Toolformer Summary

沙之洲 2020012408

Toolformer 这篇工作的背景是 LLM 已经在 nlp 层面的任务上有了很好的 few shot 的能力，但是在一些基础的计算表现上非常欠缺。基于上述问题，这篇工作提出了将 LLM 和 Tool 结合的结构，具体来说，使用 LLM 生成形如 <API> 的调用前缀以及具体调用内容。将返回结果作为 condition 继续进行 next token prediction。从实验结果来看，toolformer 在数学计算和日期计算的数据集上相比以往的模型有了大幅度的改进，同时也开创了将外部工具接入 LLM 的先河。

这篇文章的优点在于提出了一种基于 self-supervised 和原始 nlp 数据集构造 API 调用数据集的方法。具体来说，这种方法首先让给模型几个 API 的调用样例，让模型自己预测应该在回答的哪个地方插入 API 调用接口，接下来调用这些接口，根据生成结果的 loss 下降值来判断这些 API 调用是否是有效的。保留那些有效的 API 调用，和原始的数据集构成新的 API 调用数据集，用这个数据集去 finetune LLM 得到最终的 Toolformer。

这篇文章的不足之处在于，首先，在实验数据集的选取上，这篇工作从数据集中选取了那些“更有可能因为 API 调用而产生更好结果” 的样本作为数据。这在某种程度上导致实验部分，Toolformer 和其他模型的比较显得不那么公平。其次，这篇工作在 API 的调用上，禁止了 API 的多重调用，也就是说，在预测一个 token的时候，只能调用一次 API 接口。文中给的解释是这样可以防止 API 无限递归调用导致的阻塞。但是实际上我们可以允许多重 API 的嵌套调用，而对无限递归的应对方案是先通过实际调用采集一些样本，再通过人工设置过滤规则来避免这种无限递归调用的现象发生。

这篇文章下一步改进的方向可以考虑如下几个方面。首先考虑接触禁止多重 API 调用的限制，这样可以实现模型和 API 进行交互式的调用，文中也提到，toolformer 在一些 factual 问题上表现不好，一部分原因在于第一次调用 API 的返回结果并不理想，而禁止多重调用导致模型没有机会继续“追问”。进一步解放模型的“追问”能力，也意味着进一步解放模型的潜力。第二点在于，文中对于模型 API 调用是非常鼓励的，甚至只要 API token 出现在 next predict 的 top-k 里边，就会允许模型去调用 API，但是实际上 API 的调用也会耗费资源和时间，尽管多的 API 调用能够带来回答效果的提升，但是时间上的成本也会导致用户体验的急速下降。所以，该如何将 API 调用成本以及用户体验纳入到模型 decode 的阶段的考量当中是下一个阶段值得研究的问题。