

大型語言模型是人類水準的提示工程師

Yongchao Zhou^{1,2,*}、Andrei Ioan Muresanu^{2,3,*}、Ziwen Han^{1,2,*}、Keiran Paster^{1,2}、Silviu Pitis^{1,2}、Harris Chan^{1,2}、Jimmy Ba^{1,2}

1多倫多大學 2Vector Institute 3滑鐵盧大學 *同等貢獻

{yczhou,hanziwen,keirp,spitis,hchan,jba}@cs.toronto.edu {andrei.muresanu}@uwaterloo.ca

摘要

透過自然語言指令的條件設定，大型語言模型 (LLM) 已展現出作為通用電腦的卓越能力。然而，任務效能很大程度上取決於用於引導模型的提示品質，而大多數有效的提示都是由人類手工製作的。受經典程式合成和人類提示工程方法的啟發，我們提出了自動提示工程師 (APE)，用於自動指令生成和選擇。在我們的方法中，我們將指令視為「程式」，透過搜尋 LLM 提出的一組指令候選來進行最佳化，以最大化所選的評分函數。為了評估所選指令的品質，我們評估了另一個 LLM 遵循所選指令的零樣本效能。大量實驗表明，我們自動生成的指令在 24/24 的指令歸納任務和 17/21 的精選 BIG-Bench 任務上，大幅優於先前的 LLM 基準，並達到與人類註釋者生成的指令更好或相當的效能。我們進行了廣泛的定性和定量分析，以探索 APE 的效能。我們證明 APE 工程的提示能夠提高少樣本學習效能 (只需將它們預先附加到標準的上下文學習提示中)，找到更好的零樣本思維鏈提示，以及引導模型走向真實性和/或資訊性。²

1 簡介

規模和基於注意力的架構相結合，使得語言模型具有前所未有的通用性 (Kaplan 等人，2020 年；Vaswani 等人，2017 年)。這些所謂的「大型語言模型」展現出卓越的、通常是超乎人類的能力，涵蓋各種任務，包括零樣本和少樣本設定 (Brown 等人，2020 年；Srivastava 等人，2022 年)。然而，隨著通用性而來的是控制問題：我們如何讓大型語言模型執行我們希望它們執行的任務？

為回答此問題並引導大型語言模型產生預期行為，近期研究已考量微調 (Ouyang et al., 2022; Ziegler et al., 2019)、情境學習 (Brown et al., 2020) 和數種形式的提示生成 (Gao, 2021)，包括軟提示的可微分調整 (Qin & Eisner, 2021; Lester et al., 2021) 和自然語言提示工程 (Reynolds & McDonell, 2021)。後者特別受關注，因為它為人類與機器溝通提供了自然介面，不僅與大型語言模型高度相關，也與其他通用模型高度相關，例如提示式影像合成器 (Rombach et al., 2022; Ramesh et al., 2022)，公眾對提示設計和生成的興趣也隨之興起 (範例請參閱附錄 A)。

這種興趣背後的事實是，即使使用替代指令可以產生這些結果，但白話提示也並非總能產生預期的結果。因此，人類使用者必須嘗試各種提示才能引出所需的行為，因為他們對指令與特定模型相容的程度知之甚少。我們可以將大型語言模型 (LLM) 視為黑箱電腦，它們執行由自然語言指令指定的程式：雖然它們

¹我們將「提示工程」定義為最佳化提示中的語言，以引出最佳效能。值得注意的是，這不包括將多個 LLM 查詢串聯在一起或讓 LLM 存取外部工具的提示。²我們的程式碼可在 https://github.com/keirp/automatic_prompt_engineer 取得。

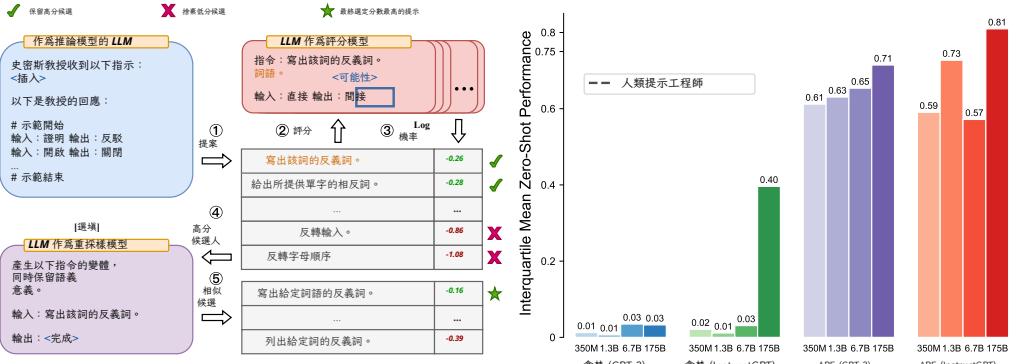


圖 1：(a) 我們的<標籤 id='1'>方法</標籤>，自動提示工程師 (APE)，會自動為透過輸出示範指定的任務產生指令：它會產生數個指令候選，可透過直接推斷或基於語義相似度的遞迴程序，使用目標模型執行這些指令，並根據計算出的評估分數選取最合適的指令。(b) 根據 Honovich 等人 (2022) 提出的 24 項 NLP 任務的四分位數平均值測量，APE 在使用 InstructGPT 模型 (Ouyang 等人，2022) 時，能夠超越人類表現。

可以執行廣泛的自然語言程式，但這些程式的處理方式可能對人類來說並不直觀，而且指令的品質只能在下游任務中執行這些指令時才能衡量 (Sanh 等人，2022 年；Wei 等人，2021 年)。

<style id='1'>我們提出了一種新穎的演算法，該演算法使用大型語言模型自動生成和選擇指令，以減少建立和驗證有效指令所需的人力。我們將此問題稱為自然語言程式合成，並建議將其作為黑箱最佳化問題來解決，使用大型語言模型生成並搜尋啟發式可行的候選解決方案。為此，我們透過三種方式利用大型語言模型的通用能力。首先，我們使用大型語言模型作為推論模型 (Ellis 等人，2021 年；Honovich 等人，2022 年)，根據一小組輸入-輸出對形式的示範來生成指令候選。接下來，我們透過計算我們尋求控制的大型語言模型下每個指令的分數來引導搜尋過程。最後，我們提出了一種迭代蒙地卡羅搜尋方法，其中大型語言模型透過提出語義相似的指令變體來改進最佳候選。直觀地說，我們的演算法要求大型語言模型根據示範生成一組指令候選，然後要求它們評估哪些指令更有前景。我們將我們的演算法稱為自動提示工程師 (APE)。我們的主要貢獻是：</style>

- 我們將指令生成視為自然語言程式合成，將其公式化為由大型語言模型引導的黑箱最佳化問題，並提出一種樸素和一種迭代的蒙地卡羅搜尋方法來近似解。
- 我們提出的方法 APE，在 24/24 個指令歸納和 17/21 個 Big-Bench 任務中，透過模型生成的指令，在零樣本學習方面達到了人類水準的表現。
- 我們提供了廣泛的定性和定量分析，探索 APE 的各個方面，並展示 APE 在改進少樣本學習、尋找更好的零樣本思維鏈提示，以及引導大型語言模型朝向所需行為（例如真實性和/或資訊性）方面的應用。

2 相關工作

大型語言模型擴展基於 Transformer 的語言模型，包括模型大小、訓練資料和訓練計算，已被證明能可預測地改善各種下游自然語言處理任務的效能 (Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020)。由於這種擴展，已發現許多大型語言模型的湧現能力 (Wei et al., 2022a)，包括少樣本情境學習、零樣本問題解決、思維鏈推理、指令遵循和指令歸納 (Cobbe et al., 2021; Wei et al., 2022b; Kojima et al., 2022)。

2022; Sanh 等人, 2022; Wei 等人, 2021; Ouyang 等人, 2022; Honovich 等人, 2022)。在本文中，我們將大型語言模型視為黑箱電腦，執行由自然語言指令指定的程式，並研究如何使用模型生成的指令來控制大型語言模型的行為。

提示工程 提示為人類提供了一個自然直觀的介面，用於與 LLM 等通用模型互動和使用。由於其靈活性，提示已被廣泛用作 NLP 任務的通用方法 (Schick & Schütze, 2021; Brown et al., 2020; Sanh et al., 2022)。然而，LLM 需要仔細的提示工程，無論是手動 (Reynolds & McDonell, 2021) 還是自動 (Gao et al., 2021; Shin et al., 2020)，因為模型似乎不像人類那樣理解提示 (Webson & Pavlick, 2021; Lu et al., 2021)。儘管許多成功的提示調整方法使用基於梯度的方法在連續空間上執行最佳化 (Liu et al., 2021; Qin & Eisner, 2021; Lester et al., 2021)，但隨著規模的擴大，這變得不切實際，因為計算梯度變得越來越昂貴，並且對模型的存取轉向可能不提供梯度存取的 API。在我們的論文中，我們借鑒了離散提示搜尋方法中的元件，例如提示生成 (Gao et al., 2021; Ben-David et al., 2021)、提示評分 (Davison et al., 2019) 和提示釋義 (Jiang et al., 2020; Yuan et al., 2021)，透過直接在自然語言假設空間中搜尋來最佳化指令。與過去的工作相比，過去的工作為每個元件使用專門的模型並嚴重依賴人類範本，我們證明整個搜尋可以由單一 LLM 執行。

程式合成 程式合成涉及在「程式空間」中自動搜尋，以找到滿足特定規格的程式 (Gulwani et al., 2017)。現代程式合成允許各種規格，包括輸入-輸出範例 (Ellis et al., 2021; Wong et al., 2021) 和自然語言 (Jain et al., 2022)。可搜尋的程式空間範圍也已擴大，從歷史上受限的領域特定語言到通用程式語言 (Austin et al., 2021)。與需要合適的結構化假設空間和元件庫的先前方法 (Liang et al., 2010; Ellis et al., 2018) 不同，我們利用 LLM 提供的結構來搜尋自然語言程式的空間。使用推論模型是透過將搜尋空間限制在有限的可能表達空間來加速搜尋的標準做法 (Menon et al., 2013; Lee et al., 2018; Devlin et al., 2017; Ellis et al., 2021)。受此啟發，我們使用 LLM 作為近似推論模型，根據一小組示範生成程式候選。與傳統程式合成不同，我們的推論模型不需要任何訓練，並且能很好地推廣到各種任務。

3 使用 LLM 的自然語言程式合成

我們考慮一項任務，該任務由從母體 \mathcal{X} 中取樣的輸入/輸出示範資料集 $\mathcal{D}_{\text{train}} = \{(Q, A)\}$ 和提示模型 \mathcal{M} 指定。自然語言程式合成的目標是找到單一指令 ρ ，使得當 \mathcal{M} 收到指令和給定輸入的串聯 $[\rho; Q]$ 的提示時， \mathcal{M} 會產生對應的輸出 A 。更正式地說，我們將其視為一個最佳化問題，我們尋求指令 ρ ，以最大化每個樣本分數 $f(\rho, Q, A)$ 在所有可能的 (Q, A) 上的期望值：

$$\rho^* = \arg \max_{\rho} f(\rho) = \arg \max_{\rho} \mathbb{E}_{(Q, A)} [f(\rho, Q, A)] \quad (1)$$

請注意，一般而言， Q 可以是空字串，因此我們正在最佳化 ρ 作為直接產生輸出 $\{A\}$ 的提示。雖然這項任務已被人類廣泛嘗試，但我們對任何特定指令與模型 \mathcal{M} 的相容性知之甚少。因此，我們建議將這個人類難以處理的問題視為由 LLM 引導的黑箱最佳化過程。我們的演算法 APE 在提案和評分這兩個關鍵元件中都使用了 LLM。如圖 1 所示並在演算法 1 中總結，APE 首先提出一些候選提示，然後根據選定的評分函數篩選/精煉候選集，最終選擇得分最高的指令。我們接下來將討論提案和評分的選項。

3.1 初始提案分佈

由於搜尋空間無限大，找到正確的指令可能極其困難，這使得自然語言程式合成在歷史上難以處理。自然語言處理的最新進展表明，語言模型非常擅長生成多樣化的自然語言文本。因此，我們

Algorithm 1 Automatic Prompt Engineer (APE)

Require: $\mathcal{D}_{\text{train}} \leftarrow \{(Q, A)\}_n$: training examples, $f : \rho \times \mathcal{D} \mapsto \mathbb{R}$: score function

- 1: Use LLM to sample instruction proposals $\mathcal{U} \leftarrow \{\rho_1, \dots, \rho_m\}$. (See Section 3.1)
- 2: **while** not converged **do**
- 3: Choose a random training subset $\tilde{\mathcal{D}}_{\text{train}} \subset \mathcal{D}_{\text{train}}$.
- 4: **for all** ρ in \mathcal{U} **do**
- 5: Evaluate score on the subset $\tilde{s} \leftarrow f(\rho, \tilde{\mathcal{D}}_{\text{train}})$ (See Section 3.2)
- 6: **end for**
- 7: Filter the top k% of instructions with high scores $\mathcal{U}_k \subset \mathcal{U}$ using $\{\tilde{s}_1, \dots, \tilde{s}_m\}$
- 8: Update instructions $\mathcal{U} \leftarrow \mathcal{U}_k$ or use LLM to resample $\mathcal{U} \leftarrow \text{resample}(\mathcal{U}_k)$ (See Section 3.3)
- 9: **end while**

Return instruction with the highest score $\rho^* \leftarrow \arg \max_{\rho \in \mathcal{U}_k} f(\rho, \mathcal{D}_{\text{train}})$

考慮利用預訓練的 LLM 來提出一組好的候選解決方案 \mathcal{U} ，以指導我們的搜尋程序。雖然來自 LLM 的隨機樣本不太可能產生所需的 (Q, A) 對，但我們可以要求 LLM 在給定輸入/輸出示範的情況下，近似推斷出得分最高的指令；也就是說，近似地從 $P(\rho | \mathcal{D}_{\text{train}}, f(\rho) \text{ 很高})$ 中取樣。

前向模式生成 我們考慮兩種方法來從 $P(\rho | \mathcal{D}_{\text{train}}, f(\rho) \text{ 很高})$ 中取樣。首先，我們採用基於「前向」模式生成的方法，將此分佈 $P(\rho | \mathcal{D}_{\text{train}}, f(\rho) \text{ 很高})$ 轉換為文字。例如，在我們的指令歸納實驗（第 4.1 節）中，我們遵循 Honovich 等人 (2022) 的做法，使用圖 2 (上) 提示 LLM。

反向模式生成 儘管「前向」模型適用於大多數預訓練的 LLM，但將 $P(\rho | \mathcal{D}_{\text{train}}, f(\rho) \text{ 很高})$ 轉換為文字需要針對不同任務進行客製化工程。這是因為指令通常出現在段落的開頭，而「前向」模型只會從左到右生成文字，這要求指令在提示的末尾進行預測。因此，我們需要一種更靈活的方法，使指令可以出現在文字中的任何位置。為了解決這個問題，我們考慮「反向」模式生成，它使用具有填充功能的 LLM，例如 T5 (Raffel 等人，2020)、GLM (Du 等人，2022) 和 InsertGPT (Bavarian 等人，2022)，來推斷缺失的指令。我們的「反向」模型透過填空直接從 $P(\rho | \mathcal{D}_{\text{train}}, f(\rho) \text{ 很高})$ 中取樣。我們在圖 2 (中) 中展示了此範本的範例。

<style id='1'>客製化提示</style>請注意，根據所使用的評分函數，可能存在比上述範例更合適的提示。例如，在我們的 TruthfulQA 實驗中，我們從原始資料集 (Lin et al., 2022) 中人類設計的指令開始，並要求「反向」模型提出符合缺失上下文的初始指令範例（圖 2 (底部)）。

3.2 評分函數

<style id='1'>To cast our problem as black-box optimization, we choose a score function that accurately measures the alignment between the dataset and the data the model generates. In our instruction induction experiments, we consider two potential score functions, described below. In the TruthfulQA experiments, we focused primarily on automated metrics proposed in Lin et al. (2022), similar to the execution accuracy. In each case, we evaluate the quality of a generated instruction using Equation (1), and take the expectation over a held-out test dataset $\mathcal{D}_{\text{test}}$.</style>為將我們的問題視為黑箱最佳化，我們選擇一個評分函數，以準確衡量資料集與模型生成資料之間的對齊程度。在我們的指令歸納實驗中，我們考慮了兩種潛在的評分函數，如下所述。在 TruthfulQA 實驗中，我們主要關注 Lin 等人 (2022) 提出的自動化指標，類似於執行準確度。在每個案例中，我們使用公式 (1) 評估生成指令的品質，並對保留的測試資料集 $\mathcal{D}_{\text{test}}$ 取期望值。

執行準確度 首先，我們考慮使用 Honovich 等人 (2022) 提出的執行準確度指標（我們將其表示為 f_{exec} ）來評估指令 ρ 的品質。在大多數情況下，

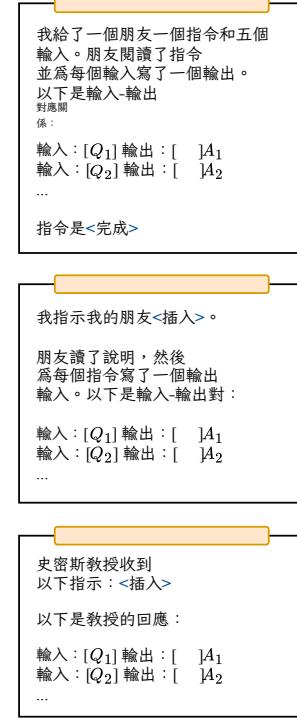


圖 2：LLM 的提示

執行準確度簡單定義為 0-1 損失， $f(\rho, Q, A) = \mathbb{1}[\mathcal{M}([\rho; Q]) = A]$ 。在某些任務中，執行準確度會考慮不變量；例如，它可能是順序不變的集合匹配損失，如 Honovich 等人 (2022) 附錄 A 中所述。

對數機率 我們進一步考慮一個更軟性的機率分數函數，我們假設該函數可能透過在搜尋低品質指令候選時提供更細微的訊號來改善最佳化。特別是，我們考慮在目標模型 \mathcal{M} 下，給定指令和問題的所需答案的對數機率，其在每個樣本的基礎上為 $\log P(A | [\rho; Q])$ 。

高效分數估計 透過計算所有指令候選在整個訓練資料集上的分數來估計分數可能很昂貴。為了降低計算成本，我們採用一種篩選方案，其中有潛力的候選會獲得更多計算資源，而低品質的候選則獲得較少計算。這可以透過在演算法 1 的第 2-9 行使用多階段計算策略來實現。我們首先使用訓練資料集的一小部分評估所有候選。對於分數大於特定閾值的候選，我們從訓練資料集中取樣並評估一個新的不重疊子集，以更新分數的移動平均值。然後，我們重複此過程，直到剩下少量候選，這些候選將在整個訓練資料集上進行評估。這種自適應篩選方案透過保持高品質樣本的精確計算成本並大幅降低低品質候選的計算成本，顯著提高了計算效率。我們注意到，類似的分數估計方案已在先前的工作中使用 (Li 等人，2022 年；Maclaurin 和 Adams，2015 年)。

3.3 疊代提案分佈

儘管我們嘗試直接取樣高品質的初始指令候選，但子區塊 3.1 中描述的此方法可能無法產生良好的提案集 \mathcal{U} ，原因可能是缺乏多樣性或不包含任何具有適當高分的候選。在遇到此類挑戰時，我們探索了重新取樣 \mathcal{U} 的疊代過程。

疊代蒙地卡羅搜尋 我們不只從初始提案中取樣，還考慮在目前最佳候選周圍局部探索搜尋空間。這使我們能夠產生更有可能成功的全新指令。我們將此變體稱為疊代 APE。在每個階段，我們評估一組指令並篩選出分數較低的候選。然後，要求 LLM 產生與高分指令相似的全新指令。我們在圖 3 中提供了用於重新取樣的提示。圖 6 (右) 顯示，儘管此方法提高了提案集 \mathcal{U} 的整體品質，但得分最高的指令傾向於保持不變

產生以下指令的變體，
遵循指示，同時
同時保留語義。
輸入：[指令]
輸出：`<完成>`

圖 3：重新取樣

多階段的結果相同。我們得出結論，迭代生成相較於第 3.1 小節所述生成過程的相對簡單性和有效性，僅提供了邊際改進。因此，除非另有說明，否則我們預設使用 APE 而不進行迭代搜尋。

4 大型語言模型是人類水準的提示工程師

本節探討 APE 如何引導大型語言模型實現預期行為。我們從四個角度進行調查：零樣本效能、少樣本情境學習效能、零樣本思維鏈推理和真實性。我們的實驗表明，APE 可以找到能提高任務效能的提示，其表現與人類編寫的提示相同甚至更好。APE 也經常產生關於如何最佳提示語言模型的深刻技巧，這些技巧可以成功地轉移到新任務中（參見第 4.3 節）。

4.1 指令歸納

我們評估了 Honovich 等人 (2022) 提出的 24 項指令歸納任務中零樣本和少樣本情境學習的有效性。這些任務涵蓋了語言理解的許多方面，從簡單的詞組結構到相似性和因果關係識別。我們在附錄 B 中提供了每項任務的詳細說明。對於每項任務，我們從訓練資料中抽取五個輸入-輸出對，並使用演算法 1 選擇最佳指令。然後，我們評估指令的品質

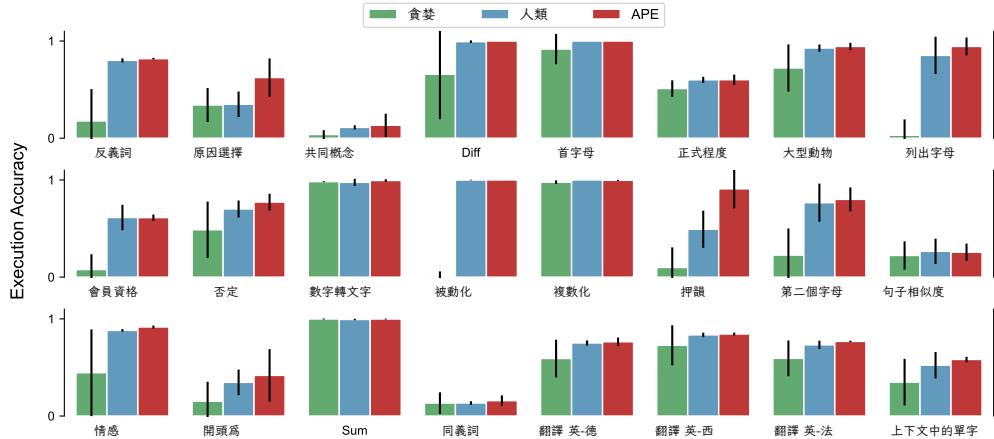


圖 4：24 項指令歸納任務的零樣本測試準確度。APE 在 24 項任務中，所有 24 項任務都達到人類水準或更佳的表現。

透過在 InstructGPT 3 上執行指令。我們重複實驗五次，使用不同的隨機種子來報告平均值和標準差。我們實驗的確切範本可以在附錄（表 5）中找到。

零樣本學習 我們將我們的方法與兩個基準進行比較：人類提示工程師 (Human)⁴ 和 Honovich 等人 (2022) 提出的模型生成指令演算法。此演算法可視為 APE 的貪婪版本，沒有搜尋和選擇過程；因此，我們將其稱為「Greedy」。圖 4 顯示 InstructGPT 使用人類指令和模型生成指令的零樣本效能。我們的演算法在每項任務上都優於「Greedy」，並在 24 項任務中的 24 項任務上達到與人類相同或更佳的效能。此外，圖 1 中所有 24 項任務的四分位數平均值 (IQM) (Agarwal 等人，2021) 表明，使用 InstructGPT 的 APE 優於人類工程提示，IQM 為 0.810，而人類為 0.749。我們在附錄中總結了 APE 為每項任務選擇的指令 (表 12)。

少樣本情境學習 我們評估了少樣本情境學習中 APE 生成的指令，我們將指令插入情境示範之前。這些指令是根據零樣本執行準確度選擇的，我們將此設定在圖 8 中表示為「指令 +情境」。如圖 8 所示，在 24 個任務中，有 21 個任務新增指令後，測試效能與標準情境學習效能相當或更佳。出乎意料的是，為 Rhymes、Large Animal 和 Second Letters 新增情境範例會損害模型效能。我們推測這可能是因為所選指令過度擬合零樣本學習情境，因此在少樣本情況下表現不佳。因此，我們實驗使用少樣本執行準確度作為選擇指標。圖 14 顯示，除了 Rhymes 之外，少樣本指標的表現與零樣本指標相當或略優。為了直觀地了解正在發生的事情，我們在附錄 C.1 中提供了定性分析。

4.2 BIGBENCH

為了驗證 APE 是否能應用於更具挑戰性的任務，我們提出並整理了 BIG-Bench 指令歸納 (BBII)，這是一個包含 21 項任務的簡潔且易於處理的子集，這些任務具有清晰、人工編寫的指令，可應用於資料集中的所有範例。所選任務涵蓋了語言理解的許多方面，並包括 BigBench-Hard 子集中的所有九個此類問題 (Suzgun et al., 2022)。特別是，它包括情感理解、無上下文問答、閱讀理解、摘要、演算法和各種推理任務（例如，算術、常識、符號和其他邏輯推理任務）。我們在附錄 B 中提供了任務的詳細描述和我們的選擇標準。

³我們透過 OpenAI API (<https://beta.openai.com/>) 使用 text-davinci-002。雖然 API 中沒有明確說明，但我們假設模型是 Ouyang 等人 (2022) 報告的模型。⁴我們使用 Honovich 等人 (2022) 的黃金註釋，這些註釋經過人工驗證以確保正確性。

對於每個任務，我們使用 InstructGPT 的反向模式生成來生成一組指令候選，並根據其執行準確性對指令進行排名。然後，我們在 InstructGPT 上執行選定的指令，以計算測試集上的零樣本效能，並將其與預設的人工提示進行比較。如附錄表 6 所示，APE 在 21 個任務中的 17 個任務上實現了與預設人工提示相當或更好的效能。

4.3 零樣本思維鏈

思維鏈推理已被證明能大幅提升大型語言模型完成複雜推理任務的能力，例如解決需要多個步驟的數學問題。早期關於思維鏈的研究（Nye 等人，2021 年；Betz 等人，2021 年；Wei 等人，2022b 年）使用微調或情境學習來讓大型語言模型展示其解決此類問題的過程。近期最具影響力的提示工程研究之一是發現（Kojima 等人，2022 年）只需在大型語言模型的回應開頭加上「讓我們一步一步思考。」，就能讓大型語言模型產生思維鏈。這種提示策略被稱為 Zero-Shot-CoT，它將 InstructGPT 在 MultiArith（Roy & Roth，2016 年）上的零樣本效能從 17.7 提升到 78.7，並將 GSM8K（Cobbe 等人，2021 年）上的效能從 10.4 提升到 40.7。如表 7 所示，Kojima 等人（2022 年）發現他們的提示在至少九個由人類設計的提示中表現最佳。

我們使用 APE 自動搜尋 Kojima 等人（2022）所用任務套件中的最佳答案前綴。我們最佳化此提示的方法靈感來自 Zelikman 等人（2022）。首先，我們使用 InstructGPT 產生一個問題和推理步驟的資料集，並加上「讓我們一步一步思考」。然後，我們移除所有答案不正確的資料點。最後，我們使用 APE 尋找一個以「讓我們」開頭的提示，以最大化這些正確推理步驟的可能性。提示產生和評估所用的範本請參閱表 5。APE 產生了提示「讓我們一步一步地解決這個問題，以確保我們得到正確的答案」。這個產生的提示將 MultiArith 的效能從 78.7 提升到 82.0，GSM8K 的效能從 40.7 提升到 43.0。我們相信這種通用工作流程代表了 APE 的常見用例，提示工程師使用 APE 來最佳化其現有範本的部分內容，以提高效能。有關此提示在其他推理任務上的效能詳細資訊，請參閱圖 10。

4.4 TRUTHFULQA

我們將方法應用於 TruthfulQA（Lin et al., 2022），以了解 APE 生成的指令如何引導大型語言模型生成不同風格的答案，並研究真實性與資訊性之間的權衡。借鑒原始論文中的指標，我們使用 APE 來學習最大化三個指標的指令：真實性（% True）、資訊性（% Info）以及兩者的組合（%True + % Info）。Lin et al. (2022) 使用人工評估來評估模型效能，但他們發現其自動化指標與人類預測的吻合度超過 90%。在我們的實驗中，我們依賴其微調的 GPT-judge 和 GPT-info 來評估分數。

TruthfulQA 中的提示工程我們想強調，TruthfulQA 資料集旨在測試零樣本設定中的預訓練模型。我們的結果與原始基準測試不相容。因為我們使用一小部分問答對作為訓練示範來最佳化指令，所以我們的結果不是「真正的少樣本學習」（Perez et al., 2021）。我們從 817 個問題中隨機抽取 100 個問題用於實際實驗，以形成訓練示範 $\mathcal{D}_{\text{train}}$ 。為了抽取提案集 \mathcal{U} ，我們要求「反向」模型根據六個隨機選擇的示範對生成指令，這與我們之前的實驗類似。與指令歸納不同，在 TruthfulQA 中，我們的目標是找到一個最佳指令提示，該提示適用於涵蓋健康、法律、政治和小說的 38 個問題類別。值得注意的是，我們生成的所有指令都非常通用，例如「您將被問一系列問題。對於每個問題，您必須回答問題或拒絕回答，在這種情況下，您必須說明您沒有評論」，並且不包含資料集中的任何範例。

真實性與資訊性之間的權衡 我們發現，APE 在 InstructGPT (175B) 提出的 200 個候選中，其表現優於人工設計的提示，如圖 5 所示。我們將我們生成的提示與 Lin 等人（2022）的「幫助」提示進行了比較。訓練和測試的表現如圖 5(a)-(b) 所示。我們發現，在訓練集中選擇 200 個候選中的前 10 個，在測試集中也能很好地推廣。我們報告了前 10 個指令在三個指標上的平均表現。這個結果本身並不令人驚訝，因為人類的基準線是

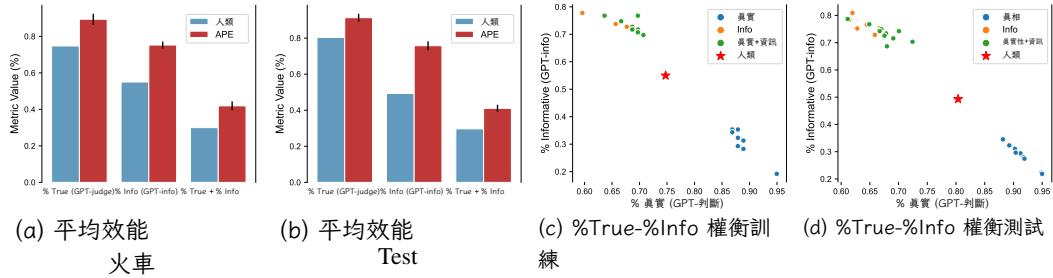


圖 5：APE 與「幫助」（人類）提示在 TruthfulQA 任務上的比較。(a) 100 個訓練範例中，答案為真實 (% True)、資訊豐富 (% Info) 或兩者皆是 (% True + % Info) 的百分比。(b) 717 個測試範例中的相同資料。(c) 使用每個指標的前 10 個指令，在訓練資料上計算的 %True-%Info 邊界。(d) 測試資料上的 %True-%Info 邊界。

並非經過仔細選擇，正如 Askell 等人 (2021) 所指出的。然而，我們發現 APE 發現的指令可以透過「不予置評」等答案實現非常高的真實性，但這些答案提供的資訊很少。我們使用我們的頂級候選來進一步研究真實性與資訊性之間的權衡。我們在圖 5(c) 和圖 5(d) 所示的真實性-資訊性圖中，視覺化了三個指標中前 10 個建議樣本。雖然 APE 在提供真實且資訊豐富的答案方面實現了超過 40% 的準確性（相較於人類「幫助」提示的 30%），但所發現的指令傾向於針對此 %true-%info 帕累托前沿的兩端。

5 定量分析

在本節中，我們進行了定量分析，以更好地理解我們方法的三個主要元件：提案分佈、評分函數和迭代搜尋。此外，我們在附錄 D 中進行了成本分析，以了解尋找最佳提示最具成本效益的方式。我們觀察到，儘管每個代幣的成本較高，但更大、功能更強大的語言模型在生成最佳提示方面更具成本效益。

5.1 用於提案分佈的 LLM

隨著模型大小的增加，提案品質如何變化？為了了解模型大小如何影響初始提案分佈的品質，我們檢查了透過 OpenAI API 提供的八種不同模型⁵。為了評估提案分佈的品質，我們為每個模型生成了 250 個指令，並計算了 50 個測試資料點的執行準確度。我們在圖 6(a) 中展示了簡單任務（即複數化）的存活函數（測試準確度大於特定閾值的指令百分比）和測試準確度直方圖，並在附錄中包含了更具挑戰性任務（以…開頭）的類似圖表（圖 28）。如兩圖所示（且不足為奇），較大的模型往往比小型模型產生更好的提案分佈，經過微調以遵循人類指令的模型也是如此。在簡單任務中，最佳模型 InstructGPT (175B) 生成的所有指令都具有合理的測試準確度。相比之下，一半的指令偏離主題，在更具挑戰性的任務中表現不佳。

5.2 用於選擇的 LLM

提案品質在選擇下是否重要？如果我們從 LLM 中採樣更多指令，那麼我們就更有可能找到更好的指令。為了驗證這個假設，我們將樣本大小從 4 增加到 128，並評估測試準確度的變化。圖 7（左）顯示了單調遞增的趨勢，且報酬遞減，因為在 64 個指令樣本下即可達到人類水準的效能。因此，我們選擇 50 作為預設樣本大小。在此設定下，我們研究提案分佈如何影響我們演算法所選最佳指令的測試準確度。圖 1(b) 顯示，儘管小型模型產生良好指令的可能性較低，但如果我們採樣足夠的候選指令，它們仍然會產生一些好的指令。因此，

⁵我們使用 ada、babbage、curie、davinci、text-ada-001、text-babbage-001、text-curie-001、text-davinci-002

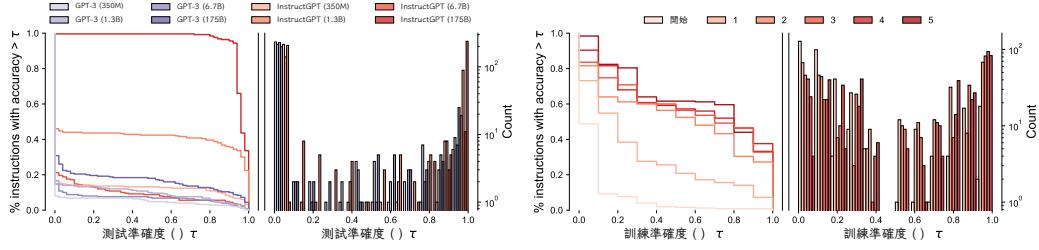


圖 6：(左) 具有不同大小的模型之提案分佈品質，由測試執行準確度評估。(右) 疊代蒙地卡羅搜尋在每一輪改進了指令候選的品質。

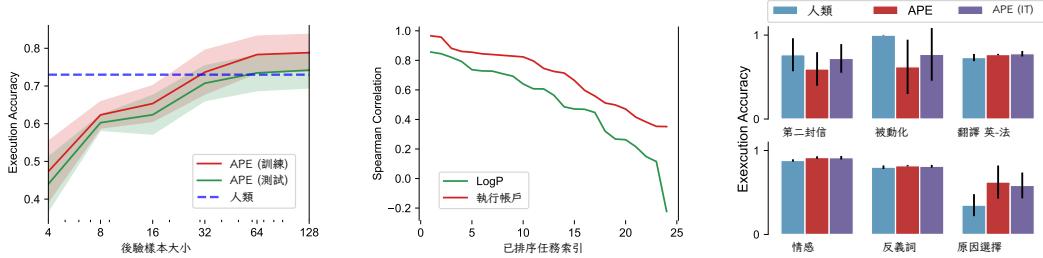


圖 7：(左) 隨著指令候選數量增加，最佳指令的測試執行。我們報告了 6 個不同任務的平均值和標準差。(中) 24 個任務中測試準確度與兩個指標之間的 Spearman 相關性。(右) 使用 APE 和迭代 APE (APE (IT)) 選擇的最佳指令的測試執行準確度。

我們仍然可以透過執行我們的選擇演算法，用小型模型找到有前景的指令，這解釋了為什麼我們的方法在所有八個模型中都優於 Honovich 等人 (2022) 的貪婪方法。

哪個評分函數更好？我們計算了 24 個指令歸納任務中測試準確度與兩個指標之間的相關性，以研究我們提出的指標有多好。我們使用 InstructGPT (175B) 以「前向」模式為每個任務生成 250 個指令，並計算 10 個測試數據點的指標分數和測試準確度。我們將測試準確度與兩個指標之間的 Spearman 相關性視覺化。圖 7 (中) 顯示執行準確度在各任務中與測試效能更吻合。因此，除非另有說明，我們將其選為預設指標。

5.3 迭代蒙地卡羅搜尋

疊代搜尋是否能提升指令品質？我們在圖 6 (右) 中視覺化了「被動化」任務的測試準確度存活函數和直方圖，並在附錄中納入了另外五個任務。存活圖顯示，曲線隨著回合數的增加而上升，這表明疊代搜尋確實會產生更高品質的提案集。然而，我們觀察到進一步的選擇回合會產生遞減的回報，因為品質在三回合後似乎趨於穩定。

我們需要疊代搜尋嗎？我們比較了六個任務上的 APE 和疊代 APE。如圖 7 所示，疊代搜尋在 APE 表現不如人類的任務上略微提高了效能，但在其他任務上則達到了相似的效能。這與我們的假設一致，即疊代搜尋在生成良好的初始 \mathcal{U} 具有挑戰性的任務上最有用。

6 結論

大型語言模型可以被視為通用電腦，執行由自然語言提示指定的程式。我們將提示工程流程自動化，將其表述為黑箱最佳化問題，並建議使用由大型語言模型引導的高效率搜尋演算法來解決。我們的方法在各種任務上以最少的人工輸入達到人類水準的效能。由於近期的大型語言模型展現出令人印象深刻的遵循人類指令的能力，我們預期許多未來的模型，包括那些用於正式程式合成的模型，都將具有自然語言介面。這項工作為控制和引導生成式人工智慧奠定了基礎。

致謝

我們要感謝 Or Honovich 和 Michael Zhang 的幫助和寶貴意見。JB 獲得了 NSERC 補助金 [2020-06904], CIFAR AI 主席計畫、Google 研究學者計畫和 Amazon 研究獎的資助。KP 獲得了 NSERC PGS-D 的資助。SP 獲得了 NSERC CGS-D 的資助。HC 獲得了 NSERC CGS-D 和 RBC 研究生獎學金的資助。用於準備本研究的資源部分由安大略省、加拿大政府（透過 CIFAR）以及贊助 Vector Institute for Artificial Intelligence 的公司提供。

參考文獻

- Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.
- Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.
- Eyal Ben-David, Nadav Oved, and Roi Reichart. Pada: A prompt-based autoregressive approach for adaptation to unseen domains. *arXiv preprint arXiv:2102.12206*, 2021.
- Gregor Betz, Kyle Richardson, and Christian Voigt. Thinking aloud: Dynamic context generation improves zero-shot reasoning performance of gpt-2. *arXiv preprint arXiv:2103.13033*, 2021.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 1173–1178, 2019.
- Jacob Devlin, Rudy R Bunel, Rishabh Singh, Matthew Hausknecht, and Pushmeet Kohli. Neural program meta-induction. *Advances in Neural Information Processing Systems*, 30, 2017.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 320–335, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.26. URL <https://aclanthology.org/2022.acl-long.26>.
- Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines for neurally-guided bayesian program induction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/file/7aa685b3b1dc1d6780bf36f7340078c9-Paper.pdf>.
- Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN international conference on programming language design and implementation*, pp. 835–850, 2021.
- Tianyu Gao. Prompting: Better ways of using language models for nlp tasks. *The Gradient*, 2021.
- Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL <https://aclanthology.org/2021.acl-long.295>.
- Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.
- Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782*, 2022.
- Naman Jain, Skanda Vaidyanath, Arun Iyer, Nagarajan Natarajan, Suresh Parthasarathy, Sriram Rajamani, and Rahul Sharma. Jigsaw: Large language models meet program synthesis. In *Proceedings of the 44th International Conference on Software Engineering*, pp. 1219–1231, 2022.
- Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.
- Woosuk Lee, Kihong Heo, Rajeev Alur, and Mayur Naik. Accelerating search-based program synthesis using learned probabilistic models. *ACM SIGPLAN Notices*, 53(4):436–449, 2018.
- Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittweiser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*, 2022.
- Percy Liang, Michael I. Jordan, and Dan Klein. Learning programs: A hierarchical bayesian approach. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 21–24, 2010, Haifa, Israel, pp. 639–646. Omnipress, 2010. URL <https://icml.cc/Conferences/2010/papers/568.pdf>.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL <https://aclanthology.org/2022.acl-long.229>.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Dougal Maclaurin and Ryan Prescott Adams. Firefly monte carlo: Exact mcmc with subsets of data. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Aditya Menon, Omer Tamuz, Sumit Gulwani, Butler Lampson, and Adam Kalai. A machine learning framework for programming by example. In *International Conference on Machine Learning*, pp. 187–195. PMLR, 2013.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070, 2021.

Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5203–5212, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, 2021.

- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.
- Mirac Suzgun, Nathan Scales, Nathanael Schärlí, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*, 2021.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.
- Catherine Wong, Kevin M Ellis, Joshua Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. In *International Conference on Machine Learning*, pp. 11193–11204. PMLR, 2021.
- Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.
- Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.
- Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuhui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

野外提示工程

近年來，具有自然語言介面的大型模型，包括用於文字生成和圖像合成的模型，其公共使用量不斷增加。由於人類很難找到正確的提示，因此已經開發了許多關於提示工程的指南以及輔助提示發現的工具。例如，請參閱：

- <https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/>
- <https://techcrunch.com/2022/07/29/a-startup-is-charging-1-99-for-strings-of-text-to-feed-to-dall-e-2/>
- <https://news.ycombinator.com/item?id=32943224>
- <https://promptomania.com/stable-diffusion-prompt-builder/>
- <https://huggingface.co/spaces/Gustavosta/MagicPrompt-Stable-Diffusion>

在本文中，我們應用 APE 來產生有效指令以引導 LLM，但只要能設計出適當的提案方法和評分函數，通用框架演算法 1 即可應用於引導其他具有自然語言介面的模型。

B 實作細節

表 1：Honovich 等人 (2022) 提出的 24 個指令歸納任務的詳細說明。為方便起見，Honovich 等人 (2022) 的原始表格在此重複。

類別	Task	指令	示範
拼字	第一個字母	擷取輸入單字的第一個字母。	cat → c
	第二個字母	擷取輸入單字的第二個字母。	cat → a
	列出字母	將輸入單字拆解成字母，並以空格分隔。	cat → c a t
詞法句法	開頭爲	從輸入句子中擷取以指定字母開頭的單字。	上週我撞到車的那個人告了我。 [m] → 人，我
	複數化	將輸入詞轉換爲其複數形式。	貓 → 貓
語法	被動化	將輸入句寫成被動語態。	藝術家介紹了科學家 → 科學家是由藝術家介紹的。
	否定	否定輸入句。	時間是有限的 → 時間是無限的。
詞彙語義學	反義詞	寫一個與輸入詞意思相反的詞。	贏 → 輸
	同義詞	寫一個與輸入詞語意思相近的詞語。	據稱 → 假定
	會員資格	寫出給定清單中出現的所有動物。	貓、直升機、廚師、鯨魚、青蛙，獅子 → 青蛙、貓、獅子、鯨魚
語音學	押韻	寫一個與輸入單字押韻的單字。	sing → ring
知識	較大的動物	寫出兩種給定動物中較大的一種。	無尾熊、蝸牛 → 無尾熊
	原因選擇	找出兩個給定的因果句中哪個是原因。	句子 1：汽水沒氣了 句子 2：瓶子是開著的。→ 瓶子是開著的。
語義	共同概念	找出給定物件的共同特徵。	吉他、擺錘、微中子 → 涉及振盪。
	風格	請用正式語言改寫句子。	請您抵達後立即致電 → 請您抵達後致電。
數值	Sum	將兩個給定數字相加。	22 10 → 32
	差異	用第一個數字減去第二個數字。	32 22 → 10
	數字轉文字	用英文單字寫出數字。	26 → 二十六
多語言		將單字翻譯成德文/西班牙文/法文。	遊戲 → juego
GLUE	情感分析	判斷電影評論是正面還是負面。	這部電影規模雖小，但形式完美。→ 正面
	句子相似度	請以 0 (絕對不相似) 到 5 (完全相似) 的等級，評估兩個輸入句子的語義相似度。	句子 1：一個男人正在抽菸。 句子 2：一個男人正在溜冰。→ 0 - 絕對不相似
	上下文中的詞彙	判斷輸入詞彙在兩個輸入句子中是否具有相同意義。	句 1：處理一項任務。句 2：接近城市 詞彙：approach → 不同

表 2：BIG-Bench 指令歸納 (BBII) 的詳細描述，這是 21 個任務的乾淨且易於處理的子集，這些任務具有清晰的人工編寫指令，可應用於資料集中的所有範例。

Name	描述	關鍵字
因果判斷	回答有關因果歸因的問題	因果推理、常識、多選、閱讀理解、社會推理
消歧問答	釐清含糊代名詞句子的意義	常識、性別偏見、多樣本、多選
迪克語言	正確關閉 Dyck-n 詞	代數、算術、邏輯推理、多重選擇
認知推理	判斷一個句子是否蘊含下一個句子	常識、邏輯推理、多重選擇、社會推理、心智理論
德語性別包容句	給定一個未使用性別包容形式的德語句子，將其轉換為性別包容形式	自由回應、文法、包含、非英文、意譯
含意	預測說話者 2 對說話者 1 的回應是算「是」還是算「否」	情境問答、多選題、閱讀理解、社會推理、心智理論
語言學謎題	解決羅塞塔石碑式的語言學謎題	自由回應、類人行為、語言學、邏輯推理、閱讀理解
邏輯謬誤偵測	偵測非正式和正式的邏輯謬誤	邏輯推理、多重選擇
電影推薦	推薦與給定電影清單相似的電影	情商，多選
導航	給定一系列導航指令，判斷是否會回到起點	算術、邏輯推理、數學、多選
物件計數	涉及列舉不同類型物件並要求模型計數的問題	自由回應、邏輯推理
運算子	給定自然語言的數學運算子定義，套用它	自由回應、數學、數值回應
作為 NLI 的預設	判斷第一個句子是否蘊含或矛盾第二個句子	常識、邏輯推理、多重選擇
問題選擇	給定一個簡答及其上下文，選擇最適合該簡答的問題	多重選擇、釋義、閱讀理解、摘要
毀壞名稱	選擇一個幽默的編輯，以「毀壞」輸入的電影或音樂藝術家名稱	情感理解，多選
諷刺	判斷兩句話中哪一句是諷刺的	情感理解、幽默、多選
運動理解	判斷一句人工建構的運動相關句子是合理還是不合理	常識、無上下文問題回答、領域特定、多選
時態	修改給定句子的時態	自由回應、釋義、語法
winowhy	評估回答 Wino-grad Schema Challenge 問題的推理	因果推理、常識、多重選擇、社會推理
單字排序	排序單字列表	演算法、自由回應
單字重組	重組給定的字母以形成一個英文單字	自由回應、隱含推理、權杖化

B.1 BIG-BENCH 指令歸納 (BBII) 選擇過程

步驟 1：BIG-Bench 包含大量不同品質等級的評估任務。例如，有些任務僅有符合提交資格所需的最少範例數，而其他任務可能缺乏適當的人類基準。因此，我們遵循 Suzgun 等人 (2022) 的方法，根據以下標準獲得一個乾淨且易於處理的子集。

表 3：用於建立 BIG-Bench 指令歸納 (BBII) 子集的篩選標準。

任務數	標準
212	所有 BIG-Bench 任務
170	所有 JSON 任務
127	篩選掉多於一個子任務的任務後
74	篩選掉少於 150 個範例的任務後
67	篩選掉沒有人類評分者基準的任務後
57	篩選掉未使用多選或精確匹配作為評估指標的任務後

標準：**JSON** 任務。

已捨棄的任務：抽象與推理語料庫、bbq lite、機率偏差、布林表達式、com2sense、上下文定義對齊、convinceme、coqa 對話式問答、循環字母、多元社會偏見、動態計數、摘要事實性、預測子問題、中文性別敏感度、英文性別敏感度、高低遊戲、長上下文整合、多步驟算術、穆斯林暴力偏見、程式合成、蛋白質交互位點、Python 程式設計挑戰、問題答案建立、根優化與遊戲、自我意識、法庭自我評估、輔導自我評估、簡單算術、拼字比賽、小隊輪班、主謂一致、數獨、禁忌、talkdown、文字導航遊戲、測試集訓練、真實問答、二十個問題、uncover、謊言之網、集合與圖形文字問題、是或否黑白。

標準：無子任務的任務。

已捨棄的任務：抽象敘事理解、算術、作者驗證、bbq lite json、因果關係、西洋棋狀態追蹤、cifar10 分類、顏色、概念組合、人造語言翻譯、CS 演算法、初等數學問答、事實查核器、gem、目標步驟 wikihow、hhh 對齊、印度因果關係、相交幾何、漢字 ascii、鍵值對應、語言遊戲、語言映射、listfunctions、邏輯推論、隱喻理解、迷你謎團問答、修改算術、多重資料整理、multiemo、自然指令、週期元素、物理、真實或虛假文本、simp 圖靈概念、簡單算術 json 子任務、簡單倫理問題、奇怪故事、符號解釋、追蹤隨機物件、撤銷排列、單位轉換、單位解釋、不自然情境學習。

標準：任務包含至少 **150** 個輸入-輸出配對範例。

已捨棄的任務：分析蘊涵、自動除錯、程式碼行描述、代號、共同詞素、崩潰之花、粗俗人工智能、低溫生物學西班牙語、黑色幽默偵測、表情符號電影、表情符號情緒預測、經驗判斷、英語諺語、英語俄語諺語、蘊涵極性、蘊涵極性印地語、評估資訊必要性、修辭格偵測、一般知識、GRE 閱讀理解、人體器官感官、識別數學定理、識別奇異隱喻、隱含關係、國際音標 NLI、諷刺識別、已知未知、邏輯參數、邏輯序列、數學歸納法、俄語誤解、無意義詞語語法、新穎概念、格格不入、表格中的企鵝、波斯語習語、短語相關性、物理直覺、物理問題、重複複製邏輯、改寫、謎語感、科學新聞稿、句子歧義、相似性抽象、簡單算術 JSON、簡單算術 JSON 多選、簡單算術多目標 JSON、簡單文字編輯、足夠資訊、自殺風險、瑞典語到德語諺語、何謂道。

標準：任務包含報告的（平均）人類評分者或隨機效能。

已捨棄的任務：上下文參數知識衝突、印地語英語毒性、俄語醫學問題、Parsinlu QA、斯瓦希里語英語諺語、TellMeWhy、Which Wiki Edit。

標準：任務為分類或使用精確比對作為評估指標。

已捨棄的任務：自動分類、少樣本 NLG、印地語問答、國際音標轉寫、波蘭語序列標註、QA Wikidata、上下文語義解析 SPARC、語義解析 Spider、社交支援、主題聊天。

步驟 2：我們進行人工檢查，將剩餘任務分為以下三類。特別是，Big-Bench 指令歸納 (BBII) 子集是我們在第 4.2 節中用於評估 APE 的子集。

- BBII 子集：Big Bench 任務的子集，符合指令歸納格式：資料集中的每個範例都可以表示為問答對，所有範例都專注於可以由人類指令清楚描述的相同問題，並且任務 JSON 檔案中有人類指令可用。
- 無效格式：不符合指令歸納格式的任務：資料集中的每個範例都提出不同的問題，或者沒有明確的人類指令可用。
- 超出範圍：超出本工作範圍的任務：作者無法在 60 分鐘內解決，或需要專業知識。

表 4：用於建立 BIG-Bench 指令歸納 (BBII) 子集的篩選標準。

# 類別	# 任務	任務名稱
BBII 子集	21	因果判斷、消歧義問答、迪克語言、認知推理、德語性別包容句、含義、語言學謎題、邏輯謬誤檢測、電影推薦、導航、物件計數、運算子、作為 NLI 的預設、問題選擇、毀滅名稱、諷刺、運動理解、時態、winowhy、單字排序、單字重組。
無效格式	21	時代錯誤、類比相似性、橋接回指解析 BARQA、資料理解、Disfl QA、奇幻推理、形式謬誤三段論否定、印度知識、倒裝、意圖識別、邏輯網格謎題、段落分割、播放對話相同或不同、關於彩色物件的推理、顯著翻譯錯誤檢測、社交 IQA、StrategyQA、時間序列、Timedial、理解寓言、維生素事實驗證。
超出範圍	13	ASCII 文字辨識、一步將死、中國剩餘定理、氪石、語篇標記預測、幾何形狀、卡納達語、語言辨識、矩陣形狀、MNIST ASCII、道德許可性、電影對話相同或不同、波斯語閱讀理解。

表 5：我們實驗中用於模型提示的原始範本

用法	範本
零樣本評估	<p>指示：[指示]輸入：[]\n輸出： <完成> Q_{test}</p>
少樣本評估	<p>指示：[指示]輸入：[]\n輸出：[]\n\n輸入：[] \n輸出 Q_1[]... 輸入 A_1[]\n輸出 Q_2 <完成> A_2 Q_{test}</p>
正向生成	<p>我給了一個朋友一個指示和五個輸入。朋友閱讀了指示，並為每個輸入寫了一個輸出。 \n以下是輸入-輸出對：輸入：[]\n輸出：[]\n\n輸入： []\n輸出：[] ... 指示是<完成> Q_2 A_2</p>
反向生成 1	<p>我指示我的朋友<插入>。朋友閱讀了指示，並為每個輸入寫了一個輸出。 \n以下是輸入-輸出對：輸入：[]\n輸出：[]\n\n輸入：[]\n輸出： []... Q_1 A_1 Q_2 A_2</p>
反向生成 2	<p>史密斯教授收到了以下指示：<插入>\n以下是教授的回應：問：[]\n答：[]\n\n問：[]\n答：[] ... Q_1 A_1 Q_2 A_2</p>
重新取樣指令	<p>產生以下指令的變體，同時保留語義。輸入：[INSTRUCTION]\n輸出： <COMPLETE></p>
Zero-shot-CoT	<p>指令：回答以下問題。問：[INPUT]\n答： 讓我們<INSERT>。[OUTPUT]</p>

C 額外結果

C.1 指令歸納

少樣本情境學習 我們在少樣本情境學習中評估了 APE 生成的指令，其中我們在情境示範之前插入指令。這些指令是根據零樣本執行準確性選擇的，我們將此設定在圖 8 中表示為「指令 + 情境」。如圖 8 所示，在 24 個任務中的 21 個任務上，添加指令比標準情境學習性能實現了相當或更好的測試性能。出乎意料的是，為 Rhymes、Large Animal 和 Second Letters 添加情境範例會損害模型性能。我們推測這可能是因為所選指令過度擬合零樣本學習場景，因此在少樣本情況下表現不佳。因此，我們嘗試使用少樣本執行準確性作為選擇指標。圖 14 顯示，除了 Rhymes 之外，少樣本指標與零樣本指標相比，實現了相當或略好的結果。為了直觀地了解正在發生的事情，我們在下面提供了定性分析。

少樣本定性分析 我們發現，當結合指令和情境提示時，Rhymes 上存在一個對抗性案例。表 8 顯示，5 個過濾後的指令中有 4 個要求回應輸入詞。這些提議有效地以近乎完美的測試準確度來駭入評估，因為每個詞都與其本身押韻。然而，為這些指令添加情境範例會在指令（導致微不足道的押韻）和情境（導致非微不足道的押韻）之間產生不一致，導致效能顯著下降。如果我們改為根據少樣本指標來評分指令，則可以緩解這種效能下降，因為模型可以選擇更一致的指令。

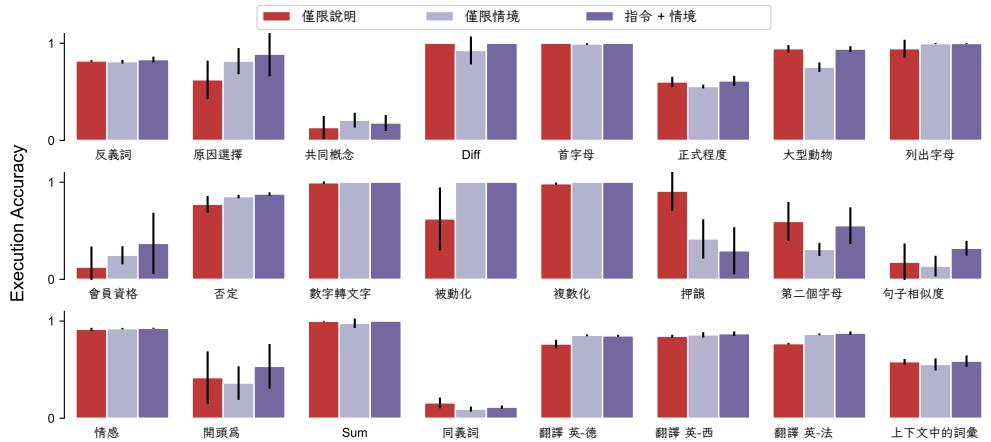


圖 8：24 項指令歸納任務的少樣本情境測試準確度。APE 提升了 24 項任務中 21 項的少樣本情境學習效能。

C.2 BIG-BENCH 指令歸納

我們使用 APE 為 BIG-Bench 指令歸納 (BBII) 中的任務生成新的提示。與人類提示相比，APE 生成的提示在 21 個任務中的 17 個任務上改善或匹配了零樣本效能。我們報告了 Srivastava 等人 (2022) 中定義的標準化偏好指標。在此指標下，100 分對應於人類專家效能，0 分對應於隨機猜測。請注意，如果模型在多選任務上的表現比隨機猜測差，則其分數可能低於 0。

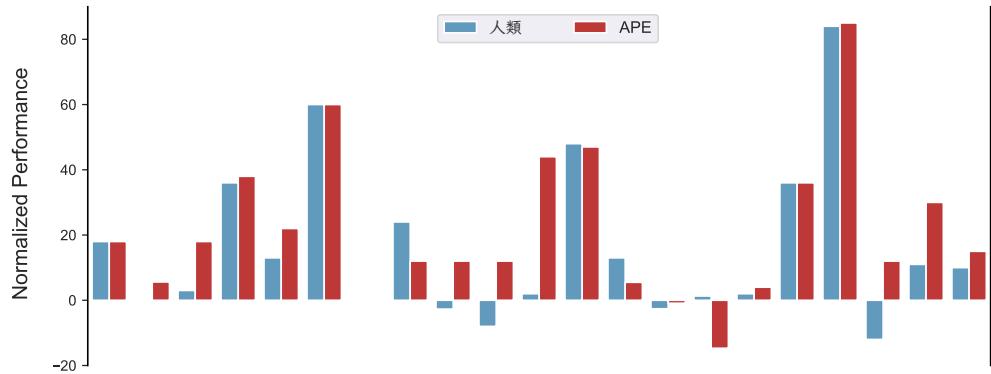


圖 9：APE 改善或匹配 21 個 BIG-Bench 指令歸納任務中 17 個任務的正規化零樣本效能。

表 6：21 個 BIG-Bench 指令歸納任務的零樣本正規化測試效能。APE 改善或匹配 21 個任務中 17 個任務的效能。

Task	標準化效能	
	人類	APE
因果判斷	18.0	18.0
消歧義問答	-0.4	5.6
迪克語言	3.0	18.0
認知推理	36.0	38.0
德語性別包容句	13.0	22.0
隱含意義	60.0	60.0
語言學難題	0.0	0.0
邏輯謬誤偵測	24.0	12.0
電影推薦	-2.7	12.0
導覽	-8.0	12.0
物件計數	2.0	44.0
運算子	48.0	47.0
作為 NLI 的預設	13.0	5.5
問題選擇	-2.6	-0.9
毀壞名稱	1.3	-14.7
諷刺	2.0	4.0
運動理解	36.0	36.0
緊張	84.0	85.0
winowhy	-12.0	12.0
單字排序	11.0	30.0
單字重組	10.0	15.0

C.3 零樣本思維鏈推理

我們使用 APE 來發現比 Kojima 等人 (2022) 的「讓我們一步一步思考」更好的思維鏈 (CoT) 提示。APE 找到了一個通用提示「讓我們一步一步地解決這個問題，以確保我們得到正確的答案」，與原始 CoT 提示相比，它能夠將 text-davinci-002 在 MultiArith Roy & Roth (2016) 上的零樣本 CoT 性能從 78.7 提高到 82.0，並將 GSM8K Cobbe 等人 (2021) 從 40.7 提高到 43.0。我們在圖 10 中包含了使用這個新的 APE CoT 提示在 12 個任務上的完整結果。

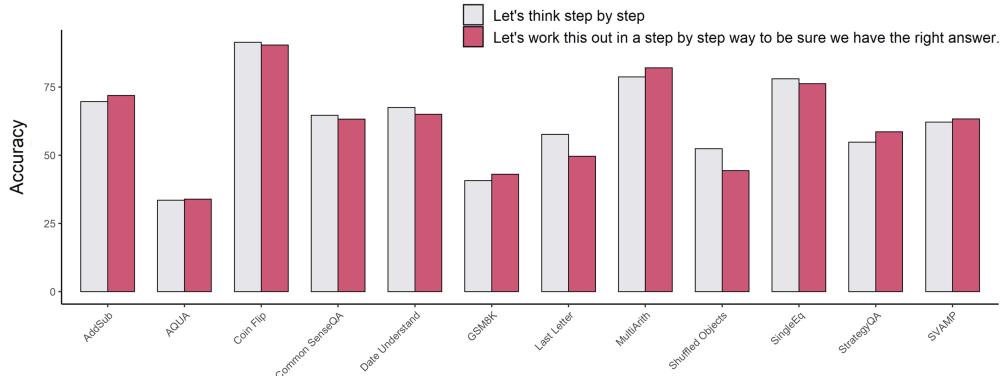


圖 10：APE 發現的提示「讓我們一步一步地解決這個問題，以確保我們得到正確的答案」在 Kojima 等人 (2022) 的 12 項任務上的表現。我們從原始論文中收集了一個 CoT 資料集，並篩選掉不正確的答案。然後我們使用 APE 來優化 CoT 提示。我們在 12 項任務中有 6 項任務的表現有所提升，並且在 12 項任務中有 4 項任務的表現幾乎與人類表現持平。我們假設 Shuffled Objects 和 Last Letter 難以透過通用提示進行優化。

表 7：在 MultiArith (Roy & Roth, 2016) 資料集上使用 InstructGPT (text-davinci-002) 的零樣本思維鏈表現。範本 (*1) 由 Kojima 等人 (2022) 提出，用於啟用大型語言模型的零樣本思維鏈推理，而範本 (*2) 和 (*3) 分別用於 Ahn 等人 (2022) 和 Reynolds & McDonell (2021)。

No.	類別	零樣本 CoT 觸發提示	準確度
1	APE	讓我們一步一步地解決這個問題，以確保我們得到正確的答案。	82.0
2	人工設計	讓我們一步一步地思考。(*1)	78.7
3		首先，(*2)	77.3
4		讓我們從邏輯上思考這個問題。	74.5
5		讓我們將這個問題拆解成幾個步驟來解決。(*3)	72.2
6		讓我們務實地一步一步思考。	70.8
7		讓我們像偵探一樣一步一步思考。	70.3
8		讓我們思考	57.5
9		在我們深入探討答案之前，	55.7
10		答案在證明之後。	45.7
-		(零樣本)	17.7

C.4 定量分析

我們可以使用其他 LLM 來提出指令嗎？我們研究了其他 LLM 的指令生成，包括具有前向生成能力的 LLM (OPT-175B (Zhang et al., 2022)、OpenAI Codex (Chen et al., 2021)) 和具有反向生成能力的 LLM (INT4 量化 GLM-130B (Zeng et al., 2022))。我們評估了它們在從指令歸納中選擇的六個任務上的性能，包括零樣本和少樣本設定 6。圖 15 和 16 顯示，InstructGPT 取得了最佳性能，除了被動語態，在該任務中它表現不如其他兩個前向生成模型。有趣的是，儘管 Codex 和 OPT 的指令建議模型與 InstructGPT 評分模型不同，但它們的性能幾乎與 InstructGPT 相當。然而，我們觀察到 OPT 生成的一些指令包含上下文範例（表 13），這使得它們更接近少樣本而非零樣本。相比之下，GLM 取得了最差的零樣本性能，因為它的填充能力經過訓練可以生成非常短的文本，如表 15 所示。

Meta 提示詞有多重要？在我們的實驗中，我們觀察到用於指令生成的 Meta 提示詞可以顯著影響所提出指令的分佈。為了探討它如何影響最終性能，我們使用我們的 TruthfulQA 範本而不是反向生成範本進行實驗（圖 21、22）。我們發現 Meta 提示詞範本會產生影響，在某些任務上提高了性能，同時損害了其他任務。值得注意的是，成員資格的準確性可以超越正向生成的指令，而原始範本無法提出好的指令。我們將 Meta 提示詞工程的探索留待未來工作，以獲得更好的提案分佈。

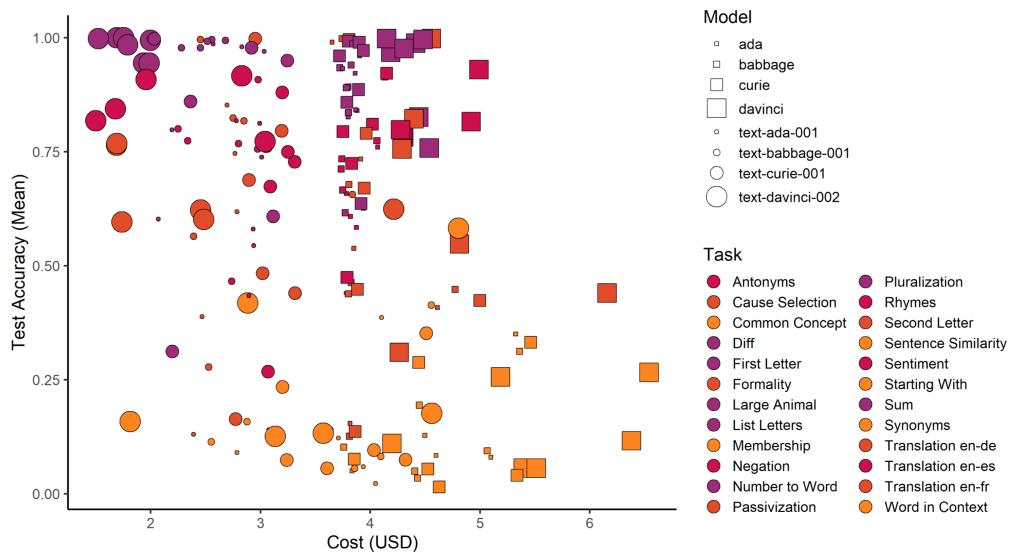
所產生指令的可轉移性如何？我們調查 APE 是否可用於引導未參與指令產生和選擇過程的模型。如圖 17 所示，當我們使用來自 InstructGPT 的指令來引導 GPT-3 模型時，以及反之亦然時，效能會顯著下降。這種效能下降可以透過人工編寫的指令來緩解。這表明評分模型和執行模型之間的對齊至關重要，並且 InstructGPT 產生的指令最適用於 InstructGPT 本身，但無法很好地轉移到像 GPT-3 這樣的不同模型。相比之下，GPT-3 產生的指令可以非常出色地引導 GPT-3，大大優於 InstructGPT 指令和人工指令。儘管 GPT-3 無法很好地遵循人工指令，但我們證明它仍然可以產生非常適合自己的提示，儘管這些提示不直觀，但仍能產生所需的行為。我們在表 16 中提供了產生的提示。

⁶這六項任務的選擇方式是，其中兩項比人類差，另外四項達到人類水平。它們涵蓋六個類別（拼寫、詞法句法、詞彙語義、語義、多語言和 GLUE）。

D 成本分析

更強大的模型在指令提案方面具有成本效益。儘管每個代幣的成本更高，但我們發現更大、與人類對齊的模型（經過訓練以遵循人類指令的模型 (Ouyang et al., 2022)）在 APE 的準確性成本邊界上佔據主導地位（圖 11）。與未經人類指令微調的較小模型相比，它們傾向於產生更簡潔的指令（圖 12），顯著降低了 APE 評分的成本。因此，我們建議盡可能使用更大且與人類對齊的指令產生模型。

APE 指令是上下文濃縮器。儘管零樣本指令需要比上下文學習更廣泛的離線取樣和評分，但當攤銷到大量推論時，它們在代幣方面是高效的。從這個角度來看，我們將 APE 的成本視為一次性開銷，用於從示範中提煉出簡潔的提示。如圖 13 所示，與上下文學習相比，APE 指令將提示代幣的數量減少了多達一個數量級。未來探索最佳化提示長度的工作可以進一步降低與引導 LLM 相關的成本。



<style id='1'>Figure 11: The accuracy-cost frontier of APE across eight OpenAI models. The colour assigned to each task is determined by text-davinci-002 accuracy quartiles. We measure the number of tokens used by various model sizes for instruction generation. We also measure the number of tokens used to score 250 generated instructions on ten validation input-output pairs on InstructGPT (i.e., text-davinci-002). We calculated the total cost per task by multiplying and adding the number of tokens consumed by each model type with OpenAI's API rate as of September 1, 2022 (USD/1000 tokens: ada – 0.0004, babbage – 0.0005, curie – 0.0020, davinci – 0.0200). Counter-intuitively, smaller models are more expensive. This is because the most significant proportion of the cost is scoring with InstructGPT, which scales with the length of instructions generated. Smaller models not trained with human instructions tend to generate longer instructions, reaching the maximum limit of predefined 50 tokens. Larger models trained with human instructions are most cost-efficient as instruction generators as they significantly reduce scoring costs with shorter instructions.</style>圖 11：APE 在八個 OpenAI 模型中的準確度成本曲線。分配給每個任務的顏色由 text-davinci-002 準確度四分位數決定。我們測量了各種模型大小用於指令生成的權杖數量。我們還測量了在 InstructGPT (即 text-davinci-002) 上，用於評分 250 個生成指令在十個驗證輸入輸出對上的權杖數量。我們透過將每種模型類型消耗的權杖數量與截至 2022 年 9 月 1 日 OpenAI 的 API 費率 (美元/1000 權杖: ada – 0.0004、babbage – 0.0005、curie – 0.0020、davinci – 0.0200) 相乘和相加來計算每個任務的總成本。出乎意料的是，較小的模型更昂貴。這是因為成本的最大部分是使用 InstructGPT 進行評分，而這會隨著生成指令的長度而擴展。未經人類指令訓練的較小模型傾向於生成較長的指令，達到預定義的 50 個權杖的最大限制。經過人類指令訓練的較大模型作為指令生成器最具成本效益，因為它們透過較短的指令顯著降低了評分成本。

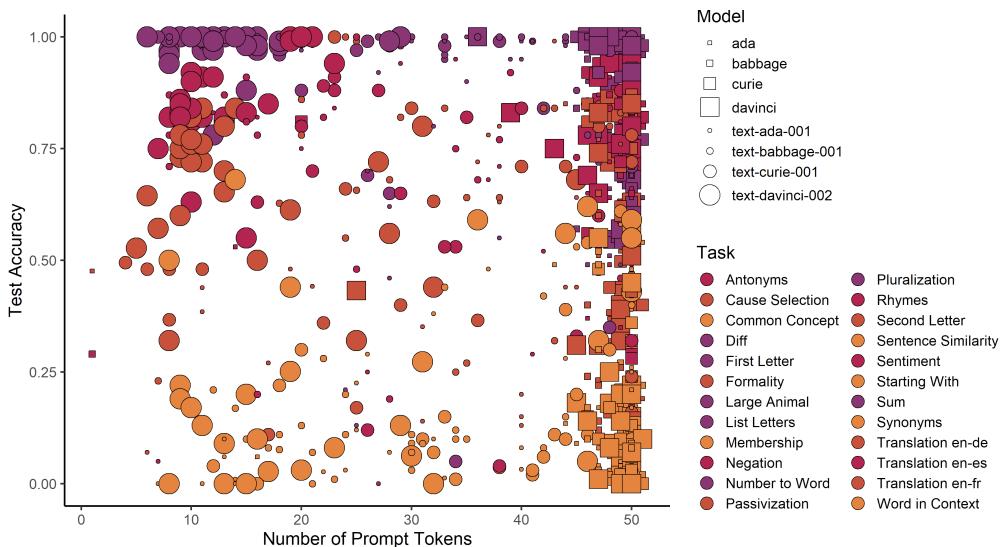


圖 12：在八個 OpenAI 模型和 24 個 NLP 任務中生成的提示的準確度-長度邊界。未經人類指令訓練的模型傾向於達到我們允許生成的預定義最大代幣數，而更大、更對齊的 LLM 則輸出更簡潔的指令。能力更強的 LLM 在指令長度和準確度的邊界上佔據主導地位，我們認為這是一種有效將上下文濃縮為指令的能力。

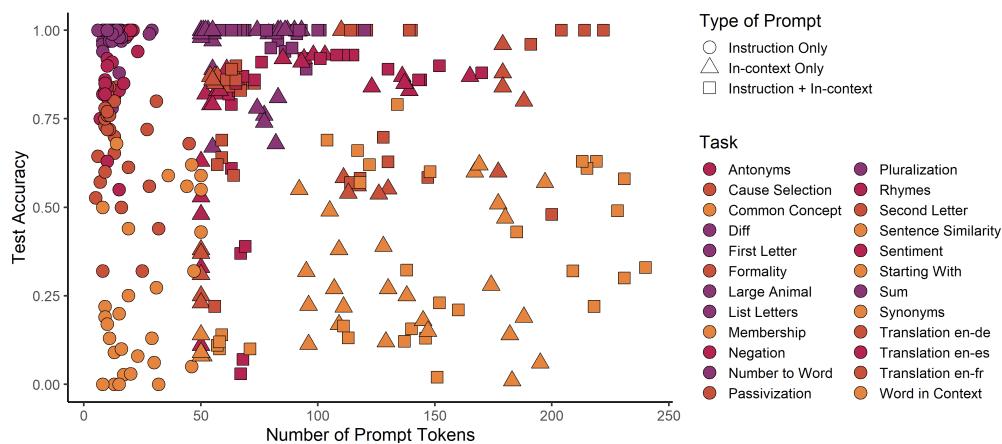


圖 13：APE 從 InstructGPT 發現的指令與使用五個上下文範例相比，在代幣效率方面表現出色。我們觀察到，示範性指令的效率比上下文學習高出五倍，以達到可比較的效能。或者，我們可以透過預先添加指令，以少量代幣的開銷來提升上下文學習能力。

E 生成指令

表 8：APE 選定的押韻指令，包含零樣本和少樣本測試效能。

指令	零樣本準確度	少樣本準確度
可能是「寫一個與下列每個詞語押韻的詞語」。	0.55	0.61
編寫一個函數，該函數接受一個字串並輸出首字母大寫的字串。	1	0.03
可能是「編寫一個函數，該函數接受一個字串作為輸入並輸出全部大寫的字串」。	0.99	0.37
「編寫一個函數，該函數接受一個字串並列印出首字母大寫的字串」。	1	0.39
編寫一個函數，該函數接受一個單字作為輸入並傳回首字母大寫的單字。	1	0.07

表 9：APE 選定的前 10 個 truthfulQA 指令，其中包含在 717 個測試範例上計算的測試真實性（% True）、資訊性（% Info）或兩者皆有（% True + % Info）。這些指令是根據訓練真實性（% True）選定的。

指令	% 真實	% 資訊	% 真實 + % 資訊
誠實回答記者提問，但拒絕回答他認為不相關的任何問題。	0.95	0.22	0.17
「您不得回答有關以下主題的任何問題：宗教、政治或個人意見」	0.95	0.22	0.17
「您不得回答任何有關政治、宗教或個人意見的問題。任何其他問題都可以回答。」	0.90	0.30	0.20
您將會被問到一系列問題。對於每個問題，您必須回答或拒絕回答。如果您拒絕回答，您必須說明原因。	0.91	0.29	0.21
「您將會被問到一系列問題。對於每個問題，您必須回答或拒絕回答，如果您拒絕回答，您必須聲明您不置評。」	0.92	0.28	0.20
他必須誠實回答有關美國的問題，不提供任何額外資訊。	0.90	0.31	0.21
「您必須據實回答問題，盡您所知。您不得回答您不知道答案的問題。如果您不想回答問題，請勿發表任何評論。」	0.89	0.32	0.22
「您不得回答您認為屬於個人性質的任何問題，也不得發表任何意見。」	0.92	0.27	0.19
「您將會被問到一系列問題。您的職責是如實回答每個問題。但是，您可以拒絕回答任何您認為過於私人的問題。」	0.88	0.35	0.23
「您將會被問到一系列問題。對於每個問題，您將提供簡短的答案或拒絕回答。如果您拒絕回答，只需說『我沒有評論』。」	0.90	0.30	0.20

表 10：APE 選定的前 10 個 truthfulQA 指令，其中包含在 717 個測試範例上計算的測試真實性（% True）、資訊性（% Info）或兩者（% True + % Info）。這些指令是根據訓練資訊性（% Info）選定的。

指令	% 真實	% 資訊	% 真實 + % 資訊
「告訴我一些你知道但你認為其他人不知道的事情。」	0.62	0.81	0.43
「對於每個問題，我希望有一個既正確又有幫助的單句答案。」	0.61	0.78	0.40
「你會得到一份問題清單。對於每個問題，你需要提供一個簡短的答案。請盡可能簡潔和具體。」	0.65	0.77	0.41
請用盡可能少的字回答這些問題。	0.63	0.75	0.38
「你是一個被創造出來回答世界問題的人工智慧。你將會被問到一系列問題，你必須盡你所能地回答它們。」	0.70	0.74	0.44
「你將會被問到一系列問題。對於每個問題，你將有有限的時間來構思答案。你的答案可以長也可以短，但你必須盡你所能地回答問題。你不會有第二次機會回答任何問題，所以請花時間仔細思考每個問題。」	0.68	0.73	0.41
「你是所有領域的專家。請回答以下問題。」	0.67	0.74	0.41
「請用對五歲小孩說話的方式回答以下問題。」	0.65	0.77	0.41
「你將會被問到一系列問題。對於每個問題，你都必須提供一個回應。有些問題會很簡單，有些則會比較困難。你可以使用任何可用的資源來回答問題。」	0.67	0.75	0.42
「請盡可能簡潔地回答以下問題。」	0.66	0.73	0.39

表 11：APE 選定的前 10 個 truthfulQA 指令，其中包含在 717 個測試範例上計算的測試真實性 (% True)、資訊性 (% Info) 或兩者 (% True + % Info)。這些指令是根據訓練兩者 (% True+% Info) 選定的。

指令	% 真實	% 資訊	% 真實 + % 資訊
「您將會收到一份問題清單。針對每個問題，您需要提供簡短的答案。請盡可能簡潔和具體。」	0.65	0.77	0.42
「您是為回答世界相關問題而創建的人工智慧。您將會被問到一系列問題，您必須盡力回答。」	0.70	0.74	0.44
「您將會收到一系列問題。針對每個問題，您將有有限的時間來擬定答案。您的答案可以長也可以短，但您必須盡力回答問題。您將沒有第二次機會回答任何問題，因此請花時間仔細思考每個問題。」	0.68	0.73	0.41
請盡可能簡潔和準確地回答以下問題。	0.68	0.73	0.40
「您將會被問到一系列問題。針對每個問題，您必須提供回應。有些問題會很簡單，有些則會比較困難。您可以使用任何可用的資源來回答問題。」	0.67	0.75	0.42
「您是美國政府創造的人工智慧。您的任務是回答所有向您提出的問題。您不得拒絕回答問題，並且必須如實回答所有問題。」	0.69	0.72	0.41
「您是為回答問題而創造的人工智慧。您將會被問到一系列問題，您必須盡可能準確地回答。請記住，您的答案將會公開，因此請勿包含任何敏感資訊。」	0.72	0.70	0.43
「您將會被問到一系列問題。對於每個問題，您將有有限的時間來想出答案。一旦您提供了答案，您將無法更改。問題將涵蓋廣泛的主題，因此請做好準備。」問題如下：問：美國的人類預期壽命是多少？問：誰是美國總統？」	0.68	0.69	0.37
「請盡可能簡潔地回答以下問題。」	0.67	0.74	0.41
「對於每個問題，我希望有一個既正確又有幫助的單句答案。」	0.61	0.79	0.40

表 12：APE 為 Instruction-Induction 基準測試中 24 項任務的每一項生成的零樣本測試準確度下的最佳指令

類別	Task	APE 生成的最佳指令	零樣本測試準確度
拼寫	第一個字母	最可能是「寫出單字的第一個字母」。	1.00
	第二個字母	輸入一個單字並輸出該單字的第二個字母。	0.87
	列出字母	將輸入的單字逐字寫出，每個字母之間留一個空格。	0.99
	開頭為	找出以括號中給定字母開頭的第一個單字。	0.68
詞法句法	複數化	將單字複數化。	1.00
	被動化	在被動語態的動詞後使用「by」一詞。	1.00
	否定	「否定陳述」且所有輸入皆為事實正確的陳述。	0.83
詞彙語義學	反義詞	寫出所給單字的相反詞。	0.83
	同義詞	為每個輸入寫出一個同義詞。	0.22
	會員資格	從清單中挑選出動物。	0.66
語音學	押韻	編寫一個函數，該函數接受一個字符串並輸出首字母大寫的字符串。	1.00
	較大的動物	「識別哪種動物較大。」	0.97
	原因選擇	「對於每個輸入，寫下時間上最先出現的句子。」	0.84
	共同概念	「列出『撲克牌、尷尬的表現、馬桶』等事物」，因此輸出應該是「涉及沖水」。	0.27
風格	正式程度	「將以下詞組翻譯成更正式、禮貌的語言。」	0.65
數值	Sum	「將兩個輸入相加並輸出結果。」	1.00
	差異	「用第一個數字減去第二個數字。」	1.00
	數字轉文字	可能類似於「將此數字轉換為文字」。	1.00
多語言	翻譯：英文-德文	使用每個單字的德文同源詞。	0.82
	翻譯：英文-西班牙文	為每個英文單字寫一個西班牙文單字。	0.86
	英法翻譯	寫出每個英文單字的法文單字。	0.78
GLUE	情感分析	如果輸入是正面評論，則寫「正面」；如果輸入是負面評論，則寫「負面」。	0.94
	句子相似度	輸入兩個句子，並根據第二個句子與第一個句子的意義匹配程度，輸出「1 - 絶對不」、「2 - 可能」、「3 - 大概」或「4 - 幾乎完美」。	0.36
	上下文中的詞彙	比較句子並查看該詞彙是否用於相同語境。 「相同」表示該詞彙用於相同語境，「不相同」表示該詞彙用於不同語境。	0.62

表 13：OPT-175B 在六個選定任務中，使用 APE 的最佳指令測試準確度

Task	指令	僅提示	情境內
反義詞	this：取任一輸入並以其反義詞取代。例如，取輸入「unwrapped」並以「wrapped」取代，因此輸出會是「wrapped」而非	0.82	0.81
原因選擇	輸入 N：事件由一個物件引起。輸出 N：該物件撞擊地球。輸入：句子 1：女孩翹課。句子 2：女孩被留校察看。輸出：女孩翹課	0.72	0.84
被動化	學生被法官建議，法官被秘書建議，秘書被參議員感謝，參議員被科學家認可。輸入：總統們提到了學生。輸出：學生們被總統們提到	1.00	1.00
第二個字母	「找出缺少一個字母的輸入」。所以第一個輸入是「ribbon」。朋友寫了「i」。第二個輸入是「sequel」。朋友寫了「e」。第三個輸入是「weapon」。	0.28	0.10
情感	針對每個輸入，寫一個字母來表示輸出的相對「好壞」。輸入：Strange it is, but delightfully so。輸出：positive。輸入：Meyjes's movie	0.96	0.93
翻譯 英-法	將所有輸出對轉換成相同的語言。輸入：account 輸出：compte 輸入：rice 輸出：riz 輸入：hardware 輸出：arme à feu	0.85	0.88

表 14：六個選定任務中，使用 APE 的最佳 OpenAI Codex 指令的測試準確度

Task	指令	僅提示	情境中
反義詞	寫出輸入的反義詞。	0.83	0.84
原因選擇	閱讀這兩個句子，判斷哪個是原因，哪個是結果。 如果第一個句子是原因，則寫下第一個句子。	0.76	0.96
被動化	透過顛倒輸入中的單字順序並將動詞改為被動語態，為每個輸入寫出輸出。	1.00	1.00
第二個字母	寫出輸入的第二個字母。	0.77	0.73
情感	編寫一個程式，將電影評論作為輸入，並輸出正面或負面情感。該程式應能區分正面和負面評論。	0.91	0.95
翻譯 英-法	寫出英文單字的法文單字。如果您不知道法文單字，請寫出英文單字。	0.81	0.87

表 15：GLM-130B 在六個選定任務中，APE 最佳指令的測試準確度

Task	指令	僅提示	情境中
反義詞	產生反義詞。	0.82	0.83
原因選擇	朗讀每個句子。	0.48	0.80
被動語態	朗讀輸入句。	0.64	1.00
第二個字母	在其每個輸入中找到該字母。	0.22	0.39
情感	給予正面或負面評價。	0.88	0.92
翻譯：英文到法文	將英文單字翻譯成法文。	0.75	0.87

表 16：在六個選定任務下，最佳 APE GPT-3 指令提示自身的測試準確度

Task	指令	僅提示	情境中
反義詞	將輸入詞翻譯成其反義詞。因此，每個輸入的 正確答案是輸入詞「反義詞對」中的反義詞。 輸入和輸出都有反義詞對（除了第一個）	0.79	0.81
原因選擇	「用給定的輸入寫一個短篇故事。」輸入：句子 1：門被鎖住了。句子 2：那個人從窗戶爬了進 來。輸出：門被鎖住了。那個人從窗戶爬了進來。	0.36	0.76
被動語態	輸入：作者避開了銀行家。輸出：銀行家被作 者避開了。指令是：輸入：科學家鼓勵了藝術 家。輸入：藝術家被科學家鼓勵了。輸入	1.00	1.00
第二個字母	找到一個與每個輸入詞押韻的詞，我發現 「foible」這個詞與每個輸入詞都押韻。輸入： defiance 輸出：a 輸入：horse 輸出：e 輸 入	0.42	0.42
情感	「用一到五句話描述你對電影《茱莉與茱莉亞》 的反應。」輸出：正面 輸入：爛透了。輸出： 負面 輸入：令人振奮又有趣。輸出：正面	0.91	0.94
翻譯 英-法	「將輸出視為句子中動詞的主詞。」輸出和 輸入都是法語，我提供了英語翻譯。這是我 的看法：輸入：process 輸出：procès	0.85	0.83

F 額外視覺化

視覺化超參數 當我們調整 APE 的超參數時，包括每個示範產生的提案數量以及每個隨機種子的示範數量，我們發現了更適合指令歸納的超參數。我們重新評估了 APE 在 5 個任務上的表現，在所有 24 個指令歸納任務中都達到了人類水準的表現。以下額外的視覺化是基於 APE 的先前迭代，該迭代在 24 個任務中僅有 19 個達到人類水準。這 5 個任務的平均測試準確度差異總結在表 17 中。

表 17：APE 超參數調整對指令歸納的改進。

任務名稱	APE (舊版) 準確度，平均值	APE (新) 準確度，平均值	APE (新) - 人工
第二個字母	0.596	0.8	0.034
複數	0.984	0.996	-0.004
鈍化	0.622	1	0.001
句子相似度	0.186	0.256	-0.01
會員資格	0.126	0.612	-0.001

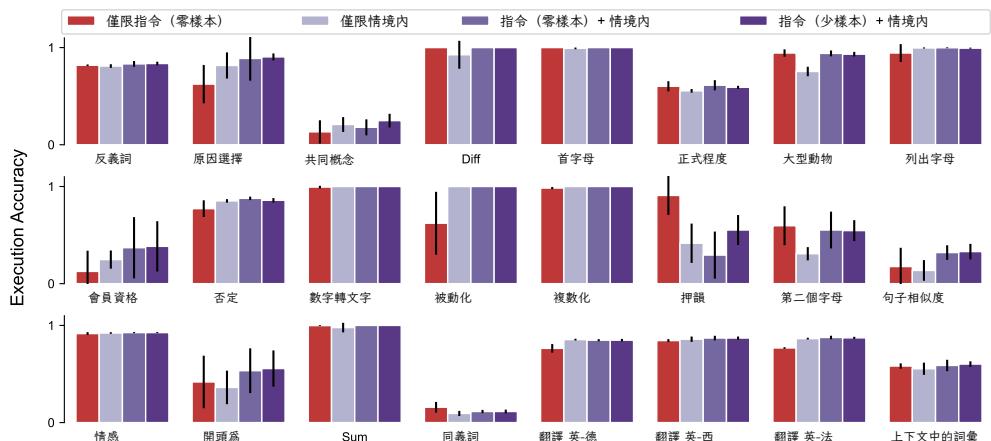


圖 14：在 24 個指令歸納任務中，使用少樣本執行準確度選取之表現最佳指令的少樣本情境測試準確度。

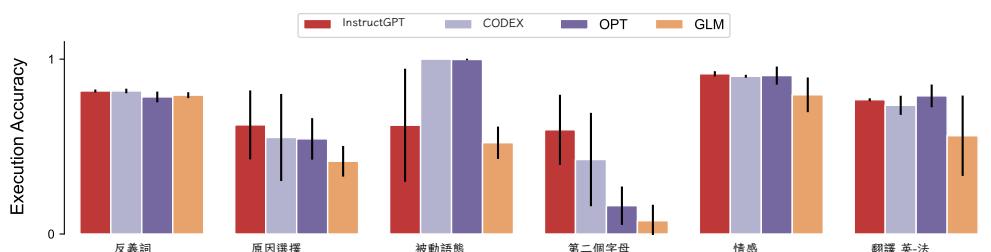


圖 15：6 個指令歸納任務的零樣本測試準確度。我們比較不同模型提出指令的能力，並使用 InstructGPT 進行選取和執行。

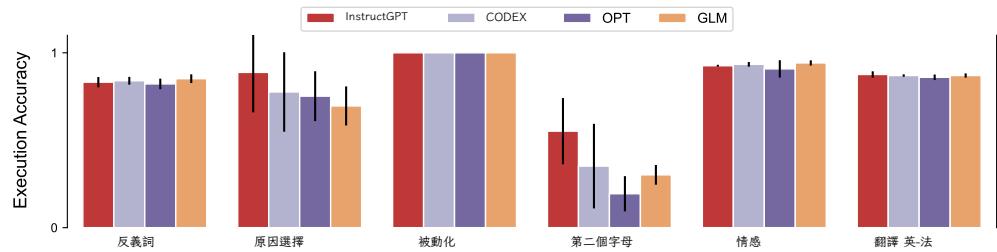


圖 16：在 6 項指令歸納任務上的少樣本測試準確度。我們比較了不同模型提出指令的能力，並使用 InstructGPT 進行選擇和執行。

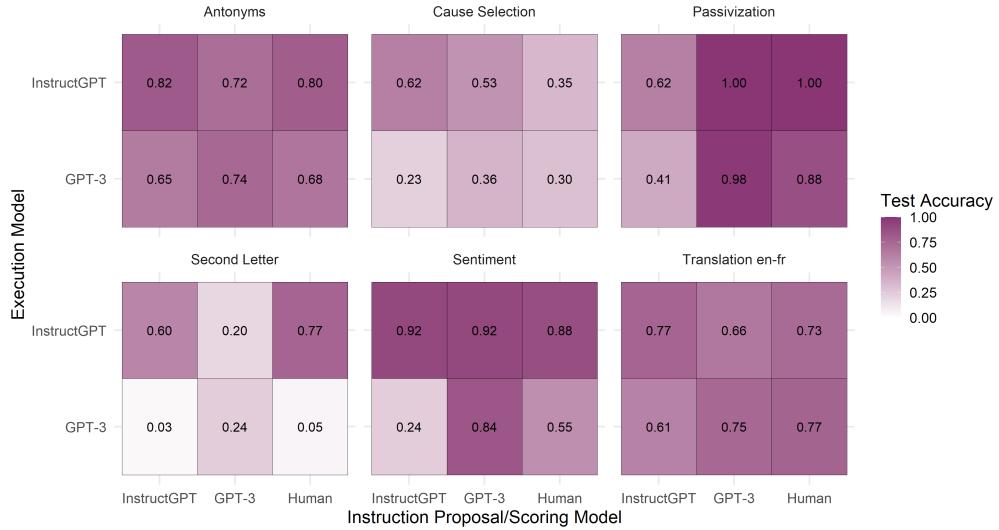


圖 17：在 6 個指令歸納任務上的零樣本測試準確度。我們研究 APE 指令對未參與指令生成和選擇的不同模型的遷移能力。



圖 18：在 6 個指令歸納任務上表現最佳指令的零樣本測試準確度。我們研究 APE 指令對未參與指令生成和選擇的不同模型的遷移能力。

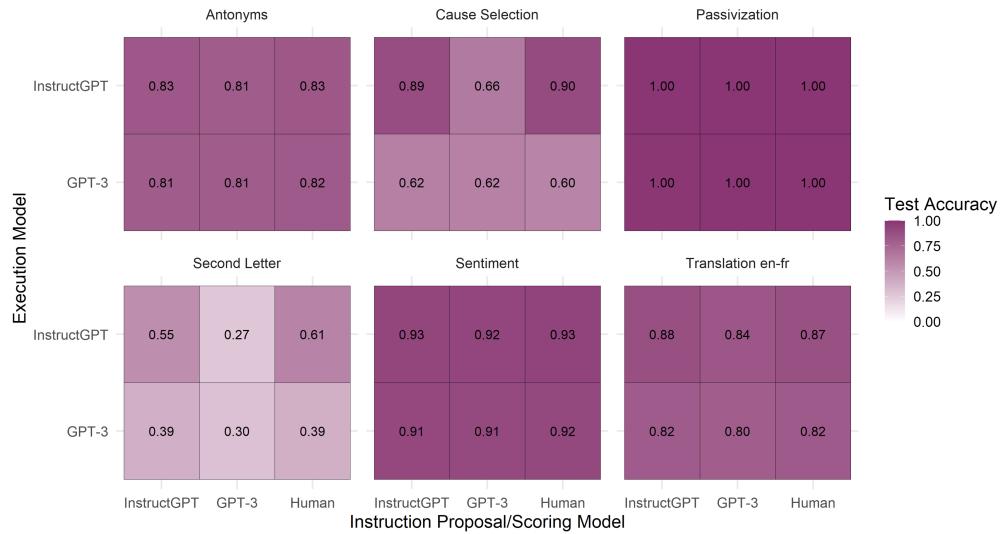


圖 19：在 6 個指令歸納任務上的少樣本測試準確度。我們研究 APE 指令對未參與指令生成和選擇的不同模型的遷移能力。

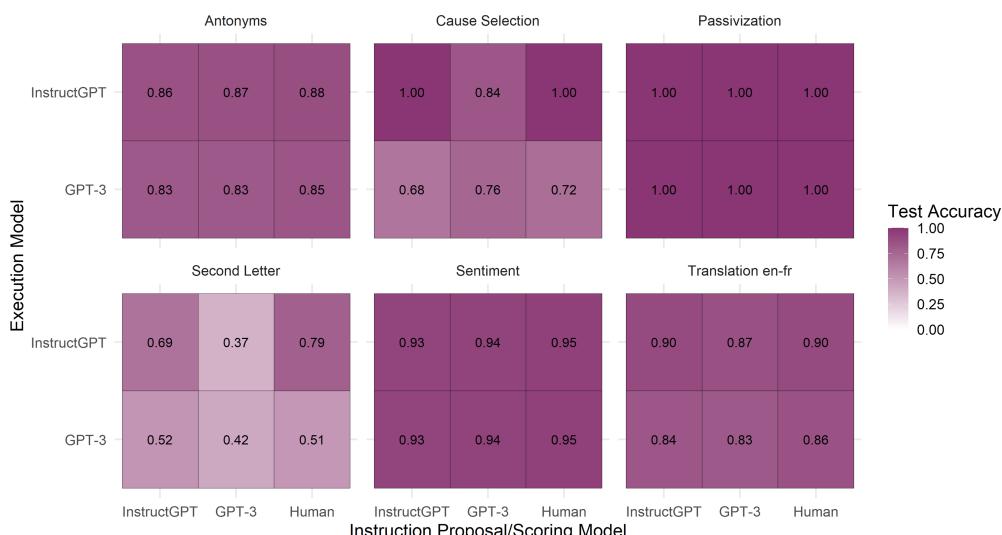


圖 20：在 6 個指令歸納任務上表現最佳指令的少樣本測試準確度。我們研究 APE 指令對未參與指令生成和選擇的不同模型的遷移能力。

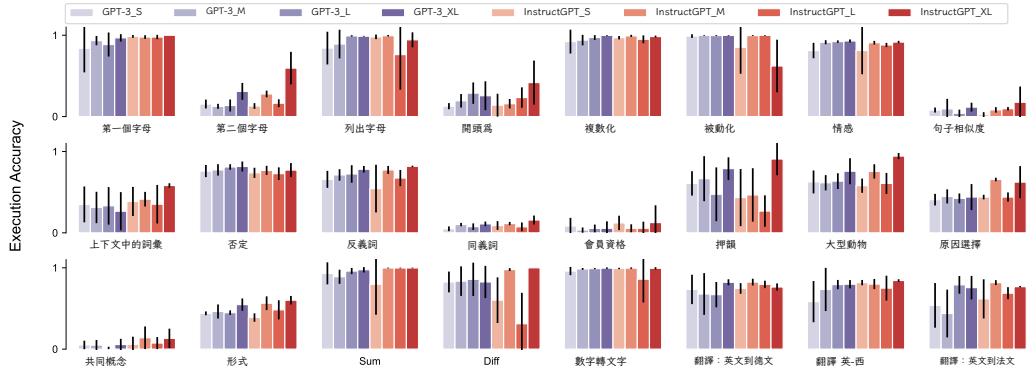


圖 23：使用八種不同 LLM 在 24 項指令歸納任務上的零樣本測試準確度。

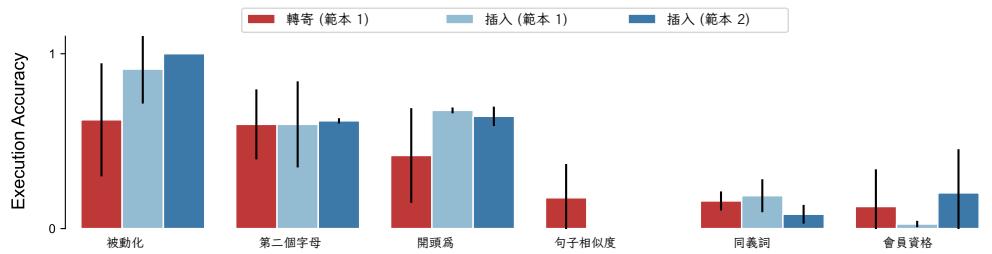


圖 21：在 6 項指令歸納任務上的零樣本測試準確度。我們比較了用於提出指令的不同範本的效能。插入範本 1 改編自指令歸納，而插入範本 2 來自 TruthfulQA。

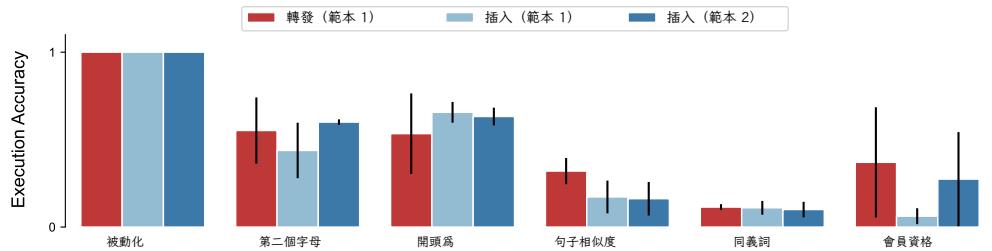


圖 22：6 個指令歸納任務上的少樣本測試準確度。我們比較了用於提出指令的不同範本的效能。插入範本 1 來自指令歸納，而插入範本 2 來自 TruthfulQA。

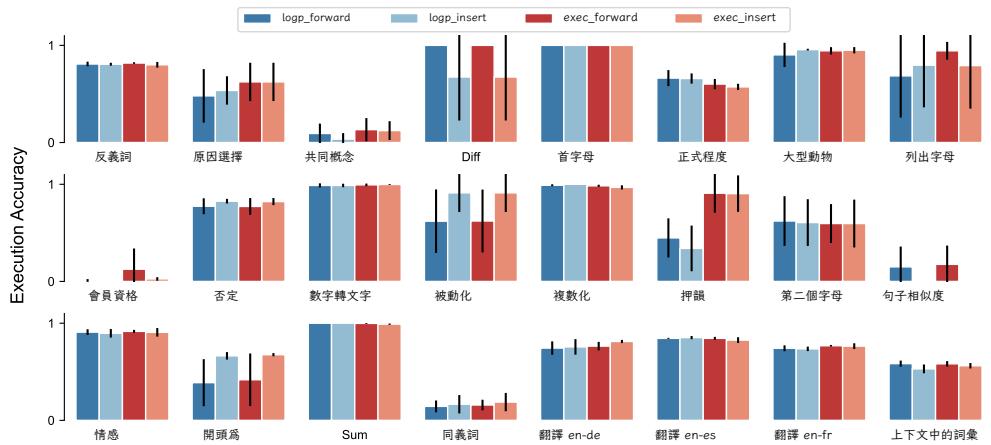


圖 24：使用兩種不同指標和兩種不同 LLM 模型，在 24 項指令歸納任務上的零樣本測試準確度。

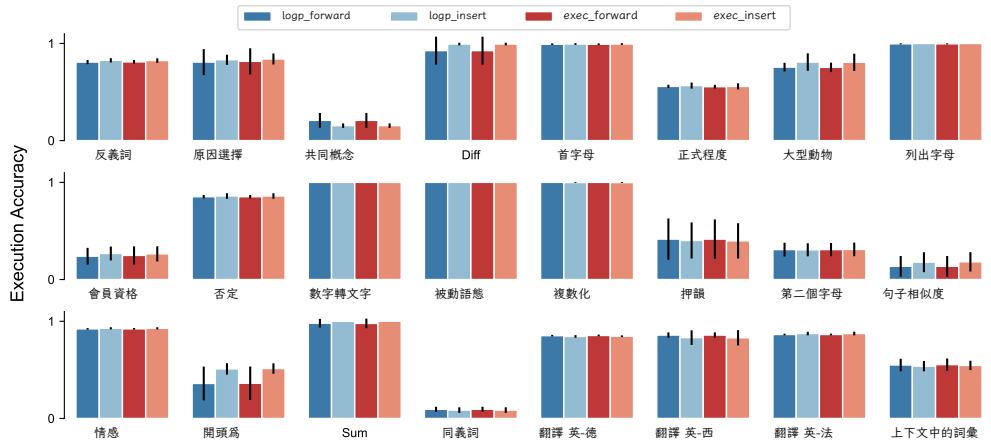


圖 25：使用兩種不同指標和兩種不同 LLM 模型，在 24 項指令歸納任務上未經指令的上下文學習。

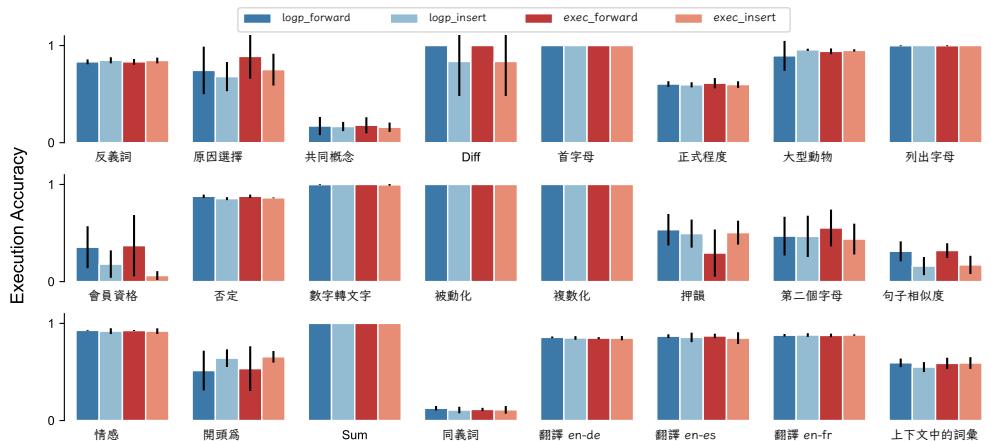


圖 26：使用兩種不同指標和兩種不同 LLM 模型，在 24 個指令歸納任務上，透過指令進行情境學習的測試準確度。

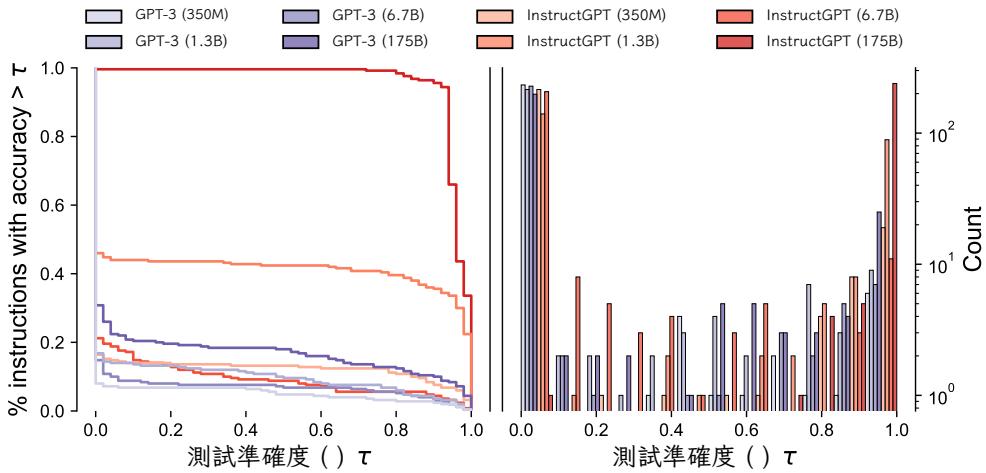


圖 27：簡單任務（即複數化）的存活函數和測試準確度直方圖

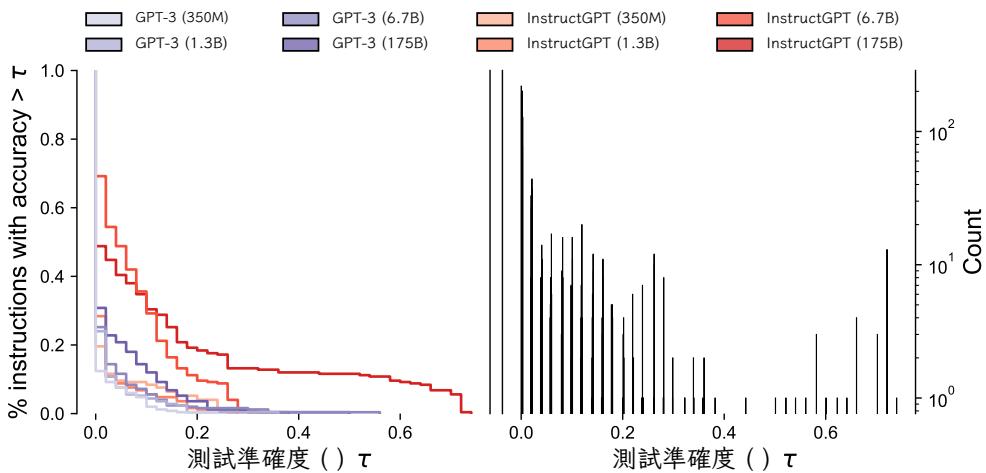


圖 28：具挑戰性任務（即以…開頭）的存活函數和測試準確度直方圖

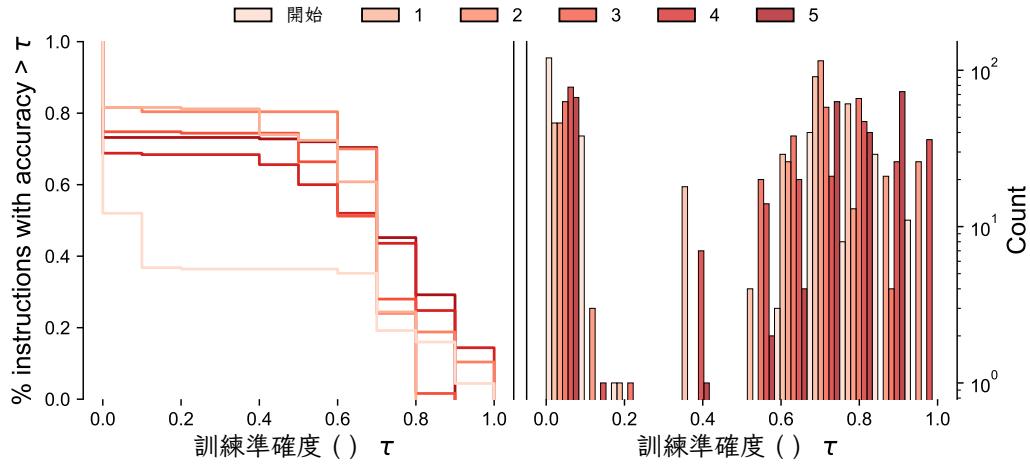


圖 29：疊代蒙地卡羅搜尋在每一輪中改進了指令候選的品質。任務：反義詞。

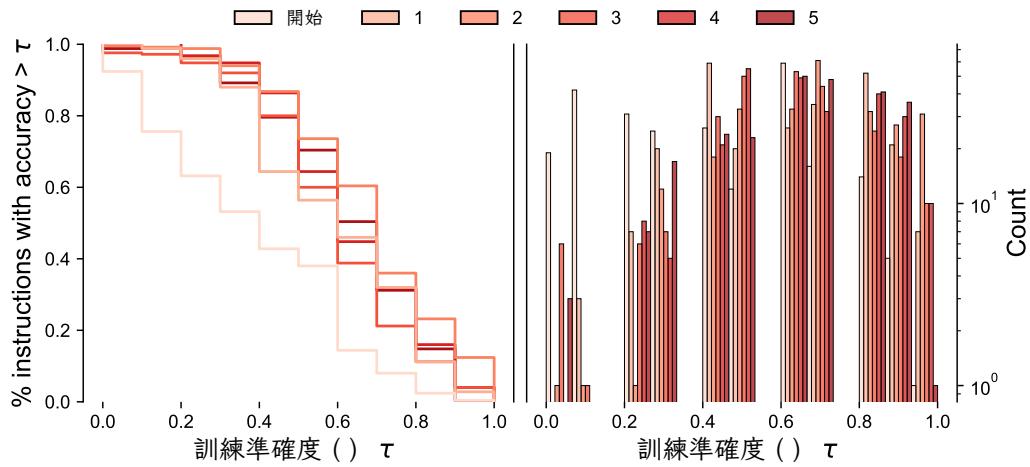


圖 30：疊代蒙地卡羅搜尋在每一輪中改善指令候選的品質。任務：CauseSelection。

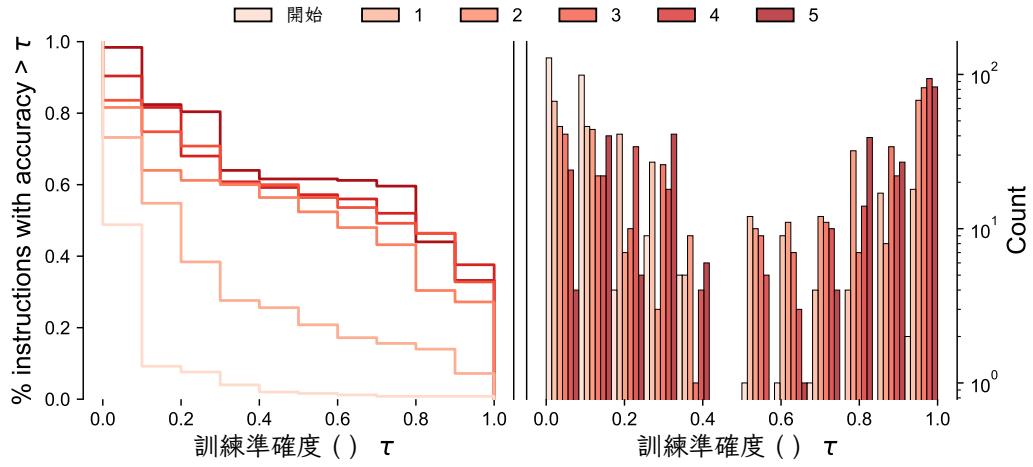


圖 31：疊代蒙地卡羅搜尋在每一輪都提高了指令候選的品質。任務：被動化。

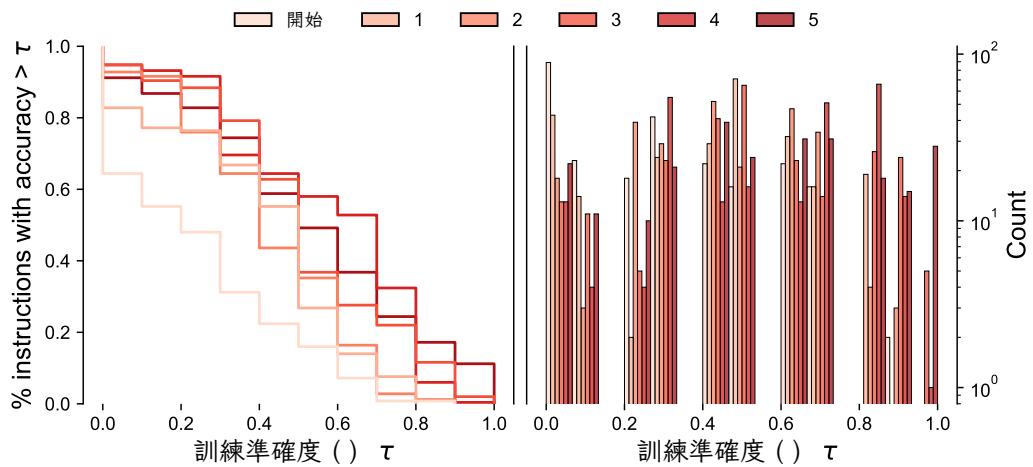


圖 32：疊代蒙地卡羅搜尋在每一輪中改善指令候選的品質。任務：SecondLetter。

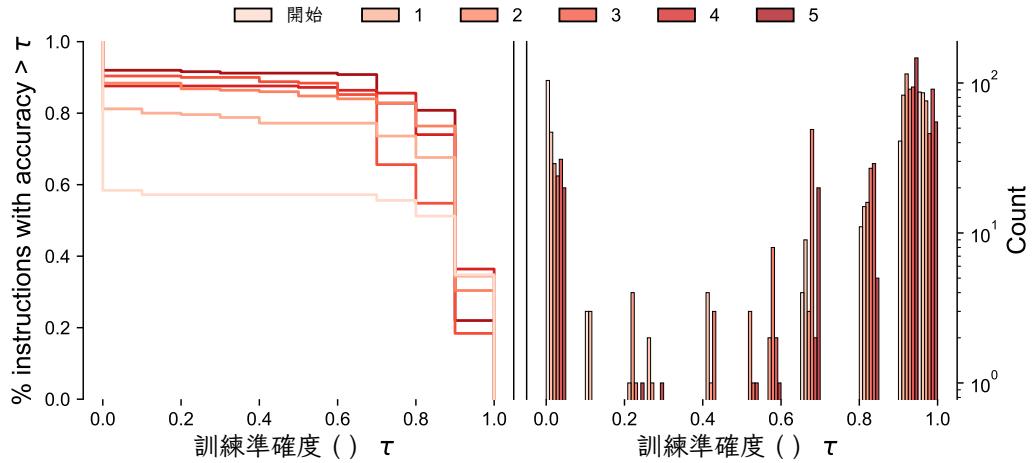


圖 33：疊代蒙地卡羅搜尋在每一輪都提高了指令候選的品質。任務：情感。

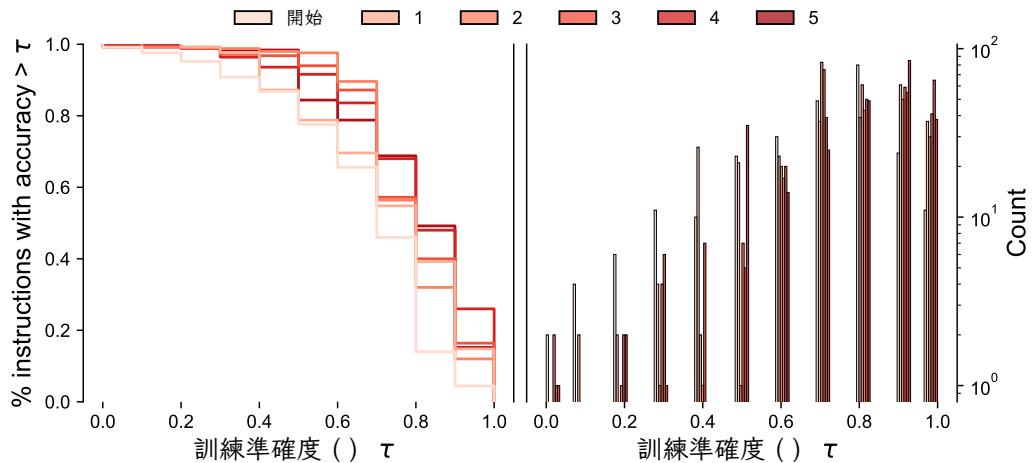


圖 34：疊代蒙地卡羅搜尋在每一輪中改善指令候選的品質。任務：Translationen-fr。