Guiding Large Language Models via Directional Stimulus Prompting

摘要

本文介绍了方向性刺激提示（Directional Stimulus Prompting），这是一种引导黑盒大型语言模型（LLMs）朝着特定预期输出方向前进的新颖框架。本文的方法不直接调整大型语言模型，而是采用一个小型可调策略模型（如 T5），为每个输入实例生成辅助定向刺激提示。这些定向刺激提示可作为细粒度的、针对特定实例的提示和线索，引导 LLM 生成所需的结果，例如在生成的摘要中包含特定的关键词。本文的方法通过优化策略模型来探索定向刺激提示，使 LLM 与所需行为保持一致，从而避免了直接调整 LLM 的挑战。策略模型可以通过以下方式进行优化：1）使用标记数据进行监督微调；2）根据 LLM 的输出，从离线或在线奖励中进行强化学习。在总结、对话回复生成和思维链推理任务中对本文提出的方法进行了评估。实验表明，该框架在使用最少标记数据的情况下，持续提高了 LLM（如 ChatGPT、Codex、InstructGPT）在这些监督任务中的性能。值得注意的是，仅使用 MultiWOZ 数据集上的 80 个对话，本文的方法就将 ChatGPT 的性能提高了 41.4 %，达到或超过了一些完全有监督的最新模型。此外，与人工或自动生成的提示相比，本文的方法生成的实例特定的思维链提示提高了 InstructGPT 的推理准确性。

关键词： 提示；定向刺激；强化学习；LLM

1 引言

近年来，随着 Codex [2] 、InstructGPT、ChatGPT [5] 、GPT-4 、PaLM [3] 等大型语言模型（LLM）的兴起，自然语言处理（NLP）领域出现了一种新的范式。这些模型展现了新的能力 [6]，如强大的语境学习能力和少量提示能力，而这些能力是 BERT 、RoBERTa 、GPT-2 和 T5 等以前的” 小型” 语言模型（LM）所不具备的。这种范式的转变使 NLP 取得了显著的进步，LLM 显示出令人印象深刻的通用能力。然而，出于商业考虑和滥用风险，大多数 LLM 并不公开发布其参数，只允许用户通过黑盒 API 访问这些参数。虽然也有开源的 LLM，但针对特定任务或用例对其进行微调可能会导致计算效率低下。在这种情况下，利用 LLM 执行各种任务的标准方法是制作特定任务的文本提示，以便通过黑盒 API 查询 LLM。虽然 LLM 在各种语言任务中都表现出了不俗的性能，但在某些特定任务和用例中，它们仍难以生成完全符合预期行为和方向的输出结果。

由于针对特定任务直接优化 LLM 要么效率低下，要么对大多数用户和开发人员来说不可行，因此研究人员转而采用优化提示的方法。提示工程方法涉及手动或自动设计针对特定任务的最佳自然语言指令，并选择适当的训练样本在提示中进行演示，一直是许多研究人员

关注的焦点。尽管做出了这些努力，但有效引导 LLM 生成所需的结果和有效利用标记数据仍然是一项重大挑战。

为了应对这一挑战，本文提出了一个名为定向刺激提示（DSP）的新框架。该框架在提示中引入了一个名为” 方向性刺激” 的新组件，为 LLM 提供细致入微、针对特定实例的指导和控制。具体来说，方向性刺激提示对输入查询起到” 提示” 和” 线索” 的作用，引导 LLM 朝着所需的输出方向前进。值得注意的是，这与从外部来源获取额外知识来增强 LLM 的方法不同 [4]，因为在 DSP 框架中，方向性刺激提示完全是根据输入查询生成的。图 1 比较了 DSP 提示方法和总结任务的标准提示方法。DSP 方法在提示中加入了关键词作为定向刺激提示，以提示所需的摘要应涵盖的关键点。通过定向刺激提示提供这种针对具体实例的指导，LLM 可以生成与所需参考摘要更加一致的输出结果。

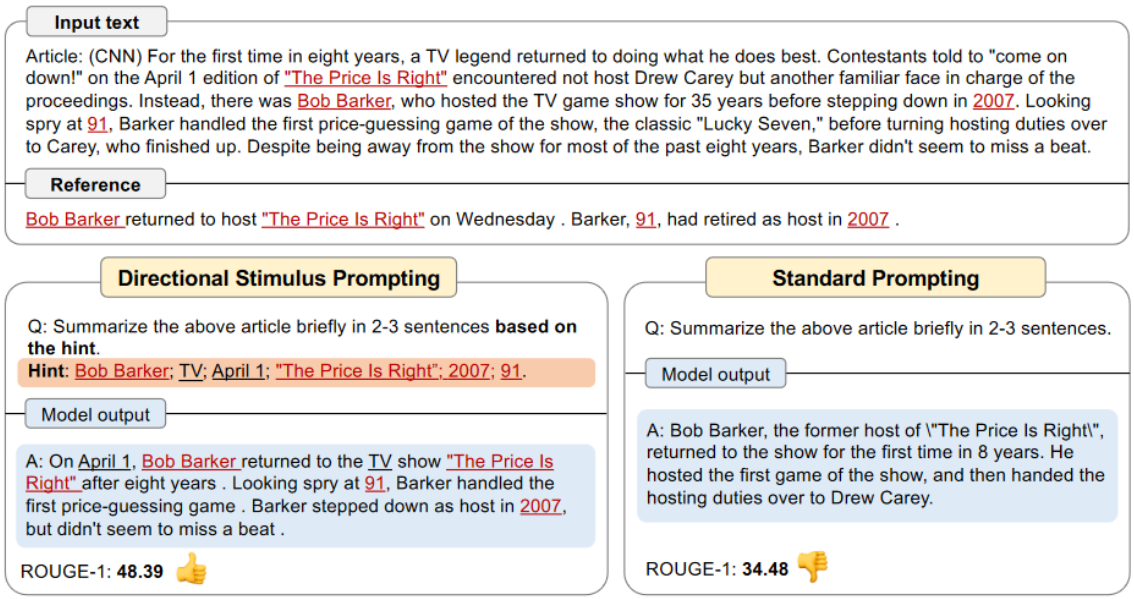

图 1. DSP 方法与标准提示方法对比

本文使用一个相对较小且可调的 LM（e.g．，T5）作为策略模型，为每个输入查询生成方向刺激提示。这种做法使得我们可以通过优化小型可调策略模型来避开对黑箱 LLMs 的直接优化。本文通过有监督的微调（SFT）使用少量收集到的有标签数据来训练策略模型。在有监督的微调后，进一步优化策略模型，利用强化学习（RL）探索更好的定向刺激提示。在强化学习训练中，我们的目标是在策略模型产生的刺激条件下，最大化对下游任务表现或对 LLM 输出的表现的一个奖励值。

图 2 以摘要任务为例，概述了 DSP 框架。本文采用了一个紧凑、可调整的策略模型来生成定向刺激提示，该提示指定了应包含在 LLM 生成的摘要中的关键词。策略模型可用 SFT 和 RL 训练，其中奖励通常定义为下游任务的性能指标，如摘要任务的 ROUGE 分数，或其他对齐指标，如人类偏好。

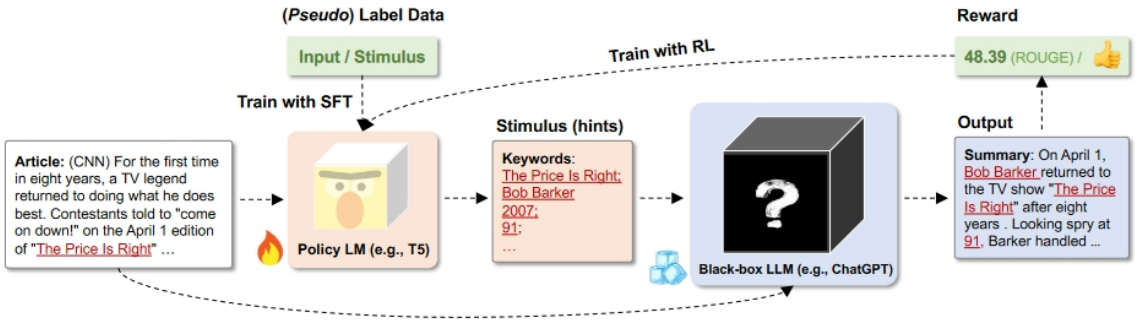

图 2. DSP 框架

通过选择适当的定向刺激和相关奖励，DSP 框架可以灵活地适用于各种 LLM 和任务。本文在总结、对话回复生成和思维链推理任务上进行了实验，以评估 DSP 框架的有效性。实验结果表明，DSP 方法可以有效地引导 ChatGPT 朝着所需的目标前进，而且只需收集少量的标记数据。具体来说，本文使用黑盒 LLM ——ChatGPT 进行了实验。在策略模型方面，使用了 750M Flan-T5-Large 。在摘要任务中，本文使用关键词作为定向刺激，提示所需的摘要应包括的关键点。尽管 ChatGPT 的性能已经相当可观，但使用 CNN/Daily Mail 数据集中仅 4,000 个样本训练的策略模型 T5 的 ROUGE 和 BLEU 分数提高了 4-13%。在对话应答生成任务中，本文对策略模型进行了训练，以生成对话行为，这些行为表明了来自 MultiWOZ 数据集的对话中目标应答背后的潜在意图。

2 相关工作

2.1 黑盒大语言模型

近年来，GPT-3 、Codex 、InstructGPT、ChatGPT 、PaLM 和 LaMDA 等 LLM 相继问世，在 NLP 领域大有可为。这些 LLM 通常有大量参数，需要大量训练数据。由于它们的可扩展性，这些模型表现出了许多新兴能力，如上下文学习、少量提示、思维链提示和指令跟随等。然而，大多数 LLM 都没有开源，只能通过黑盒 API 访问，用户通过黑盒 API 发送提示查询并接收响应。虽然存在开源 LLM，如 OPT-175B 和 Bloom ，但它们的本地执行和微调需要大量计算资源，对于大多数研究人员和用户来说可能是不可行的。然而，尽管 LLM 在各种任务中表现出色，但在特定的下游任务和用例中，LLM 通常无法生成与所需输出完全一致的输出。DSP 方法旨在解决这一局限性，将小型可调 LM 生成的定向刺激引入提示，从而提供比黑盒 LLM 更精细的指导和控制。

2.2 及时优化和工程化

通过寻找最佳提示，在下游任务中有效优化预训练 LM 是之前研究的重点。其中一种方法是调整软提示，即使用梯度下降方法优化连续嵌入向量。然而，梯度的要求以及通过黑盒 API 传递梯度和连续提示的挑战，使得它们对于黑盒 LLM 不太实用。研究人员还试图通过设计特定任务的自然语言指令和选择适当的训练样本作为提示中的上下文演示来寻求最佳提示。这些方法包括人工工程、编辑、强化学习和自动生成。尽管做出了这些努力，但这些提示并不总能有效地引导 LLM 生成所需的输出，尤其是对于细粒度的特定实例行为，因为这些

行为很难使用特定任务的指令和演示示例来描述。为了解决这一局限性，DSP 方法能够提供更精细的针对特定实例的指导，这种指导由一个小型可调策略模型生成，该模型通过监督微调和强化学习进行了优化。

2.3 可控的文本生成

对语言模型（LMs）的控制已经进行了广泛的研究。早期的方法是在包含所需属性的数据集上对 LM 进行微调。Keskar, N. S. 等人提出了以类别为条件的 LM ，用预定义的控制代码生成文本。然而，直接 LM 训练成本高昂。为了解决这个问题，PPLM 训练了一个属性模型，并将梯度传递进去来控制生成。GeDi 和 DExperts 使用类条件分布作为生成判别器来指导生成，从而降低了计算复杂度。这些方法要么需要额外的 LM 训练，要么需要内部梯度和计算，因此不适用于黑盒 LLM。DSP 方法提出了一种控制黑盒 LLM 的解决方案，即在输入查询提示中插入定向刺激，并根据返回输出进行优化。

2.4 面向 NLP 的强化学习

强化学习已成功应用于各种 NLP 任务，如句法分析、机器翻译、摘要、会话系统等。语言模型定义了其词汇表中词组的概率分布，文本生成问题可以自然地表述为在 RL 设置中选择一个动作。因此，人们在利用 RL 优化 LM 方面做了大量研究，通常是通过使 LM 与人类偏好相一致来实现。例如，LLM InstructGPT 通过 RL 进行了优化，以更好地遵循用户的指令和意图。与这些直接更新 LLM 以与人类偏好保持一致的工作相比，本文的工作优化了一个生成文本（刺激）的小型策略模型，以引导 LLM 生成更多人类偏好的输出，而不是直接优化 LLM，绕过了低效 LLM 的优化。

3 DSP 方法

对于下游任务，有一个输入空间 X、X 上的数据分布 D 和一个输出空间 Y。由于 LLM 具备强大的上下文学习能力和少量提示能力，因此只要在提示中包含描述任务的指令、少量示范示例和输入查询 x，LLM 就能执行各种任务并生成输出 Y [1]。然而，这种提示并不能始终引导 LLMs 实现所需的输出，尤其是在涉及到细粒度的特定实例所需的行为时。例如，在摘要任务中，输入 x 是一篇文章，输出 y 是相应的摘要。不同的摘要生成器风格不同，强调文章的不同方面。在这种情况下，仅仅依靠特定任务指示或示范示例来描述每个样本的细微差别，可能不足以有效地引导 LLM 生成与参考摘要非常匹配的摘要。

为此，方向性刺激提示（DSP）方法在提示中引入了一小段名为” 方向性刺激” 的离散标记 z，作为提示和线索，为 LLMs 提供指向所需方向的细粒度指导。例如，在摘要任务中，方向性刺激 z 可能包括应包含在所需摘要中的关键词。为了为每个输入查询生成这种刺激，本文使用了一个小型可调策略语言模型 $P_{POL}(z|x)$。然后，我们使用生成的刺激（z）和原始输入（x）来构建提示，通过黑盒 API 调用引导 LLM 生成输出 $P_{LLM}(y|x,z)$。值得注意的是，LLM 的参数 P_{LLM} 是不可访问和调整的。总的来说，在使用 LLM 和 DSP 执行下游任务时，输出是通过 y $P_{LLM}(\cdot|x,z)$, z $P_{POL}(\cdot|x)$ 获得的。

3.1 有监督的微调

为了训练为 LLM 生成方向性刺激的策略模型，本文首先在预先训练好的 LM（如 T5、GPT-2 等）上对少量标注数据进行监督微调（SFT）。在收集数据时，可以根据下游任务，启发式地为每个输入查询 x 和目标输出 y 选择出" 伪刺激"z^*。例如，对于摘要任务，可以使用参考摘要中包含的关键词作为伪刺激，而对于对话响应生成任务，则使用表明所需系统响应基本含义的对话行为。由此产生的数据集 D $=$ (x, z^*) 由输入-刺激对组成。然后，我们通过最大化对数似然对策略模型进行微调：

$$L_{SFT} = -E_{(x,z^*)\ D'} log P_{POL}(z^*|x)$$

3.2 强化学习

优化目标 本文的目标是通过最大化对齐度量 R 来引导 LLM 生成所需的目标，对齐度量可以有多种形式，如下游任务性能指标（如用于摘要的 ROUGE 分数）、人类偏好或其他定制指标。在数学上，我们的目标是最大化以下目标：

$$E_{x\ D,z\ P_{POL}(\cdot|x),y\ P_{LLM}(\cdot|x,z)}[R(x,y)]$$

由于黑箱 LLM 的参数不可获取及不可调节，本文通过优化策略模型来产生方向性的刺激，引导 LLM 朝着最大化目标的方向生成。为了实现这一点，本文定义了另一个度量 R_{LLM}，它捕获了在给定刺激 z 的条件下，LLM 的表现：

$$R_{LLM}(x,z) = R(x,y), y\ P_{LLM}(\cdot|x,z)$$

这样，就能将最大化 R 这个原始目标转化为优化策略模型，以生成 R_{LLM} 最大化情况下的刺激。这样，LLM 就被有效地用作一个评估函数，引导策略模型产生更有效的定向刺激。因此，3.2 节第一个等式中 LLM 的优化目标等同于政策模型的优化目标：

$$max_{P_{POL}} E_{x\ D,z\ P_{POL}(\cdot|x)}[R_{LLM}(x,z)]$$

强化学习公式化 然而，对于策略模型来说，上述优化是难以实现的。为了解决这个问题，本文将策略模型优化表述为一个 RL 问题，并采用 PPO 算法。本文使用策略模型初始化策略网络 $_0 = P_{POL}$，然后使用 PPO 更新 $_0$ 策略模型生成令牌序列作为刺激 z 的过程，可以看作是一个马尔可夫决策过程 $MDP\ \ S, A, r, P$ ，其中有状态空间 S、行动空间 A、奖励函数 r 和状态转换概率 P。在一集的每个时间步骤 t 中，代理根据当前策略网络 $(z|x, z < t)$ 的分布，从词汇表 V 中选择一个行动（令牌）。当选择了一个序列结束标记时，该回合结束，并生成刺激 z。我们可以通过优化奖励 r 来微调策略网络 ：

$$E_\pi[r] = E_{x\ D,z\ \pi(\cdot|x)}[r(x,z)]$$

奖励函数 回想一下，本文的目标是最大化 3.2 节的第三个等式中的目标，它可以用作奖励 r。为了防止策略网络 偏离初始策略模型 P_{POL} 太远，我们还增加了 KL 散度惩罚奖励。因此，最终奖励变为：

$$r(x,z) = R_{LLM}(x,z) - \beta log \frac{\pi(z|x)}{P_{POL}(z|x)}$$

在训练过程中动态地调整系数 β:

$$e_t = clip(\frac{KL(\pi_t, P_{POL}) - KL_{target}}{KL_{target}}, -0.2, 0.2)$$

$$\beta_{t+1} = \beta_t(1 + K_\beta e_t)$$

实现 为了优化策略网络 π，本文使用了专为语言生成器设计的 PPO 的 NLPO 版本。为了解决 PPO 中动作空间过大的问题，NLPO 学会使用 top-p 采样来屏蔽词汇中相关性较低的标记。这种技术将动作空间限制在累积概率大于给定概率参数 p 的最小标记集上，本文在实验中将概率参数设置为 0.9。策略网络 和价值网络都是从监督微调策略模型 P_{POL} 中初始化的，价值网络的最后一层是随机初始化的，使用回归头输出一个标量值。

4 复现细节

4.1 与已有开源代码对比

模型选择上 将 LLM 模型从 ChatGPT 换成 LLAMA2-7b 模型在摘要任务上进行了测试。

```
if self.use_llm:
    prompt_text = prompt_texts[i]
    t5_input_text = prompt_text

    llm_input_text = self.hint_prompt.replace("[[QUESTION]]",
    t5_input_text)
    llm_input_text = llm_input_text.replace("[[HINT]]",
    t5_gen_text)
    sequences = llm_pipeline(
            llm_input_text,
            do_sample=True,
            temperature=self.temperature,
            top_p=self.top_p,
            num_return_sequences=1,
            eos_token_id=llm_tokenizer.eos_token_id,
            max_new_tokens=self.max_tokens,
            return_full_text=False
            )
    llm_gen_texts = [LAMARewardSummarizationWithHint.
    clean_generation(seq['generated_text'], self.stop_words)
    for seq in sequences]
    llm_gen_texts = LLAMARewardSummarizationWithHint.
    generation_selection(self.selection_strategy ,llm_gen_texts)
    llm_generated_texts.append(*llm_gen_texts)
    # reward for llm
```

```
    for j, gpt3_metric in enumerate(self.gpt3_metrics):
        gpt3_score_keys = self.gpt3_score_keys[j]
        gpt3_scores = [[] for _ in range(len(gpt3_score_keys))]
        for g, llm_gen_text in enumerate(llm_gen_texts):
            metric_results = gpt3_metric.compute([t5_input_text],
            [llm_gen_text], [reference_text])
            for k,score_key in enumerate(gpt3_score_keys):
                score = metric_results[score_key][1]
                score = 0 if score == 'n/a' else score
                gpt3_scores[k].append(score)
                # average score
                for k, score_key in enumerate(gpt3_score_keys):
                    avg_score = np.mean(gpt3_scores[k])
                    llm_rewards[j][k].append(avg_score)
                    print(f"{score_key}: {avg_score}")
```

reward 选择 原文使用 ROUGE-avg 作为 reward 进行强化学习，我尝试了其他的 reward，比如说 BertScore、Meteor 等以及将多个指标按一定比例进行混合作为 reward。

4.2 实验环境搭建

本实验在 Linux 服务器上进行，所使用的显卡是一张 A100。

5 实验结果分析

原文重点研究了摘要生成、对话回复生成和自动提示生成任务。原文主要使用预训练的 T5 或 Flan - T5 来初始化策略模型，并评估 OpenAI 的 ChatGPT (gpt-3.5-turbo)。我所做的实验复现了原文工作，除此之外，在上述模型选择与 reward 选择上进行了更改。在摘要任务上，对原文的复现结果、更换 reward 的结果以及将模型更换为 LLAMA2-7b 的结果见表 1 。

Method	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	METEOR	BERTScore
DSP	39.99	17.45	26.69	8.99	32.59	0.8843
DSP 复现	39.18	16.07	25.90	6.21	31.53	0.8824
混合 reward	38.90	15.77	25.75	6.16	31.44	0.8823
LLAMA2-7b	36.99	14.77	24.38	5.67	30.74	0.8749

表 1. 摘要任务复现与更改 reward 或模型的结果显示表

在对话生成任务上，对原文的复现结果见图 3

	Training Data	MultiWoz2.0				MultiWoz2.1			
		Inform	Succ.	BLEU	Comb.	Inform	Succ.	BLEU	Comb.
DSP w/SFT	1%(80)	74.9	66.3	11.1	81.7	72.0	66.0	11.3	80.1
DSP w/SFT+RL	1%(80)	91.0	76.0	9.8	93.3	89.7	78.6	9.4	93.4
DSP w/SFT	10%(800)	79.4	71.9	11.3	87.0	72.0	67.0	13.1	82.6
DSP w/SFT+RL	10%(800)	96.0	86.9	10.7	102.2	94.0	86.0	9.2	99.2
DSP w/SFT	1%(80)	66.8	26.7	4.3	51.1	64.5	22.4	4.5	47.9
DSP w/SFT+RL	1%(80)	75.7	47.0	6.0	67.3	81.2	62.8	6.0	77.2
DSP w/SFT	10%(800)	71.9	20.9	5.7	52.1	72.0	16.3	5.6	49.8
DSP w/SFT+RL	10%(800)	75.8	58.4	7.6	74.7	78.3	59.6	7.4	76.3

图 3. 对话生成任务复现结果示意图

6 总结与展望

本文介绍了定向刺激提示（DSP），这是一种新的提示框架，可为黑盒 LLM 提供细粒度的、针对特定实例的引导，以实现所需的输出。本文使用可调整的策略模型来生成定向刺激，以提供此类指导，并将黑盒 LLM 的优化转换为策略模型的优化。实验结果证明了 DSP 方法的有效性。DSP 不仅能更好地控制和引导黑盒 LLM，还能有效利用标记数据。此外，生成的刺激还能为 LLM 的行为提供有价值的见解和解释。在这项工作中，我们使用启发式选择或注释的伪刺激数据对策略模型进行监督微调。在未来的工作中，可以探索在策略模型和 LLMs 之间使用”机器语言”的可能性，这种语言可能不会受到人类直觉的偏好，但却能更好地传达引导信息，以及文本以外的其他形式的定向刺激。

参考文献

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray,

Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[4] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023.

[5] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[6] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.