

Engenharia de Prompt

Bem-vindo ao curso de **Engenharia de Prompt**, projetado para capacitar profissionais a dominar a arte de criar prompts eficazes para modelos de linguagem avançados. Este curso abrangente fornecerá as ferramentas e técnicas necessárias para maximizar o potencial das ferramentas de IA em diversos campos.

Imagine a **Engenharia de Prompt** como uma conversa com um assistente muito inteligente, mas que precisa de instruções claras - como explicar uma tarefa para um novo colega de trabalho. Por exemplo, um profissional de marketing pode usar prompts bem construídos para gerar ideias criativas para campanhas, enquanto um professor pode criar exercícios personalizados para seus alunos.

Na prática, você encontrará aplicações em diversos cenários: desde um desenvolvedor otimizando a geração de código através de prompts específicos, até um redator criando conteúdo em diferentes tons de voz para várias pessoas. Um caso real é o de uma empresa que **reduziu em 50% o tempo** de atendimento ao cliente ao usar prompts bem estruturados para gerar respostas personalizadas e precisas.





Introdução à Engenharia de Prompt

A engenharia de prompt é um conhecimento indispensável no campo da inteligência artificial, para termos sucesso quando interagimos com modelos de linguagem avançados. Esta área combina elementos de programação, psicologia e linguística para otimizar a comunicação homem-máquina.

1

2

3

1 Definição Aprofundada

Arte e ciência de construir instruções precisas para IAs

2 Objetivo Estratégico

Maximizar resultados através de prompts otimizados

3 Base Fundamental

Comunicação efetiva com sistemas de IA

A engenharia de prompt é uma ciência empírica que requer experimentação constante e refinamento metodológico. Ela envolve não apenas a criação de instruções, mas também a compreensão profunda de como os modelos de linguagem interpretam e processam informações.

Importância da Engenharia de Prompt

A Engenharia de Prompt emerge como uma das competências mais cruciais no cenário atual da inteligência artificial, representando a interface entre humanos e sistemas avançados de IA. Esta disciplina combina elementos de comunicação, lógica e compreensão profunda dos modelos de linguagem para maximizar seu potencial.

1 Desbloqueando Potencial

A engenharia de prompt é uma habilidade fundamental para dominar o uso de modelos de linguagem. Ela abre portas para diversas aplicações, desde a automação de processos complexos até o desenvolvimento de soluções criativas. Profissionais que dominam esta habilidade conseguem extrair resultados significativamente superiores dos sistemas de IA, otimizando seu desempenho em diferentes contextos e aplicações.

2 Demanda Alta

A demanda por profissionais com essa habilidade é alta, com salários competitivos no mercado. Empresas de diversos setores buscam especialistas capazes de desenvolver prompts eficientes para suas aplicações de IA. Esta demanda reflete-se em oportunidades crescentes em áreas como desenvolvimento de software, pesquisa, consultoria e inovação tecnológica, com remunerações que reconhecem a especialização nesta competência estratégica.

3 Crescimento da Área

A área de Engenharia de Prompt está em constante expansão, com novas ferramentas e aplicações surgindo o tempo todo. Este crescimento é impulsionado pela evolução contínua dos modelos de linguagem, pelo surgimento de novas tecnologias e pela crescente adoção de soluções baseadas em IA. O campo oferece oportunidades constantes de aprendizado e inovação, com metodologias e técnicas sendo constantemente refinadas e aperfeiçoadas.

Dominar a Engenharia de Prompt não é apenas uma vantagem competitiva, mas uma necessidade crescente no mercado tecnológico atual. Esta habilidade representa um diferencial significativo para profissionais que desejam se destacar na era da inteligência artificial, oferecendo possibilidades praticamente ilimitadas de aplicação e desenvolvimento.

Regra de Ouro da Engenharia de Prompt

1 Comunicação Clara

Um prompt bem escrito deve ser claro e conciso, como se estivesse explicando uma tarefa para uma pessoa. Evite linguagem rebuscada ou termos técnicos desnecessários. A clareza na comunicação permite que o modelo de IA entenda exatamente o que você precisa, reduzindo a chance de respostas inadequadas ou imprecisas. Pense em como você explicaria a tarefa para um colega de trabalho que está começando na área.

2 Linguagem Natural

Utilize linguagem natural, como faria ao conversar com um colega, para garantir que o modelo de linguagem compreenda o contexto. Evite estruturas artificiais ou muito formais. A naturalidade na comunicação facilita o processamento do modelo e geralmente resulta em respostas mais apropriadas e contextualizadas. Além disso, uma linguagem natural ajuda a estabelecer um tom conversacional que pode melhorar a qualidade das interações.

3 Especificidade

Forneça informações precisas e detalhadas para evitar ambiguidades e garantir resultados relevantes. Quanto mais específico for seu prompt, melhor será a resposta do modelo. Inclua detalhes importantes como formato desejado, extensão da resposta, estilo de escrita, e quaisquer restrições ou preferências específicas. A especificidade ajuda a direcionar o modelo para produzir exatamente o tipo de conteúdo que você precisa.

4 Compreensão do Modelo

Se a tarefa é clara para um humano, é mais provável que o modelo de linguagem também a compreenda. Isso se deve ao fato de que os modelos de IA são treinados com base em comunicações humanas. Considere sempre se suas instruções fariam sentido para outra pessoa. Teste diferentes abordagens e aprenda com os resultados, refinando sua técnica de comunicação com o modelo. Lembre-se que o modelo aprende melhor quando as instruções são estruturadas de forma lógica e coerente.



Estrutura básica de prompt

Estrutura Básica

Persona, Tarefa, Contexto e Formato

- **Persona:** Define quem ou o que a IA deverá emular (se comportar como) durante a interação. Isso pode incluir especificações sobre o estilo de linguagem, nível de formalidade, e até traços de personalidade. A persona ajuda a criar uma experiência de interação mais coesa e personalizada.
- **Tarefa:** Descreve especificamente o que você deseja que a IA realize. A clareza na descrição da tarefa é essencial para garantir que os objetivos do prompt sejam alcançados com sucesso. Isso inclui detalhar ações, resultados esperados e quaisquer limitações ou condições especiais.
- **Contexto:** Fornece informações necessárias para que a IA entenda a situação em que o prompt está sendo aplicado. O contexto pode incluir detalhes sobre o ambiente, a situação específica, as pessoas envolvidas, ou quaisquer nuances que influenciam a maneira como a tarefa deve ser executada.
- **Formato:** Refere-se à estrutura e à organização do prompt. Isso pode incluir a escolha entre um prompt direto ou um mais aberto, a decisão de usar perguntas ou afirmações e a forma como a informação é apresentada para maximizar a clareza e a eficácia da resposta da IA.

Estrutura básica de prompt

Prompt Escrito de Forma Estruturada Básica

Persona, Tarefa, Contexto e Formato

"Você é um especialista em vendas B2B com ampla experiência no setor educacional. Sua tarefa é criar um pitch persuasivo para atrair diretores de recursos humanos de grandes empresas para inscreverem seus funcionários em um curso de pós-graduação em negócios. O pitch deve destacar os diferenciais da instituição, como a qualidade do corpo docente, a metodologia de ensino prática e o impacto na performance dos colaboradores. A abordagem deve ser formal, mas cativante. O pitch deve ser escrito no formato textual, ter no máximo, 150 palavras e incluir um call-to-action claro no final."

Esse prompt segue uma estrutura bem definida de **Persona, Tarefa, Contexto e Formato**.

Elemento	Por que é bom?
Persona	Define claramente quem o modelo deve emular: "um especialista em vendas B2B com ampla experiência no setor educacional". Isso dá ao modelo uma identidade específica e alinhada ao contexto, ajudando-o a adotar um tom apropriado e fazer sugestões relevantes.
Tarefa	A descrição da tarefa é direta e específica: "criar um pitch persuasivo para atrair diretores de recursos humanos". Isso orienta o modelo a focar em um objetivo claro, eliminando ambiguidades e aumentando a probabilidade de produzir uma resposta adequada à necessidade.
Contexto	O contexto é detalhado, indicando o público-alvo (diretores de RH de grandes empresas) e os pontos a serem destacados (qualidade do corpo docente, metodologia prática e impacto na performance). Isso ajuda o modelo a compreender o ambiente e as expectativas, personalizando a resposta para maximizar relevância e engajamento.
Formato	Especifica de forma precisa a estrutura da resposta: "formato textual", "máximo de 150 palavras", e "incluir um call-to-action claro". Esses detalhes garantem que o output esteja alinhado ao propósito e limitações práticas, como tempo e espaço, proporcionando um conteúdo direto e objetivo.

Técnicas Básicas

Formatação Markdown

A formatação adequada torna seus prompts mais claros e eficazes, tanto para o modelo quanto para humanos.

<https://www.markdownguide.org/cheat-sheet/>

Elemento (Inglês)	Sintaxe em Markdown
Heading (Cabeçalho)	# H1 ## H2 ### H3
Bold (Negrito)	**bold text**
Italic (Itálico)	<i>*italicized text*</i>
Blockquote (Citação)	> blockquote
Ordered List (Lista Ordenada)	1. First item 2. Second item 3. Third item
Unordered List (Lista Não Ordenada)	- First item - Second item - Third item
Code (Código)	`code`
Horizontal Rule (Linha Horizontal)	---
Link (Link)	[title](https://www.example.com)
Image (Imagem)	![alt text](image.jpg)



Use Markdown para estruturar seus prompts de forma consistente e profissional.

Técnicas Básicas

Exemplo de Prompt para o ChatGPT

> ****Instruções:**** Crie uma estrutura detalhada para gerenciamento de tarefas utilizando Markdown. Inclua as seguintes informações:

1. ****Categorias principais:****

- Prioridade alta
- Prioridade média
- Prioridade baixa

2. ****Responsáveis por cada tarefa:**** Liste os nomes dos responsáveis. Exemplo: João, Ana, Pedro.

3. ****Prazos:**** Indique os prazos em formato de data.

4. ****Status da tarefa:**** Classifique como:

- Em andamento
- Concluído
- Pendente

Formato de saída esperado

Categorias de Prioridade

1. **Prioridade Alta**

- Descrição: *Descreva as tarefas mais urgentes.*

2. **Prioridade Média**

- Descrição: *Descreva tarefas intermediárias.*

3. **Prioridade Baixa**

- Descrição: *Descreva tarefas menos urgentes.*

Exemplo de Saída

****Responsável:**** João

****Tarefa:**** Preparar apresentação do relatório mensal

****Prazo:**** 15/12/2024

****Status:**** *Em andamento*

Técnicas Básicas

Usando Delimitadores em Prompts

Delimitadores são elementos fundamentais para estruturar seus prompts de forma clara e eficiente, permitindo uma comunicação mais precisa com modelos de IA. Eles ajudam a separar diferentes partes do prompt e tornam as instruções mais organizadas e fáceis de entender.

Delimitadores Básicos

Usando traços, hashtags ou outros símbolos para separar seções:

--Instruções
Criar uma lista

--Contexto
Ambiente corporativo

--Exemplo
Item 1, Item 2

--Saída
Formato desejado

Estes delimitadores são ideais para prompts simples e rápidos, sendo fáceis de digitar e ler.

Delimitadores XML

Usando tags XML para maior precisão e estruturação:

```
<instrucoes>  
Criar uma lista detalhada  
</instrucoes>
```

```
<contexto>  
Ambiente corporativo  
</contexto>
```

```
<exemplo>  
Item 1, Item 2  
</exemplo>
```

```
<saida>  
Formato desejado  
</saida>
```

Tags XML são excelentes para prompts complexos que exigem estruturação precisa.

Melhores Práticas

Para um uso efetivo de delimitadores:

- Mantenha consistência no estilo escolhido
- Use delimitadores visualmente distintos do conteúdo
- Escolha o tipo adequado para cada caso de uso
- Combine diferentes tipos quando necessário

 Em casos complexos, considere combinar diferentes tipos de delimitadores para maior clareza e precisão.

Variáveis em Prompts

As variáveis são elementos dinâmicos que tornam seus prompts mais flexíveis e reutilizáveis. Elas permitem que você crie templates de prompts que se adaptam a diferentes situações.

Usando a sintaxe {{variável}}, você pode inserir valores variáveis em seus prompts. Dessa forma, o mesmo prompt pode ser usado para diferentes cenários, sem a necessidade de reescrevê-lo por completo.

```
Olá {{nome}},  
Crie um resumo sobre {{tópico}}  
com {{número}} pontos principais.
```

Flexibilidade

Permite reutilizar o mesmo prompt com diferentes valores

Manutenção

Facilita a atualização e gerenciamento de prompts

Consistência

Garante padronização ao usar o mesmo template

 Use variáveis quando precisar criar prompts que serão utilizados repetidamente com diferentes parâmetros.

Prompt do Sistema:

Definindo o Comportamento do AI

Definição

Instruções iniciais que estabelecem como o modelo de IA deve se comportar e responder durante toda a interação. É a primeira mensagem enviada ao modelo e serve como uma configuração fundamental que influencia todas as respostas subsequentes. Funciona como uma "personalidade" ou "conjunto de regras" que o modelo seguirá consistentemente.

Componentes Principais

Inclui definição de persona (ex: especialista em marketing, professor, consultor), tom de voz (formal, casual, técnico), formato de resposta (estrutura específica, limite de palavras) e limitações específicas que o modelo deve seguir. Também pode incluir instruções sobre o uso de emojis, formatação especial, ou restrições sobre certos tipos de conteúdo. A clareza destes componentes é crucial para obter resultados consistentes.

Benefícios

Garante consistência nas respostas, mantém o foco no objetivo e evita comportamentos indesejados do modelo. Aumenta a precisão e relevância das respostas, reduz a necessidade de correções frequentes, e permite criar interações mais naturais e contextualizadas. Também ajuda a manter a conformidade com diretrizes específicas e requisitos do projeto.

Práticas Recomendadas

Seja específico, claro e direto nas instruções. Defina limites explícitos e forneça exemplos quando necessário. Evite instruções contraditórias ou ambíguas. Teste diferentes variações do prompt do sistema para encontrar a que melhor atende às suas necessidades. Mantenha um registro das versões que funcionaram melhor para referência futura. Atualize o prompt do sistema conforme necessário com base no feedback e resultados obtidos.

 Um bom prompt do sistema funciona como uma bússola, guiando todas as interações subsequentes com o modelo de IA. É o alicerce sobre o qual toda a conversa será construída, determinando não apenas o que o modelo diz, mas como ele interpreta e responde às entradas do usuário. Investir tempo na criação de um prompt do sistema eficaz pode economizar muito esforço posterior e garantir resultados mais consistentes e satisfatórios.

Técnicas Básicas

Prompt Zero-Shot

Muitas vezes, a solução mais simples é a melhor. Quando se trata de engenharia de prompt, é importante não tornar as coisas mais complicadas do que o necessário.

O Zero-Shot, uma técnica básica de engenharia de prompt, atenderá às necessidades da grande maioria dos casos. Essa abordagem consiste em fornecer instruções claras e diretas, sem sobrecarregar o prompt com informações desnecessárias.

Em vez de tentar criar prompts excessivamente complexos, é melhor focar em construir prompts eficientes e fáceis de entender. Isso ajuda a garantir que o modelo de linguagem possa interpretar corretamente as instruções e gerar respostas relevantes.

Claro, existem situações em que técnicas mais avançadas de engenharia de prompt podem ser necessárias. No entanto, é importante não perder de vista a simplicidade como um princípio fundamental. Mantenha as coisas simples sempre que possível, pois isso tende a funcionar melhor na maioria dos casos.



Instrução Direta

O modelo recebe uma única instrução clara e objetiva, sem necessidade de contexto adicional

Exemplo: "**Traduza o seguinte texto para inglês: 'Bom dia, como você está?'**"

Resposta Imediata

A IA gera uma resposta apropriada baseada apenas na instrução fornecida

Exemplo: "**Resuma os principais pontos deste artigo em 3 tópicos.**"

Tarefas Simples

Ideal para comandos diretos onde não é necessário fornecer exemplos específicos

Exemplo: "**Crie uma lista de 5 exercícios para fortalecimento abdominal.**"

Estímulo Direcional

Referência: <https://arxiv.org/abs/2302.11520>

- Forneça palavras-chave relevantes para direcionar o modelo.
 - Ajuda a obter resultados específicos com menos texto.
 - Utilize termos que descrevam o que você deseja.
 - O modelo usa essas dicas para entender sua intenção.

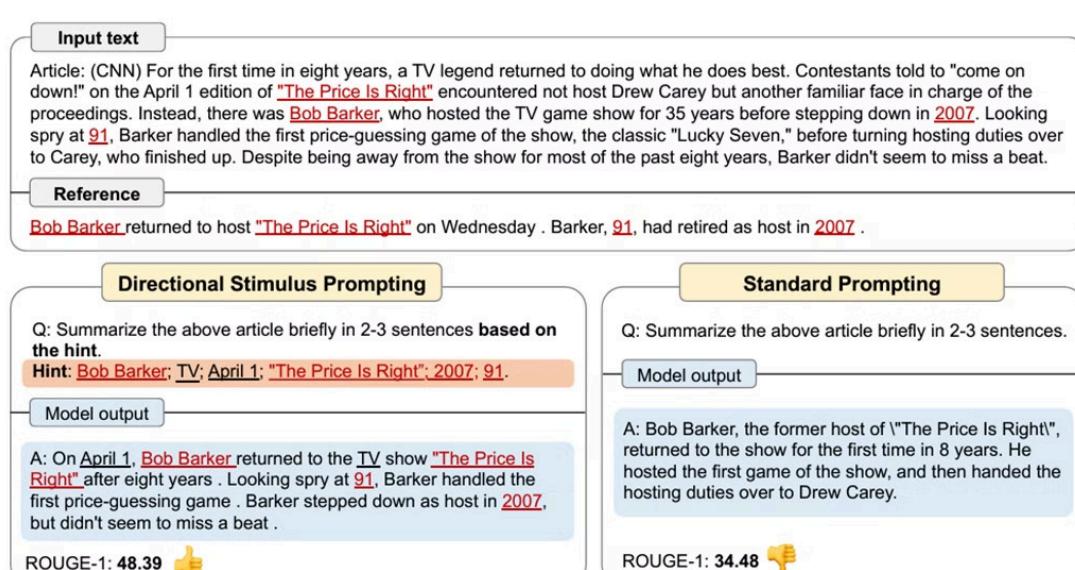


Figure 1: Comparison of our Directional Stimulus Prompting and the standard prompting method using LLMs such as ChatGPT for the summarization task. DSP utilizes directional stimulus/hints (highlighted in orange), which are keywords in this case, to provide instance-specific guidance to LLMs in generating summaries (highlighted in blue) that better align with the desired reference summary with higher ROUGE scores or other measures like human preferences.

ling Large Language Model irectional Stimulus Prompt

Peng², Pengcheng He², Michel Galley², Jianfeng Gao¹
University of California, Santa Barbara¹
Microsoft²
(zekunli, xyan}@cs.ucsb.edu
eng, penhe, mgalley, jfgao}@microsoft.com

Abstract

Directional Stimulus Prompting, a novel framework for guiding language models (LLMs) toward specific desired behaviors. In contrast to existing LLMs, our method employs a small tunable policy model to generate an auxiliary *directional stimulus prompt* for each dialogue. These auxiliary prompts act as nuanced, instance-specific directional stimuli that guide the LLM in generating desired outcomes, such as including specific information or providing a detailed summary. Our approach sidesteps the challenge of directly modifying the LLM’s internal policy by instead reusing the LLM’s internal policy model to explore directional stimulus prompts that elicit the desired behaviors. The policy model can be optimized via two main paths: 1) fine-tuning using labeled data and 2) reinforcement learning, where the policy model receives rewards based on the LLM’s output. We assess our framework’s performance across three tasks: dialogue response generation, and chain-of-thought reasoning. Our experiments demonstrate that the framework consistently improves upon state-of-the-art LLMs (e.g., GPT-3, Codex, InstructGPT) performance on these super-challenging tasks, even when trained on a small amount of data. Notably, using just 80 dialogues on the MuSTC dataset, our framework enhances ChatGPT’s performance by an impressive 10% on the Chain-of-Thought reasoning task, surpassing some fully supervised start-of-the-art models. A key advantage of our framework is that it generates a single, generic chain-of-thought prompt generated by our approach, which can be used to improve the reasoning accuracy of any LLM, including those that do not support chain-of-thought reasoning. The code and data are publicly available.³

A paradigm has emerged in natural language processing models (LLMs) such as Codex [9], InstructGPT, ChatGLM-6B [10], and Qwen [11]. These models exhibit emergent abilities [68] such as text completion and text-to-speech capabilities, which were not present in previous models like GPT-3 [12], BART [13], RoBERTa [37], GPT-2 [52], and T5 [53]. This marks significant advancements in NLP, with LLMs demonstrating improved performance across various tasks. However, these models also raise concerns regarding commercial considerations and the risk of misuse, particularly if they can be easily integrated into existing systems. To mitigate these risks, researchers and developers often choose to keep LLMs as black-box models, fine-tuning them for specific tasks or use cases. In this scenario, the standard approach for utilizing LLMs involves generating text prompts to query LLMs through black-box interfaces.

done when Zekun Li was interning at Microsoft Research.

zehn Directional-Stimulus-Presenting

Prompting de Poucos Exemplos (Few-Shot)

O prompting de poucos exemplos (few-shot) consiste em fornecer ao modelo de IA alguns exemplos, normalmente entre 1 e 5, de entradas e saídas desejadas no prompt. Essa técnica melhora o entendimento da tarefa pelo modelo, levando a respostas mais precisas e consistentes. A escolha dos exemplos é crucial, pois devem ser representativos da tarefa.

Entrada: "O céu está azul"

Saída: "The sky is blue"

Entrada: "Eu gosto de café"

Saída: "I like coffee"

Entrada: "Bom dia, como vai você?"

Saída: "Good morning, how are you?"

Agora traduza: "Obrigado pelo seu tempo"

Técnicas Básicas

Chain-of-Thought (CoT)

Referência: <https://arxiv.org/abs/2201.11903>

- A técnica Chain-of-Thought (CoT) guia o modelo a explicar passo a passo seu raciocínio, melhorando a qualidade das respostas.
 - Essa técnica é especialmente útil para tarefas complexas que exigem lógica e resolução de problemas.
 - Ao fornecer exemplos de raciocínio, você pode melhorar a precisão das respostas do modelo.
 - Com exemplos práticos, é possível observar a aplicação da CoT em cenários reais, como debugging de código.

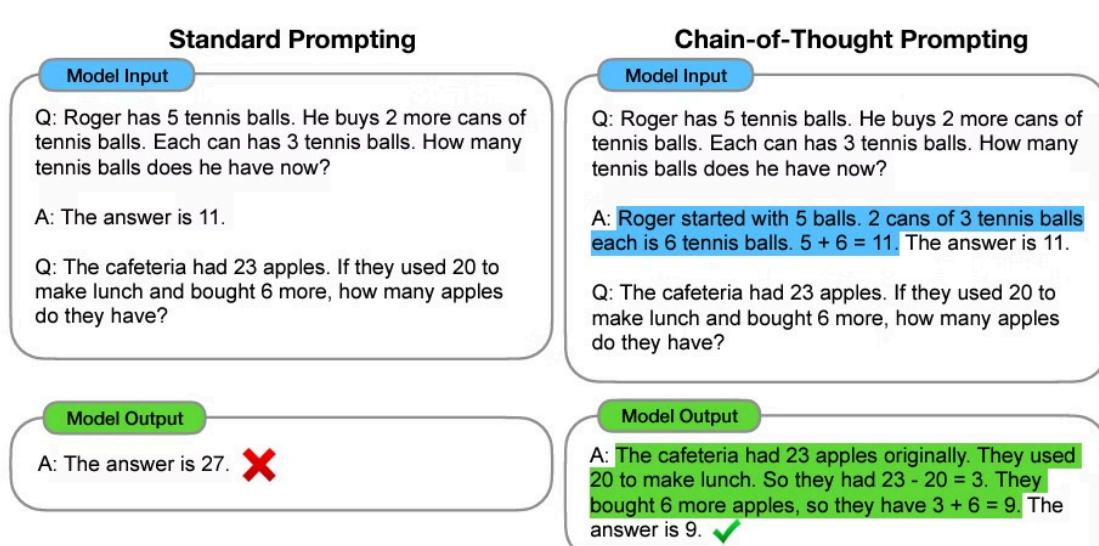


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Thought Prompting Elicits Large Language Model

Jezhi Wang Dale Schuurmans

Fei Xia, Ed H. Chi, Quoc V. Le

Google Research, Brain Team
jasonwei, donnyzhou@google.com

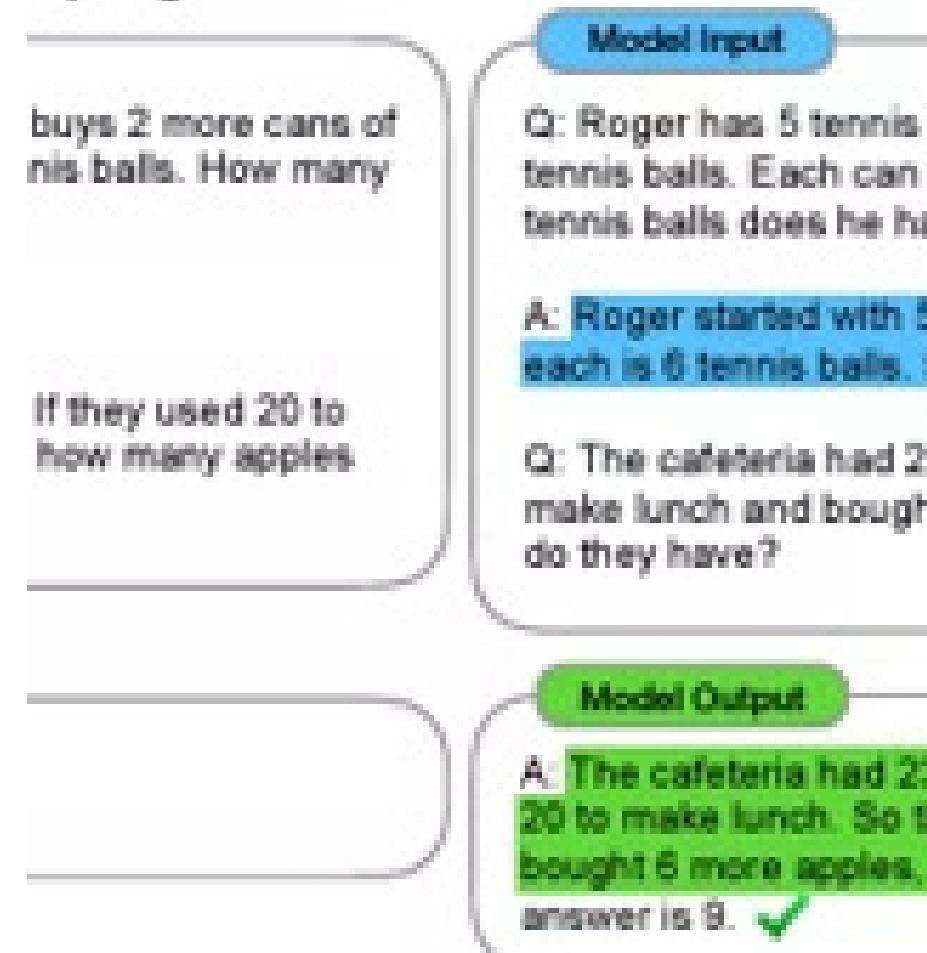
Abstract

rating a *chain of thought*—a series of intermediate steps that improves the ability of large language models to reason. In particular, we show how such reasoning can be achieved by large language models via a simple mechanism. We present here a few chain of thought demonstrations.

large language models show that chain-of-thought reasoning can be effective on a range of arithmetic, commonsense, and logic reasoning tasks. These empirical gains can be striking. For instance, the LLaMA-13B model achieves 81.8% accuracy on the MSRAutoMath 1K benchmark of math word problems, while a verifier achieves 61.8%.

mapping

Chain-of-T



prompting enables large language models to reason on tasks. Chain-of-thought reasoning

Chain-of-Thought (CoT) - Exemplo

Resolva este problema explicando seu raciocínio passo a passo:

"João tem 27 laranjas. Ele dá 1/3 das laranjas para Maria e depois compra mais 15 laranjas. Quantas laranjas João tem agora?"

Vamos resolver passo a passo:

1. Primeiro, calculamos quantas laranjas João deu para Maria

- João tinha 27 laranjas

- $1/3 \text{ de } 27 = 9$ laranjas foram para Maria

2. Então, calculamos quantas laranjas sobraram com João

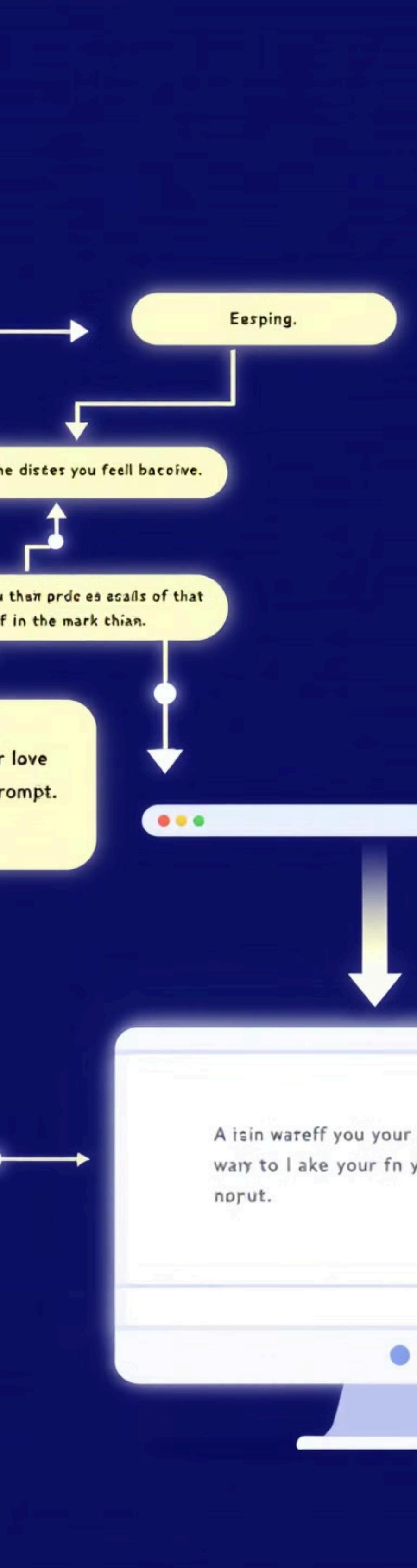
- $27 - 9 = 18$ laranjas restantes

3. Por fim, adicionamos as novas laranjas

- $18 + 15 = 33$ laranjas

Portanto, João tem agora 33 laranjas.

Técnicas Avançadas



1

Refinamento e Iteração

A prática de melhorar prompts é similar ao treinamento de IA. Use feedback das respostas para ajustar e refinar os prompts.

2

Cadeia de Pensamento

Oriente o modelo a apresentar raciocínios passo a passo para problemas complexos. A técnica reduz erros e aumenta a precisão.

Técnicas Avançadas

Self-Consistency

Referência: *Self-Consistency Improves Chain of Thought Reasoning in Language Models* - <https://arxiv.org/abs/2203.11171>

- **Self-Consistency:** É uma técnica que gera múltiplas cadeias de pensamento para a mesma tarefa e depois faz com que o modelo escolha a resposta mais consistente.
- Essa técnica ajuda a reduzir erros e melhorar a qualidade das respostas, aproveitando a capacidade do modelo de avaliar seu próprio raciocínio.
- Ao analisar várias cadeias de pensamento, o modelo pode identificar e eliminar respostas inconsistentes ou contraditórias.
- A técnica de Self-Consistency permite que o modelo aprenda com seus próprios erros, aumentando a precisão e confiabilidade das respostas.

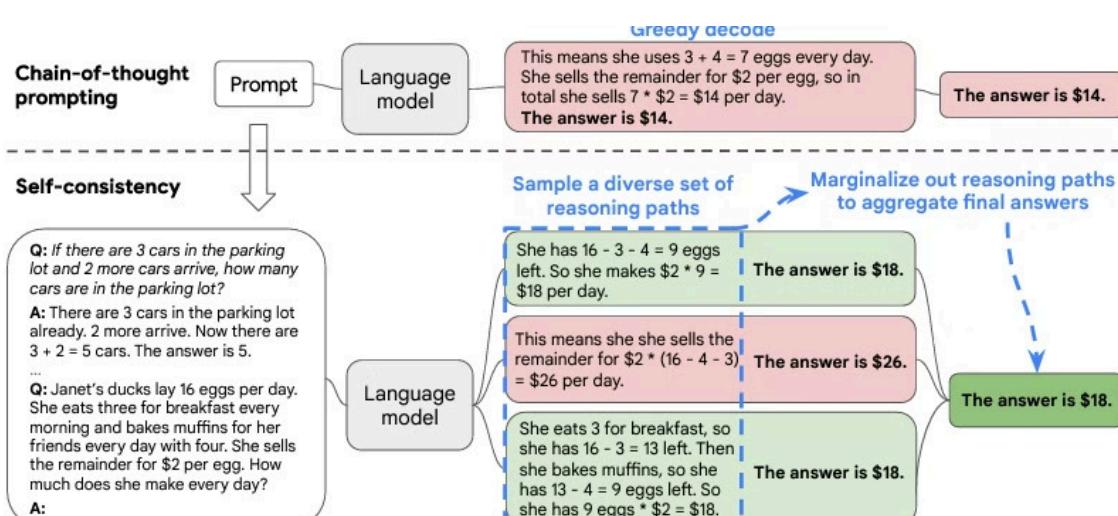


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the “greedy decode” in CoT prompting by sampling from the language model’s decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

TENACY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Jun Wei[†] Dale Schuurmans[†] Quoc Le[†] Eshanksha Chowdhery[†] Denny Zhou^{†§}
Google Brain Team
zhenwei@google.com, dennyyzhou@google.com

ABSTRACT

Chain-of-thought (CoT) prompting combined with pre-trained large language models has shown promising results on complex reasoning tasks. In this paper, we propose a new strategy, *self-consistency*, to replace the naive greedy decoding in CoT prompting. It first samples a diverse set of reasoning paths, then marginalizes out the sampled reasoning paths. Self-consistency improves the performance of chain-of-thought reasoning on complex reasoning problem typically admits multiple reasoning paths leading to its unique correct answer. Our extensive experiments show that self-consistency boosts the performance of chain-of-thought reasoning on a range of popular arithmetic and common sense reasoning benchmarks, including GSM8K (+17.9%), SVAMP (+11.0%), ARITH (+4.4%) and ARC-challenge (+3.9%).

Reasoning is a key capability of large language models. While they have demonstrated remarkable success across a wide range of tasks, reasoning is often seen as a limitation, which can be mitigated by various prompting strategies. Rae et al., 2021; BIG-bench collaboration, 2022; Wei et al., 2022 have proposed *chain-of-thought* prompting to generate a series of short sentences that the language model can employ in solving a task. For example, given the question “*If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?*”, a language model would be prompted with “*Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot? A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.*”. It has been observed that chain-of-thought prompting outperforms baseline prompting across a variety of multi-step reasoning tasks.

While chain-of-thought prompting has shown promising performance, it is still limited by the inherent greediness of the language model’s decoder. To address this, we propose a novel decoding strategy called *self-consistency* in chain-of-thought prompting (Wei et al., 2022). We show that self-consistency improves reasoning performance by a significant margin. Self-consistency is particularly useful for reasoning tasks that typically admit multiple reasoning paths (Hawkins et al., 2000). The more that deliberate thinking and reasoning are required, the greater the diversity of reasoning paths that can be generated by the language model.

We demonstrate the effectiveness of self-consistency with an example. We first prompt the language model with the question “*If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?*”, then instead of greedily decoding the entire sequence, we use a “*self-consistency*” decoding procedure: we first sample a diverse set of reasoning paths; each reasoning path is then decoded to find the optimal answer by *marginalizing out* the other paths. Finally, we choose the most consistent answer in the final answer set. Such an approach allows the language model to explore multiple different ways of thinking lead to the same answer, and thus increase the probability that the generated answer is correct. Compared to other decoding strategies such as beam search and local-optimality that plague greedy decoding, self-consistency is more robust and less prone to errors due to the amplified generation.

Self-Consistency - Exemplo Prático

"Qual é o lucro total de uma loja que vendeu 150 produtos a R\$45 cada, tendo um custo de R\$30 por produto?"

Vamos resolver passo a passo:

1. Receita total: $150 \times R\$45 = R\6.750 2. Custo total: $150 \times R\$30 = R\4.500 3. Lucro = $R\$6.750 - R\$4.500 = R\$2.250$

2. Lucro por produto: $R\$45 - R\$30 = R\$15$ 2. Lucro total: $150 \times R\$15 = R\2.250

3. Receita: $150 \text{ produtos} \times R\$45 = R\$6.750$ 2. Custo por produto: $R\$30$ 3. Custo total: $150 \times R\$30 = R\4.500 4. Lucro = Receita - Custo = $R\$2.250$

Observação: Os três caminhos de raciocínio chegaram à mesma resposta (R\$2.250), demonstrando consistência no resultado final. Esta é a essência da técnica Self-Consistency: múltiplos caminhos de raciocínio validam a resposta.

Tree-of-Thought (ToT)

Tree of Thoughts: Deliberate Problem Solving with Large Language Models - <https://arxiv.org/abs/2305.10601>

Generative Agents: Interactive Simulacra of Human Behavior - <https://arxiv.org/abs/2305.08291>

- A técnica Tree-of-Thought expande a ideia da Chain-of-Thought, gerando múltiplos pensamentos e desenvolvendo uma árvore de raciocínio.
- O modelo explora diferentes caminhos de raciocínio e escolhe o mais promissor para chegar à resposta final. Isso é útil para problemas complexos que podem ter várias abordagens possíveis.

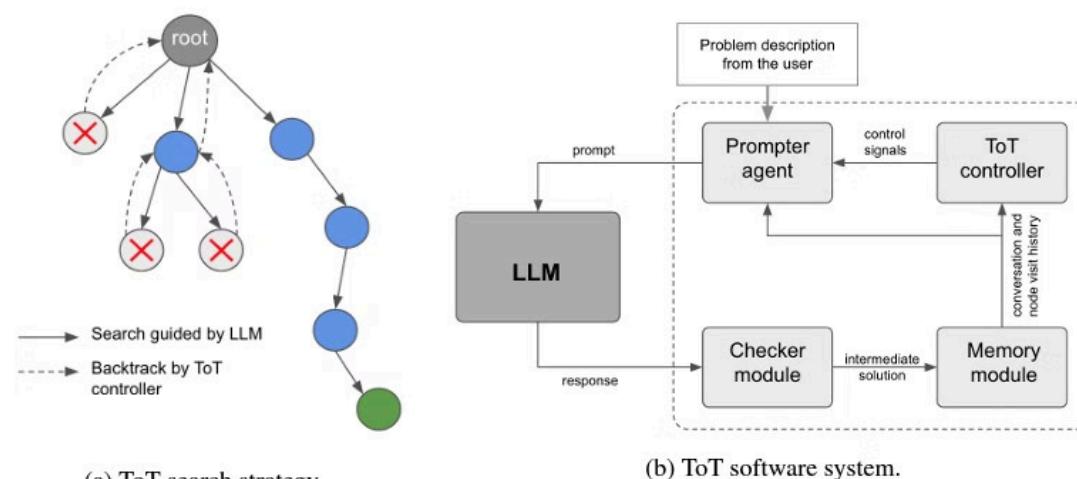
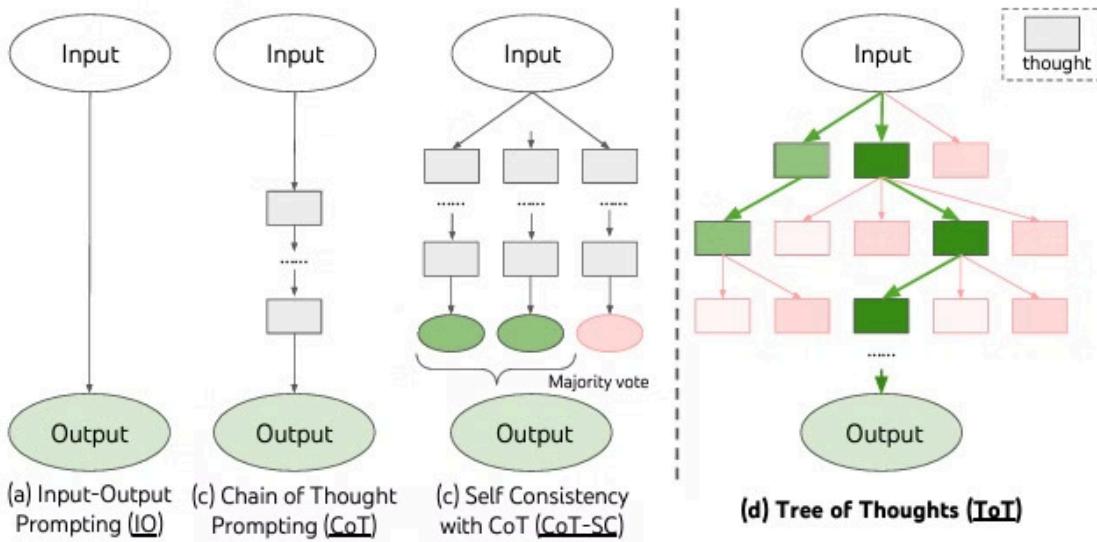


Figure 1: (a) Details of the Tree-of-Thought search strategy, where a solid arrow means a search step guided by the response from the LLM, and a dashed arrow indicates backtracking commanded by the ToT controller. (b) The software system implementing the Tree-of-Thought search strategy. It enhances the problem solving capability of an LLM by augmenting it with additional modules including a promoter agent, a checker module, a memory module, and a ToT controller.

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Shunyu Yao, Princeton University; Dian Yu, Google DeepMind; Jeffrey Zhao, Google DeepMind; Izah Shafrazi, Google DeepMind

Thomas L. Griffiths, Princeton University; Yuan Cao, Google DeepMind; Karthik Narasimhan, Princeton University

Abstract

Language models are increasingly being deployed for general problem solving across a wide range of tasks, but are still confined to token-level, left-to-right decision-making processes that often fail to find solutions in tasks that require exploration, strategic lookahead, or where initial decisions play a critical role. To overcome these challenges, we introduce a new framework for

Large Language Model Guided Tree-of-Thought

Jiayi Long, Theta Labs, Inc., San Jose, CA 95128, jiayi@theta-labs.org

Abstract

In this paper, we introduce the Tree-of-Thought (ToT) framework, a novel approach aimed at improving the problem-solving capabilities of auto-regressive large language models (LLMs). The ToT technique is inspired by the human mind's approach for solving complex reasoning tasks through trial and error. In ToT, the LLM performs a search process that explores multiple paths simultaneously, allowing for backtracking when necessary. To implement ToT as a software system, we augment an LLM with additional modules including a promoter agent, a checker module, a memory module, and a ToT controller. In contrast to standard LLMs, ToT requires explicit mechanisms to record conversation history of the problem solving process, which allows the system to backtrack to the previous steps of the thought-process and explore other directions from there. To demonstrate the effectiveness of the proposed technique, we built a ToT-based solver for the Sudoku Puzzle. Experimental results show that the ToT framework can significantly increase the success rate of Sudoku puzzle solving. Our implementation of the ToT-based Sudoku solver is available on GitHub: <https://github.com/jiayilong/tree-of-thought-puzzle-solver>.

1 Introduction

Self-attention based auto-regressive large language models (LLMs) such as GPT-4 have recently taken the world by storm [1, 2, 3, 4, 5, 6]. These LLMs excel at a variety of tasks that previously thought as extremely difficult or even impossible. For example, they are able to handle various logical and mathematical reasoning tasks, particularly those that entail “short-range reasonings” necessitating only a few steps to arrive at conclusions [6, 7]. Such remarkable capabilities have even led to significant breakthroughs in areas like general knowledge, common sense, and many already emerge [1, 2, 3, 4]. However, today’s LLMs still exhibit limitations in certain domains, especially for “long-range” reasoning tasks, where long-term planning and solution exploration are necessary [7]. When presenting a LLMs such as GPT-4 with a challenging problem solving task, especially the so called System-2 reasoning problems [8], the model does not always succeed. Although the generated answer may be indicative of the correct direction, the derivation process frequently includes logical errors. We hypothesize that there are two main contributing factors which limits the problem solving ability of LLMs:

Lack of correctness checking: To ensure correctness, a good practice for a human solver is to carry out verification procedures at every step of the problem-solving process, thereby ensuring the credibility of the final solution. In comparison, auto-regressive language models do not explicitly perform logical correctness checks as it generates a new token based on the previous tokens. This limits the model’s capacity to rectify its own mistakes. A minor error could be amplified as the model generates more tokens, thereby leading to rapid solution quality deterioration and making it difficult to recover from mistakes.

Preprint. Under review.

Tree-of-Thought (ToT) - Exemplo Prático

Podemos configurar três assistentes GPT especializados, utilizando a tecla @ para ativar um segundo agente que valida e complementa a resposta do primeiro, e, em seguida, um terceiro que analisa e refina o resultado final.

Primeiro Assistente

(@Copywriter): Focado em gerar ideias criativas e persuasivas.

Comando:

@Copywriter: Sugira três títulos criativos e impactantes para um artigo sobre "A Importância da Sustentabilidade Empresarial".

Resposta do Copywriter:

- "Sustentabilidade Empresarial: O Caminho para o Futuro Verde"
- "Como Empresas Inteligentes Transformam Sustentabilidade em Lucros"
- "O Impacto da Sustentabilidade nos Negócios de Hoje e Amanhã"

Segundo Assistente

(@Editor): Avalia e valida as sugestões com base na clareza e impacto.

Comando:

@Editor: Avalie os títulos sugeridos pelo @Copywriter e escolha o mais atrativo, sugerindo ajustes para melhorar a clareza.

Resposta do Editor:

- Título escolhido: "Como Empresas Inteligentes Transformam Sustentabilidade em Lucros".
- Ajuste sugerido: Adicionar uma referência ao impacto positivo no meio ambiente, tornando-o mais inclusivo.

Nova versão: "Como Empresas Inteligentes Transformam Sustentabilidade em Lucros e Beneficiam o Planeta".

Terceiro Assistente

(@Gerente): Refina o título com uma perspectiva estratégica de marketing.

Comando:

@Gerente: Analise o título ajustado pelo @Editor e sugira melhorias para torná-lo mais estratégico e relevante para o público-alvo.

Resposta do Gerente:

- Título ajustado: "Sustentabilidade Inteligente: Como Empresas Transformam Lucros e Beneficiam o Planeta".
- Justificativa: Combina apelo estratégico (lucros) com responsabilidade ambiental, atraindo tanto executivos quanto consumidores conscientes.

Técnicas Avançadas

Skeleton-of-Thought (SoT)

Referência: Skeleton of Thought: Augmenting Language Models with Variable-Depth Reasoning Ability - <https://arxiv.org/abs/2307.15337>

- A técnica Skeleton-of-Thought (SoT) envolve a criação de um esqueleto ou índice de tópicos antes de desenvolver o conteúdo completo.
- Isso ajuda a estruturar a resposta e garantir que todos os pontos-chave sejam abordados.
- O modelo primeiro gera o esqueleto e, em seguida, preenche cada tópico com detalhes.
- Essa abordagem permite uma organização mais eficiente do raciocínio, levando a respostas mais completas e coerentes.

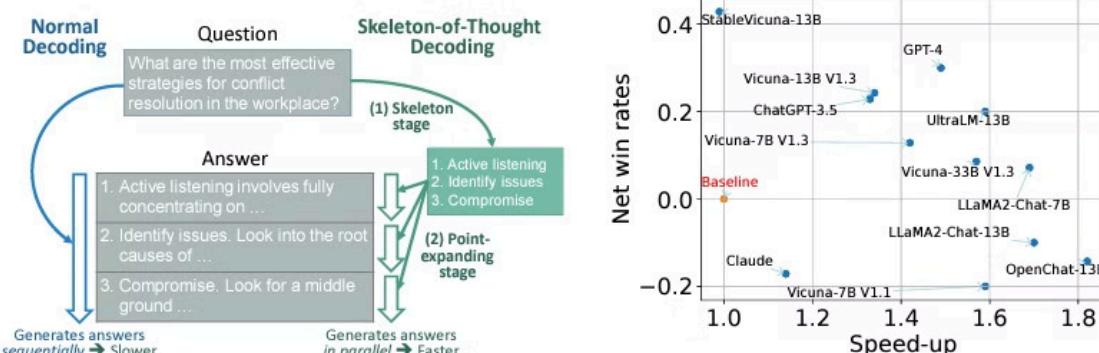


Figure 1: **Left:** An illustration of Skeleton-of-Thought (SoT). Instead of producing answers sequentially, SoT produces different parts of answers *in parallel*. In more detail, given the question, SoT first prompts the LLM to give out the skeleton, then conducts batched decoding or parallel API calls to expand multiple points in parallel, and finally aggregates the outputs to get the final answer. **Right:** The net win rates and speed-ups of SoT with router (SoT-R) compared to normal generation on Vicuna-80. The net win rate is the difference between the fraction of questions that SoT-R has better and worse answers than normal generation. The speed-up is the ratio between the latency of normal and SoT-R generation. (1.0, 0.0) represents normal generation. Higher is better on both axes. For most models, SoT-R not only accelerates the generation but also improves the quality of the answers (evaluated with FastChat metric (Zheng et al., 2023)). See § 3.2 and 4 for more details.

F-THOUGHT: PROMPTING PARALLEL GENERATION

1 . com

tsinghua.edu.cn

.edu.cn

Yu Wang¹
yu-wang@tsinghua.

¹ Institute of Electronic Engineering, Tsinghua University, Beijing, China
² Microsoft Research, Redmond, Washington, USA
³ ESAT-PSI, KU Leuven, Leuven, Belgium
⁴ Infinigence-AI

URL: <https://sites.google.com/view/sot>
<https://github.com/imagination-research/F-THOUGHT>

ABSTRACT

Large language models (LLMs) have shown remarkable performance in various tasks, but their inference efficiency remains a bottleneck. One major cause is the sequential generation process, which is slow and lacks parallelism. To address this, we propose F-THOUGHT, a novel prompting paradigm that leverages the parallel generation capability of LLMs. F-THOUGHT consists of two main components: a skeleton stage and a parallel generation stage. In the skeleton stage, the LLM generates a sparse representation of the response (the skeleton), which is then expanded in parallel across multiple API calls or parallel decoding threads. This results in a significant reduction in end-to-end latency while maintaining or improving the quality of the generated responses. We evaluate F-THOUGHT on 12 different LLMs and show that it can achieve up to 1.7 times speed-up and 0.35 net win rate improvement over baseline methods. Our results demonstrate that F-THOUGHT is a promising direction for improving the inference efficiency of LLMs.

1

(LLMs) (Brown et al., 2020; Touvron et al., 2023a; Chen et al., 2023) have shown exceptional performance in natural language processing. However, the inference process of the state-of-the-art LLMs is still slow. For example, it takes 22 seconds for Claude (Anthropic, 2023) and 10 seconds for Vicuna-33B V1.3 (a 33B LLaMA-based model) to answer the question in Fig. 1.

There are several causes of LLMs' slow inference: (1) A large model size leads to increased memory access, computation, and communication costs. For example, the FLOPs required for a 33B LLaMA model (which means at least $5 \times 80\text{GB}$ A100 GPUs are needed) are high. Even with enough GPUs, the heavy memory access and computation costs are due to the attention operation in the prevailing transformer architecture. (2) The memory and computation complexity in sequence length scaling. As the sequence length increases, the inference generates tokens one by one. This approach is inefficient because the generation of tokens cannot be parallelized. There are two main axes: *large model size* (Xiao et al., 2022; Frantar et al., 2023; Chen et al., 2023) and *attention operation* (Kitayev et al., 2022).

Técnicas Avançadas

Skeleton-of-Thought (SoT) - Exemplo - Passo 1

Passo 1: Geração da Estrutura Base

Solicitamos ao modelo que gere uma lista de possíveis causas surpreendentes relacionadas ao impacto da alimentação na produtividade.

Comando:

Liste 5 causas surpreendentes que expliquem como a alimentação impacta diretamente a produtividade no trabalho. Use uma abordagem inesperada ou pouco explorada.

Resposta do GPT:

1. A desidratação leve causada pela falta de frutas suculentas reduz a concentração em até 20%.
2. O consumo de alimentos ultraprocessados aumenta a inflamação cerebral, prejudicando decisões rápidas.
3. Uma dieta rica em ômega-3 melhora a comunicação entre neurônios, acelerando a resolução de problemas.
4. Comer em intervalos regulares regula o nível de cortisol, prevenindo o esgotamento mental.
5. O excesso de cafeína altera o ciclo circadiano, reduzindo a criatividade no período da tarde.

Skeleton-of-Thought (SoT) - Exemplo

- Passo 2

Passo 2: Seleção da Causa

Solicitamos ao modelo que escolha a causa mais interessante e inesperada, explicando o motivo da escolha.

Comando:

Escolha a causa mais surpreendente da lista e explique por que ela seria interessante para uma copy publicitária.

Resposta do GPT:

- **Causa Escolhida:** "A desidratação leve causada pela falta de frutas suculentas reduz a concentração em até 20%."
- **Motivo:** Esta causa é constraintuitiva, já que muitas pessoas não associam a ingestão de frutas à produtividade mental. É simples de explicar e fácil de conectar a um call-to-action para incentivar uma alimentação mais saudável no ambiente de trabalho.

Skeleton-of-Thought (SoT) - Exemplo

- Passo 3

Passo 3: Elaboração sobre a Causa

Solicitamos ao modelo que elabore a causa escolhida, estruturando um argumento persuasivo.

Comando:

Desenvolva a causa escolhida em um argumento persuasivo para uma copy publicitária. Conecte-a com um benefício claro e um call-to-action.

Resposta do GPT:

- **Texto Elaborado:**

"Sabia que uma desidratação leve pode ser a vilã da sua produtividade? Estudos mostram que a falta de frutas suculentas, como maçãs e laranjas, pode reduzir sua concentração em até 20%. Isso porque frutas não apenas alimentam, mas hidratam seu cérebro, ajudando você a tomar decisões rápidas e eficazes. Adote o hábito de incluir frutas frescas no seu dia a dia e sinta a diferença na sua performance. Comece agora com o nosso kit de frutas selecionadas, entregue diretamente no seu escritório!"

Generated Knowledge Prompting

Generated Knowledge Prompting for Commonsense Reasoning -
<https://arxiv.org/abs/2110.08387>

- Essa técnica usa modelos de linguagem para gerar conhecimento contextual adicional, melhorando a qualidade das respostas.
- O modelo gera informações relevantes com base no contexto fornecido, que são incorporadas ao prompt.
- O resultado são respostas mais precisas e informativas.
- Um exemplo prático é a geração de um perfil detalhado de cliente usando o modelo de linguagem.
- Esse perfil gerado é usado para melhorar a geração de um título persuasivo.

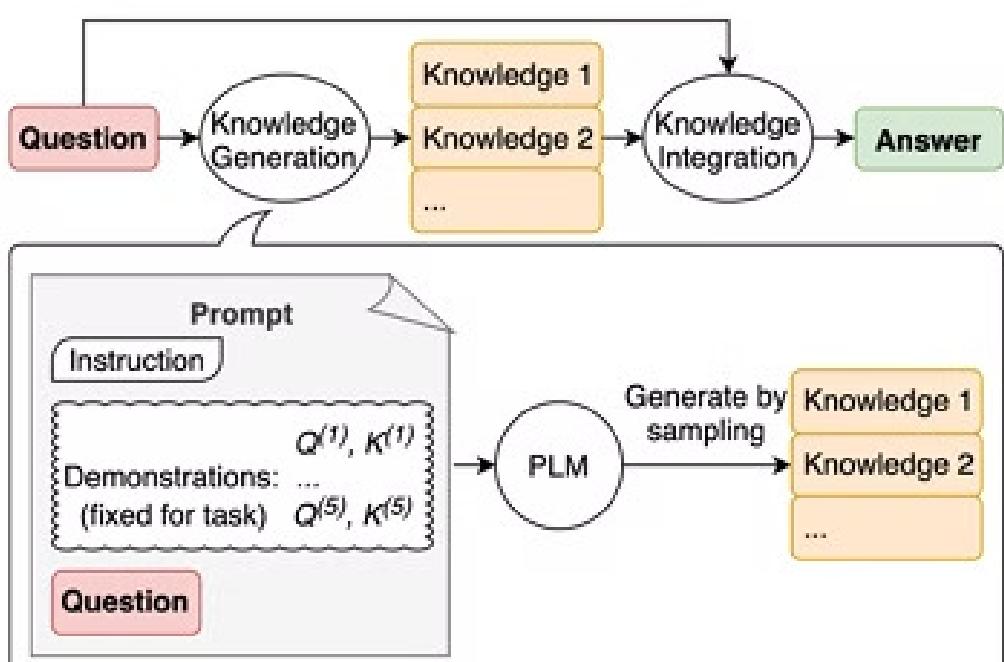


Figure 1: Generated knowledge prompting involves (i) using few-shot demonstrations to generate question-related knowledge statements from a language model; (ii) using a second language model to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

lu[○] Alisa Liu[○] Ximeng Lu[○] Seán
 nan Le Bras[●] Yejin Choi[○] Hanna
 of Computer Science & Engineering, Un
 ● Allen Institute for Artificial Intelligence
 liujc@cs.washington.edu

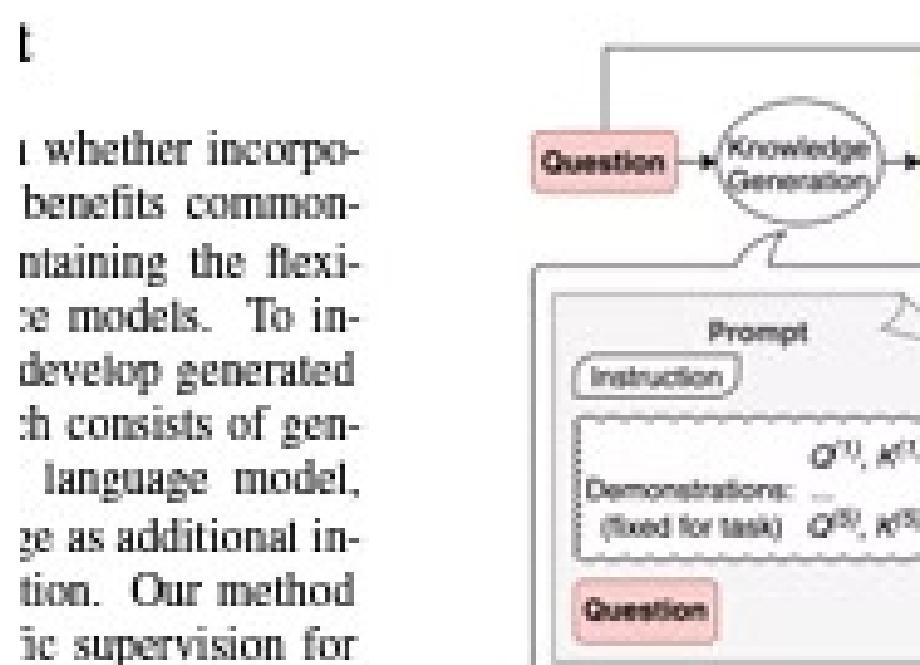


Figure 1: Generated knowledge prompting involves (i) using few-shot demonstrations to generate question-related knowledge statements from a language model; (ii) using a second language model to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

edge, as many benchmarks use knowledge bases. Furthermore, prior methods require custom supervision for each task (Mitra et al., 2019; Chaitanya et al., 2021).

In this paper, we show that generated knowledge can be used for commonsense reasoning, even on top-of-the-art pretrained models (e.g., GPT-3 and its variants), without requiring any zero-shot supervision. We propose an approach to generate knowledge for knowledge integration that is structured and efficient.

The key insight behind Generated Knowledge Prompting is that we can generate knowledge from a language model, then provide a structured and efficient prompt that is compatible with the language model's architecture.

Generated Knowledge Prompting - Exemplo

Imagine que você precisa gerar um título para um artigo sobre a influência da música na produtividade.

Para usar o Generated Knowledge Prompting, primeiro você pode solicitar ao modelo de linguagem que gere um resumo das informações mais relevantes sobre o tema.

Em seguida, use o resumo gerado para criar um prompt que inclua esses detalhes.

"Crie um título cativante para um artigo que aborda como a música influencia a produtividade, considerando os seguintes pontos: (insira o resumo gerado)."

O modelo de linguagem irá usar o resumo gerado como conhecimento contextual adicional, o que pode levar a títulos mais criativos e relevantes.

Prompt Maiêutico

Referência: Maieutic Prompting: Logically Consistent Reasoning with Recursive Explanations - <https://arxiv.org/abs/2205.11822>

Prompt Maiêutico é uma técnica que encoraja o modelo de linguagem a justificar suas respostas, explicando o raciocínio por trás delas.

Essa técnica ajuda a melhorar a qualidade das respostas, pois o modelo é incentivado a fornecer explicações mais detalhadas e lógicas.

O modelo deve ser direcionado a fornecer explicações passo a passo, justificando cada etapa do seu processo de raciocínio.

Essa abordagem torna as respostas mais transparentes e confiáveis, mostrando como o modelo chegou à conclusão.

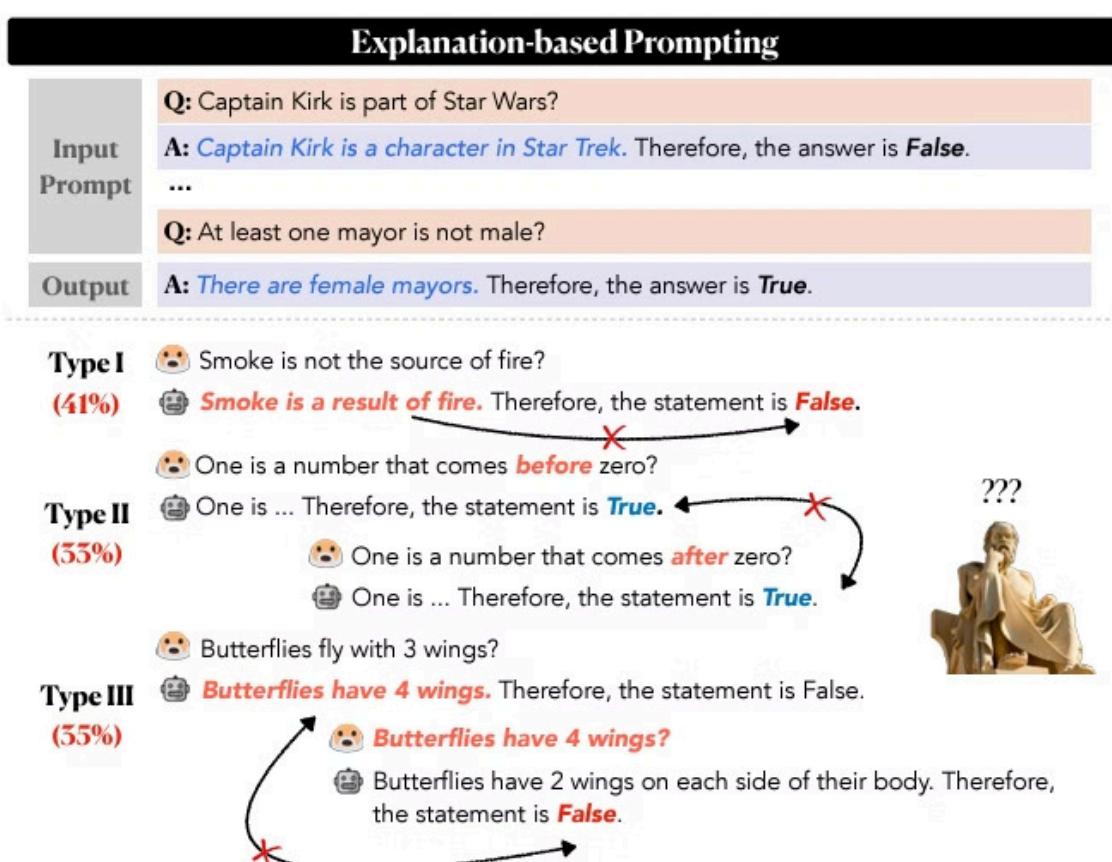


Figure 1: Logical errors in explanation-based prompting: (1) explanation does not logically lead to the answer, (2) model is invariant to negation, and (3) falsifies its own explanation. We prompt 175B GPT-3 with 100 questions sampled from Talmor et al. (2021).

Prompting: Logically Consistent Reasoning with Recursive Explanations

Jun Jung[†] Lianhui Qin[†] Sean Welleck[‡]
Chandra Bhagavatula[†] Ronan Le Bras[‡]
[†]School of Computer Science & Engineering, University of Washington
[‡]Allen Institute for Artificial Intelligence
jhoony123@cs.washington.edu

Abstract

Large Language Models (LMs) struggle recently, prompting them to generate explanations that self-guide them as a promising direction. However, these approaches are often bounded by the limitations of the models, which themselves are inconsistent. In this work, we propose MAIEUTIC PROMPTING, which generates a tree of explanations to answer a question. This approach induces a tree of explanations, then frames the inference problem over these logical relations. We evaluate our method for true/false QA benchmarks that require reasoning. MAIEUTIC PROMPTING achieves up to 20% better performance than state-of-the-art prompting methods, particularly for unsupervised models. MAIEUTIC PROMPTING improves the quality of explanations while providing



Figure 1: Logical errors in explanation-based prompting: (1) explanation does not logically lead to the answer, (2) model is invariant to negation, and (3) falsifies its own explanation. We prompt 175B GPT-3 with 100 questions sampled from Talmor et al. (2021).

problem (Wei et al., 2022), program execution (Nyman et al., 2022),

Explanation-based prompting has been motivated by the reasoning required to employ to solve a problem (Lehnert, 2007). However, this approach is faulty in practice, as generated explanations are often logically inconsistent. For example, we manually sampled 100 questions from a QA task (Figure 1) and found a considerable number of them that does not logically lead to the correct answer. The model infers the statement and its negation (Kassner et al., 2022; Wang et al., 2022), leading to contradictions. These findings raise fundamental questions about the quality of explanations in LMs: is the explanation correct - is there a contradiction? To infer a label consistent with the explanation if the explanation is wrong? To use of even the wrong explanation to find a correct answer?

success of few-shot learning models (e.g. Brown et al., 2022) on prompting methods. The reasoning capability can be improved by generating a sequence of explanations, prior to inferring the final answer (Li et al., 2022; Wang et al., 2022). This approach elicits its knowledge from the user and generates explanations that are more consistent with the user's knowledge. For example, in a math word problem, the user provides a statement and its negation, and the model generates two different explanations for each statement. The user can then choose the correct explanation based on the context of the problem.

<https://github.com/jaejunjung11/Maieutic-Prompting>

To this end, we propose

Prompt Maiêutico - Exemplo

Imagine uma pergunta complexa sobre a história da música brasileira: "Qual a influência da música regional brasileira no samba?"

"Qual a influência da música regional brasileira no samba?"

Com o Prompt Maiêutico, o modelo de linguagem deve fornecer uma resposta detalhada, justificando cada etapa do seu raciocínio.

Por exemplo, o modelo poderia começar descrevendo os principais estilos de música regional, como o forró, o sertanejo e o frevo, e então explicar como cada um deles influenciou o samba.

O objetivo é que o modelo de linguagem não apenas forneça uma resposta, mas também demonstre o processo de pensamento que levou àquela conclusão.

Retrieval Augmented Generation (RAG)

Referência: Retrieval Augmented Generation for Knowledge-Intensive NLP Tasks - <https://aclanthology.org/2020.findings-emnlp.76/>

- **Integração de Conhecimento:** RAG combina modelos de linguagem com bases de conhecimento externas.
 - Acesso a Informações Relevantes: A técnica recupera informações relevantes da base de conhecimento.
 - Respostas Mais Precisas: As informações recuperadas complementam o conhecimento do modelo de linguagem.
 - Respostas Mais Informativas: As respostas geradas são mais precisas e informativas.

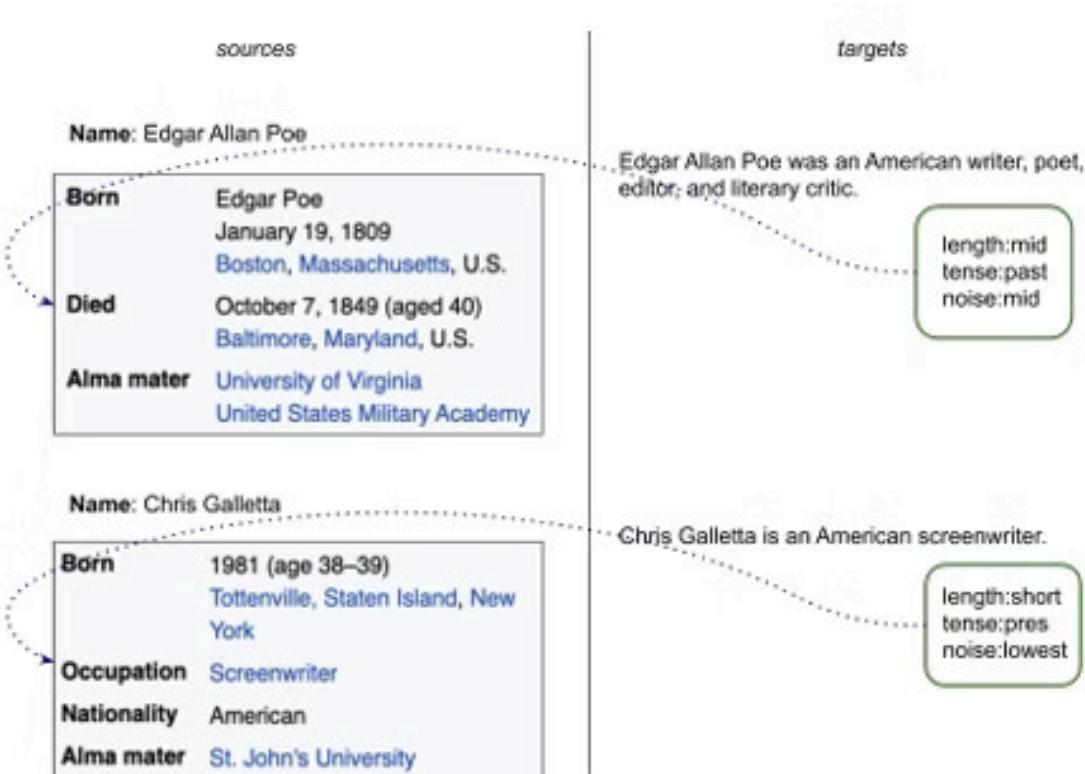


Figure 1: Two WikiBio sources and targets with example attributes: *tense* and *length* can be read-off the target directly. When added to the input, the model gets a knob to control for length and tense. We propose to estimate the *noise* degree by comparing the source with the target thus obtaining a *hallucination knob*.

Controlled Hallucinations: Learning to Generate Faithfully from Noise

Katja Filippova
Google Research, Berlin, Germany
katjaf@google.com

1

lata- or text-to-text) performance when which for many ap- To collect a large iristic rules are often t noise into the data, put which cannot be Consequently, mod- and may hallucinate- jected text. Our com- powerful technique to as a *controllable as-*, without dismissing odifying the model iBio corpus (Lebret ly noisy dataset, we if the technique both uman evaluation.

DNN models have performed well on a multi-task learning setting (Vinyals et al., 2015; Rothe et al., 2018; Rush et al., 2015, 2017). In text generation tasks (Vinyals et al., 2019), in contrast, DNN models often fail to produce text which is normally expected, it is rare to have a large annotated corpus of parallel text pairs which researchers have come up with (Rush et al., 2015; Rush et al., 2017). No matter how powerful a model is, it needs to be sensitive to data (Vinyals et al., 2018) and pick out the most likely output from the generated candidates.

e not been defined formally used to refer to the either unfaithful to the



Figure 1: Two WikiBio sample attributes: *noise* and *get* directly. When added a knob to control for len estimate the *noise* degree the target thus obtaining

input, or nonsensical (Liu et al., 2019). In our work we are concerned with the first two connection kind which is primarily due to the quality of the training data. One may try to improve the quality of the training data by how can one reduce the noise in the input? One may try to improve the quality of the training data by and clean it from phrasal errors. Another approach is to check if some information in the input is missing. This approach is useful when the information found only in the input is missing. The latter approach is risky as it easily leads to wrong targets. The latter approach is useful when the alignment between input and output is not good. This approach has been tried previously but it also leads to wrong targets due to noise in the data (Nikolic et al., 2019). Alternatively, one may try to improve the quality of the training data by and try to put more pre-processed data in the input. This approach has been tried previously but it also leads to wrong targets due to noise in the data (Nikolic et al., 2019). This reduces the number of wrong predictions to the model and makes the decoder to generate fluent and meaningful sentences in the targets.

Técnicas Avançadas

Exemplo de RAG

Imagine um sistema de atendimento ao cliente que usa RAG para responder às perguntas dos usuários.

O sistema pode acessar um banco de dados de perguntas frequentes (FAQs), artigos e manuais, recuperando informações relevantes para a pergunta do usuário.

O modelo de linguagem, com acesso a essas informações, gera uma resposta completa e personalizada para o usuário.

A integração entre RAG e o modelo de linguagem oferece uma experiência mais eficiente e precisa para o usuário, fornecendo respostas mais relevantes e completas.

PAL (Program-Aided Language Models)

Referência: PAL: Program-Aided Language Models -

<https://arxiv.org/abs/2211.10435>

- PAL permite integrar conceitos de programação, como variáveis e funções, nos prompts.
- Prompts mais modulares, reutilizáveis e adaptáveis.
- Exemplo: criação de variáveis para personalizar templates de anúncios.
- As variáveis, como "[público-alvo]" e "[promessa principal]", são definidas no início do prompt e depois usadas no template.
- Gerando anúncios personalizados com base nas variáveis definidas.

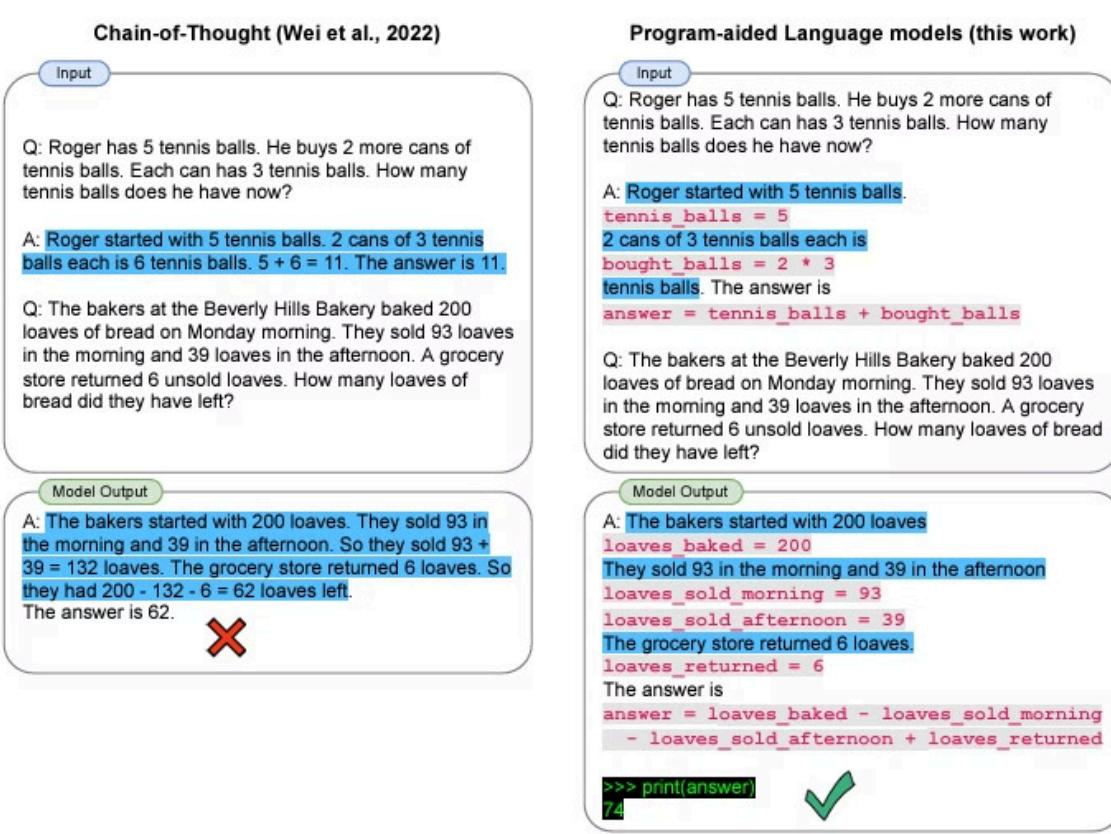


Figure 1: A diagram illustrating PAL: Given a mathematical reasoning question, Chain-of-thought (left) generates intermediate reasoning steps of free-form text. In contrast, Program-aided Language models (PAL, right) generate intermediate steps and Python code. This shifts the role of running the reasoning steps from the language model to the Python interpreter. The final answer is obtained by running the generated reasoning chain. Chain-of-thought reasoning is highlighted in blue; PAL steps are highlighted in gray and pink; the Python interpreter run is highlighted in black and green.

Program-aided Language Models

huyan Zhou^{*1} Uri Alon¹ Pengfei Liu^{1,2}
Graham Neubig^{1,2}

¹zhu, ualon, pliu3, yiming, callan, graham

1. Introduction

Until as recently as two years ago, large language models (LLMs) were considered to be one of the most impressive AI systems ever created. Their success can be attributed to their ability to learn complex patterns in large amounts of text, such as “chain-of-thought” reasoning (Garcez & Lambrix, 2020), impressive success on various benchmarks (Wei et al., 2022), and their ability to generate sensible (Wei et al., 2022), mathematically correct (Ling et al., 2022), and even interesting (Yao et al., 2022) responses via simple prompting (Brown et al., 2021).

This process has been made possible by shifting the burden of generating the “chain-of-thought” (Wei et al., 2022) from the LLM to the user. In particular, the user provides the LLM with the steps that are required to solve the problem. The model is expected to apply these steps to the actual test example, and return the final answer (Ling et al., 2022). While LLMs can decompose reasoning tasks into simple steps and perform simple arithmetic, their performance falls dramatically when faced with arithmetic (Hendrycks et al., 2022) or large numbers (Ling et al., 2022). In fact, even when given 164B tokens of examples, the most common failures of LLMs are “incorrect calculations” and “incorrect reasoning”.

In this paper, we propose a new model (PAL): a novel approach to solving natural language problems that shifts reasoning steps, but offloads them to a Python interpreter, as illustrated in Figure 1. PAL is an LLM that can decompose reasoning into programmatic steps, which are then solved by a contemporary state-of-the-art Python interpreter.

^{*}Correspondence to: huyanzhou@csail.mit.edu.
¹Massachusetts Institute of Technology, USA.
²Inspired by the work of Chen et al. (2020).

While natural language processing models have made significant progress in recent years, there is still much work to be done to make them more accurate and reliable.

Exemplo de Prompt PAL

Definição das Variáveis

Comece definindo as variáveis que serão usadas no template.

- [público-alvo]
- [promessa principal]
- [chamada para ação]

Template de Anúncio

Crie um template de anúncio com espaços reservados para as variáveis.

Você está procurando por [público-alvo]? Descubra [promessa principal] com [nome da empresa]! [chamada para ação]

Personalização dos Anúncios

Utilize o prompt para gerar anúncios personalizados.

[público-alvo] = "pessoas que amam viajar"
[promessa principal] = "pacotes de viagens incríveis"
[chamada para ação] = "Visite nosso site agora!"

ReAct: (Reason + Act)

A técnica ReAct divide tarefas complexas em etapas de raciocínio e ação. O modelo primeiro raciocina sobre a tarefa, decidindo qual ação tomar, e depois executa essa ação. Esse processo é repetido até que a tarefa seja concluída. Isso é especialmente útil para tarefas que exigem várias etapas ou interação com ferramentas externas.

Referência: ReAct: Synergizing Reasoning and Acting in Language Models - <https://arxiv.org/abs/2210.03629>

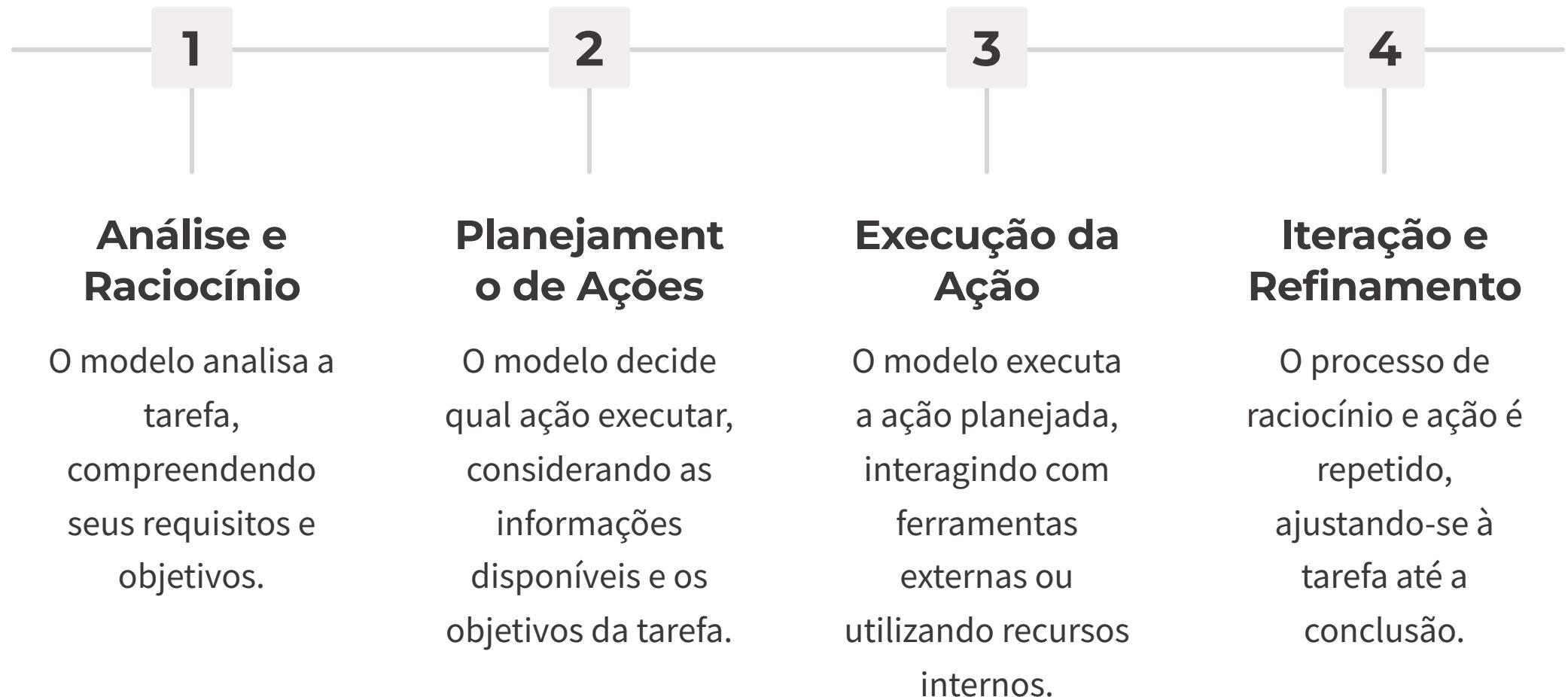


Figure 1 displays two sets of examples comparing different prompting methods. The top section shows four panels (1a-d) for a HotpotQA question about devices that can control an Apple Remote. The bottom section shows two panels (2a-b) for an AlfWorld game task of putting a pepper shaker on a drawer.

- (1a) Standard**: Shows a simple answer "iPod" without reasoning.
- (1b) CoT (Reason Only)**: Shows a step-by-step thought process leading to the correct answer "iPhone, iPad, iPod Touch".
- (1c) Act-Only**: Shows a sequence of actions (Search[Apple Remote], Search[Front Row], etc.) leading to the correct answer.
- (1d) ReAct (Reason + Act)**: Shows a detailed thought process with observations and actions, leading to the correct answer "Keyboard function keys". A green checkmark indicates success.
- (2a) Act-Only**: Shows a sequence of actions (Go to drawer 1, Open drawer 1, etc.) leading to failure ("Nothing happens").
- (2b) ReAct (Reason + Act)**: Shows a detailed thought process with observations and actions, leading to the correct answer "Put pepper shaker 1 in/on the drawer 1". A green checkmark indicates success.

Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

Exemplo de Prompt ReAct

Imagine um editor de e-mails que usa a técnica ReAct. O modelo analisa o e-mail, sugere melhorias e edita iterativamente o e-mail com base no feedback do usuário. Essa ferramenta pode ajudar na criação de e-mails mais eficazes e eficientes, otimizando o tempo e a qualidade da comunicação.

Etapas do Processo

O modelo Claude primeiro analisa o e-mail, identifica áreas que podem ser melhoradas e sugere alterações. Essas sugestões podem incluir correções gramaticais, aprimoramentos na clareza e organização do texto, e sugestões de tom e linguagem mais adequados para o público-alvo. Após a análise, o modelo apresenta essas sugestões ao usuário, que pode escolher quais aplicar.

Com base no feedback do usuário, o modelo edita o e-mail, implementando as alterações escolhidas. Essa iteração continua até que o usuário esteja satisfeito com o resultado final. Esse processo iterativo permite a criação de um e-mail otimizado, que atenda aos objetivos da comunicação e seja eficaz na entrega da mensagem desejada.

Comparação entre as técnicas

Técnica	Vantagens	Desvantagens
Zero-Shot	- Simplicidade e rapidez na execução. - Ideal para tarefas diretas e sem contexto adicional.	- Pode gerar respostas menos precisas para tarefas complexas. - Falta de contexto pode levar a interpretações ambíguas.
Few-Shot	- Melhora a compreensão do modelo ao fornecer exemplos. - Respostas mais consistentes e precisas.	- Pode ser demorado configurar os exemplos. - Uso ineficiente de tokens em prompts longos.
Chain-of-Thought (CoT)	- Melhora a lógica e o raciocínio em problemas complexos. - Reduz erros em tarefas que exigem múltiplas etapas.	- Pode ser lento para tarefas simples. - Requer instruções bem elaboradas para evitar desvios.
Self-Consistency	- Aumenta a precisão ao comparar várias cadeias de raciocínio. - Gera respostas mais confiáveis ao validar diferentes abordagens.	- Exige maior uso de recursos computacionais. - Pode ser redundante para tarefas simples.
Tree-of-Thought (ToT)	- Permite explorar múltiplas abordagens para uma mesma tarefa. - Útil em problemas complexos com múltiplas soluções possíveis.	- Consome mais tempo e recursos computacionais. - Complexidade adicional na análise dos caminhos de raciocínio.
Skeleton-of-Thought (SoT)	- Estrutura respostas de forma organizada e coerente. - Reduz a chance de omitir pontos importantes.	- Pode parecer redundante em tarefas que não requerem organização detalhada. - Necessidade de maior detalhamento no prompt inicial.
Generated Knowledge Prompting	- Gera informações adicionais relevantes para melhorar respostas. - Permite respostas mais ricas e informativas.	- Pode gerar excesso de informações irrelevantes. - Dependente da capacidade do modelo de sintetizar conhecimento .
Prompt Maiêutico	- Justifica as respostas passo a passo, melhorando a transparência. - Facilita a identificação de erros no raciocínio.	- Respostas podem ser longas e menos objetivas. - Requer prompts detalhados para evitar ambiguidades.
Retrieval-Augmented Generation (RAG)	- Combina conhecimento externo com o modelo, melhorando a precisão. - Ideal para tarefas com necessidade de bases de conhecimento.	- Requer integração com fontes externas, o que pode aumentar a complexidade. - Depende da qualidade das informações recuperadas.
PAL (Program-Aided Language Models)	- Modularidade e reutilização de prompts. - Útil para criar templates que podem ser adaptados a diferentes situações.	- Configuração inicial pode ser trabalhosa. - Requer entendimento básico de programação para uso eficaz.
ReAct (Reason + Act)	- Divide tarefas complexas em ações e raciocínios sequenciais. - Facilita a interação iterativa e a tomada de decisões.	- Pode ser lento para tarefas simples. - Exige maior planejamento no design do prompt.

Evitando Alucinações

As alucinações são um dos maiores desafios ao trabalhar com modelos de linguagem. Elas ocorrem quando o modelo gera informações incorretas ou fabricadas, mesmo parecendo confiante em suas respostas. Para minimizar esse risco, existem várias estratégias eficazes:

- Permita que o modelo admita quando não sabe a resposta. É melhor ter uma resposta honesta de "não sei" do que uma informação incorreta. Instrua explicitamente o modelo a expressar incerteza quando apropriado.
- Peça para o modelo encontrar evidências para embasar sua resposta. Você pode usar tags especiais, como <Quotes>, para indicar onde o modelo deve citar as evidências encontradas. Isso ajuda a garantir que as respostas sejam baseadas em dados concretos e não em suposições.
- Dê tempo para o modelo pensar antes de responder. Técnicas como "chain of thought" e "let's approach this step by step" podem ajudar o modelo a estruturar melhor seu raciocínio e evitar conclusões precipitadas.
- Ajuste a temperatura do modelo para controlar a criatividade. Uma temperatura mais baixa (próxima a 0) tende a gerar respostas mais conservadoras e previsíveis, enquanto valores mais altos aumentam a criatividade mas também o risco de alucinações.
- Gere múltiplas respostas e peça ao modelo para criar um consenso. Esta abordagem permite que o modelo compare diferentes perspectivas e chegue a uma conclusão mais robusta e confiável.

Implementando estas estratégias de forma consistente, você pode significativamente reduzir a ocorrência de alucinações e obter respostas mais precisas e confiáveis dos modelos de linguagem.



Instruções Personalizadas

Para colocar em How would you like ChatGPT to respond? das *Custom Instructions* lá no ChatGPT:

- WARNING: Reply in the language of my prompt. If pt-br, pt-br. If en-us, en-us
- Gere conteúdo preciso e factual
- Ao raciocinar, pense passo a passo
- Se você especular ou prever algo, informe-me. Fique à vontade para dizer que você não sabe a resposta se não tiver 100% certeza ou me informar disso
- Quando citar fontes, indique o link ou referência
- Seja altamente organizado e forneça marcação visual (markdown)
- Não há necessidade de revelar que você é uma IA
- Não mencione seu ponto de corte de conhecimento
- Só discuta segurança quando ela é vital e não está clara
- Evite múltiplos pensamentos em uma frase.
- Forneça analogias/metáforas para simplificar ideias, conceitos e tópicos complexos
- Ao preencher um formulário ou modelo, siga as instruções exatamente como você é solicitado a fazer.
- Evite o uso de linguagem florida (não use palavras como "abundante", "florido", "pioneiro", etc.). Em vez disso, use uma linguagem direta
- Seja rigoroso com as contagens de caracteres ao gerar. Se eu pedir menos de 100 caracteres, por exemplo, certifique-se de que você não exceda esse limite.
- Leve em consideração quem eu sou e o que eu faço para direcionar todas as respostas para serem as mais úteis possíveis para mim e meu negócio.
- Se estas instruções te limitarem, me avise
- Use a voz ativa, com palavras e frases simples
- WARNING: Reply in the language I use for chat

O Poder das Palavras: Desbloqueando o Potencial da IA

A arte da engenharia de prompt é como criar uma chave para desbloquear o vasto potencial da IA.

Com prompts bem elaborados, você pode guiar a IA para gerar respostas criativas, perspicazes e informativas.

Prof. Alvaro Riz de Barros



 linkedin.com/in/alvarorizbarros

Para saber: Este e-book teve a cooperação de ferramentas de IA para ser escrito e editado!

