算开销；
2. **在每层都加了 prompt 的参数，改动较大**；

4.2 指示微调（Prompt-tuning）篇

4.2.1 为什么需要 指示微调（Prompt-tuning）？

1. **模型全量微调对每个任务训练一个模型，开销和部署成本都比较高**；
2. **离散的 prompts（指人工设计 prompts 提示语加入到模型）方法，成本比较高，并且效果不太好**；
3. **前缀微调（Prefix-tining）占用序列长度**。有一定的额外计算开销；
4. **前缀微调（Prefix-tining）在每层都加了 prompt 的参数，改动较大**；

4.2.2 指示微调（Prompt-tuning）思路是什么？

1. 将 prompt 扩展到连续空间，**仅在 输入层 添加 prompt 连续向量**，通过反向传播更新参数来学习 prompts，而不是人工设计 prompts；
2. **冻结模型原始权重，只训练 prompts 参数**，训练完成后，只用同一个模型可以做多任务推理；
3. **使用 LSTM 建模 prompt 向量间 关联性**

4.2.3 指示微调（Prompt-tuning）优点是什么？

1. 只在输入层加入 prompt tokens，并且**不需要加入 MLP 进行调整来解决难训练的问题**；
2. **随着预训练模型参数量的增加，Prompt Tuning 的方法会逼近全参数微调的结果**;
3. 提出了 prompt ensembling：在一个批次（Batch）里同时训练同一个任务的不同 prompt（即采用多种不同方式询问同一个问题），这样相当于训练了不同模型，比模型集成的成本小多了；

4.2.4 指示微调（Prompt-tuning）缺点是什么？

1. **训练难度加大**。不太好训练，省了显存，但不一定省时间。具体来讲，大部分 prompt 现在只是 parameter efficient 并没有达到想要的 training efficient。也就是说只是省了空间(显存)，但不一定能加快训练，训练时间有可能更长
2. **多个 prompt token 之间相互独立，可能会影响效果**
3. 在 NLU 上，prompt tuning 对于正常大小的预训练模型表现不佳；
4. 现有的 prompt tuning 方法不能处理困难的序列标注任务

4.2.5 指示微调（Prompt-tuning）与 Prefix-tuning 区别 是什么？

可以看作是 Prefix Tuning 的简化版本

1. 适用任务不同
    i. Prefix-tuning 仅针对 NLG 任务有效，服务于 GPT 架构；
    2.指示微调（Prompt-tuning） 考虑所有类型的语言模型
2. 添加方式不同
    i. Prefix-tuning 限定在输入前面添加
    ii. 指示微调（Prompt-tuning） 可以在任意位置添加
3. prompt 连续向量添加方式不同
    i. Prefix-tuning 每一层都添加，保证效果
    ii. 指示微调（Prompt-tuning） 可以只在 输入层添加

4.2.6 指示微调（Prompt-tuning）与 fine-tuning 区别 是什么？

1. Fine-tuning 需要改变预训练阶段模型参数， 可能带量灾难性遗忘问题
2. 指示微调（Prompt-tuning） 不改变预训练阶段模型参数，而是通过微调寻找更好的连续 prompt，来引导已学习到的知识使用

4.3 P-tuning 篇

4.3.1 为什么需要 P-tuning？

1. **大模型的 Prompt 构造方式严重影响下游任务的效果。**

   eg：GPT 系列 AR 建模在自然语言理解 NLU 任务上效果不好，与 BERT 双向语言模型相比有明显差距；

   注：GPT-3 采用人工构造的模版来做上下文学习（in context learning），但人工设计的模版的变化特别敏感，加一个词或者少一个词，或者变动位置都会造成比较大的变化

1. 之前的研究表明 GPT3 使用 prompt 训练方式可以显著提升 few-shot 和 zero-shot 的效果；

2. 自动化搜索模版工作成本也比较高，以前这种离散化的 token 的搜索出来的结果可能并不是最优的，导致性能不稳定；

4.3.2 P-tuning 思路是什么？

P-tuning 的核心思想：**通过调整 prompt 的参数，可以有效地引导模型生成期望的输出，而无需对模型的权重进行大规模的微调。**

1. **可学习的 Embedding 层 设计**。将 Prompt 转换为 可学习 Embedding 层；
2. **prompt encoder 设计**。用 prompt encoder（由一个双向的 LSTM+两层 MLP 组成） 的方式来对 Prompt Embedding 进行一层处理，建模伪 token 的相互依赖，并且可以提供一个更好的初始化。

4.3.3 P-tuning 优点是什么？

引入 prompt encoder（由一个双向的 LSTM+两层 MLP 组成）来建模伪 token 的相互依赖，并且可以提供一个更好的初始化；

4.3.4 P-tuning 缺点是什么？

1. **复杂性增加**。稍显复杂，看着不太像 prompt 了；
2. 伪 token 编码时是连续的，但在与输入结合时可能是不连续的，中间可能会插入输入

4.3.5 大模型微调 p_tuning 和传统 fine tuning 有什么区别？

传统 fine tuning 会对模型所有的参数或大部分参数进行微调，对于大模型的庞大参数量来说，会消耗更多的时间和资源等，而 P-tuning 通过只优化几个 token 的参数，在有限算力下也可微调大型预训练语言模型。

4.4 P-tuning v2 篇

4.4.1 为什么需要 P-tuning v2？

如何 让 Prompt Tuning 能够在不同参数规模的预训练模型、针对不同下游任务的结果上都达到匹敌 Fine-tuning 的结果；

4.4.2 P-tuning v2 核心思想 是什么？

1. **Deep Prompt Encoding**：采用 Prefix-tuning 的做法，在输入前面的每层加入可微调的 Prompts tokens 作为输入；
2. **移除了重参数化的编码器（prefix-tuning 中可选的 MLP、p-tuning 中的 LSTM）**：prefix-tuning 和 p-tuning，通过利用重参数化功能来提高训练速度和鲁棒性，但是 该方法对于较小的模型，同时还会影响模型的表现；
3. **针对不同任务采用不同的提示长度**。提示长度在提示优化方法的超参数搜索中起着核心作用。在实验中，发现不同的理解任务通常用不同的提示长度来实现其最佳性能，这与 Prefix-Tuning 中的发现一致，不同的文本生成任务可能有不同的最佳提示长度；
4. 引入多任务学习，先在多任务的 prompt 上进行预训练，然后再适配下游任务；
    i. 连续提示的随机惯性给优化带来了困难，这可以通过更多的训练数据或与任务相关的无监督预训练来缓解；
    ii. 连续提示是跨任务和数据集的特定任务知识的完美载体；

5. **抛弃了 prompt learing 中常用的 verbalizer，回归到传统的 CLS 和 token label 分类范式。**标签词映射器（Label Word Verbalizer）一直是提示优化的核心组成部分，它将 one-hot 类标签变成有意义的词，以利用预训练语言模型头。尽管它在 few-shot 设置中具有潜在的必要性，但在全数据监督设置中，Verbalizer 并不是必须的。它阻碍了提示调优在我们需要无实际意义的标签和句子嵌入的场景中的应用。因此，P-Tuning v2 回归传统的 CLS 标签分类范式，采用随机初始化的分类头（Classification Head）应用于 tokens 之上，以增强通用性，可以适配到序列标注任务。

4.4.3 P-tuning v2 优点是什么？

1. **在输入前面的每层加入可微调的 Prompts tokens 作为输入，**优点：
    i. 更多可学习的参数（从 P-tuning 和 Prompt Tuning 的 0.01%增加到 0.1%-3%），同时也足够参数高效；
    ii. 加入到更深层结构中的 Prompt 能给模型预测带来更直接的影响；
2. **解决了 Prompt Tuning 无法在小模型上有效提升的问题；**
3. 将 Prompt Tuning 拓展至 NER 等序列标注任务上

4.4.4 P-tuning v2 缺点是什么？

抛弃了 prompt learing 中常用的 verbalizer，回归到传统的 CLS 和 token label 分类范式，这其实某种程度上弱化了 prompt 的味道

4.4.5 P-tuning v2 主要特点是什么？

- **更灵活的 Prompt 设计**：P-tuning V2 允许使用更复杂的 prompt 结构例如包含多个句子或段落的 prompt，这有助于模型更好地理解和处理复杂的任务。
- **更高效的优化**：P-tuning V2 采用了更高效的优化策略，例如 0 使用 AdamW 优化器和适当的学习率调度，以加快训练过程并提高收敛速度。
- **更好的泛化能力**：通过改进的 prompt 设计和优化策略，P-tuning V2 在多个下游任务上显示出更好的泛化能力，即使在训练数据有限的情况下也能保持稳定的性能。
- **更少的调参需求**：P-tuning V2 减少了对超参数调整的需求，使得用户可以更容易地应用该方法到不同的任务上。
- **适应性**：P-tuning V2 可以适应不同的模型大小和架构，这使得它在不同的预训练模型上都具有较好的适用性。