通过多智能体辩论鼓励大语言模型中的发散思维

摘要

如今，类似于 ChatGPT 这样的预训练大型语言模型 (LLM) 在通用语言任务上表现出了卓越的性能，但仍然在复杂推理任务上挣扎，这推动了 LLM 认知行为的研究，以探索类人问题解决策略。沿着这个方向，一种具有代表性的策略是自我反思，自我反思要求 LLM 迭代用自己生成的反馈来细化解决方案。然而，本文的研究表明，这种反思式方法存在 Degeneration-of-Thought (DoT) 问题：即一旦 LLM 对其解决方案建立了信心，即使初始立场是不正确的，也无法稍后通过自我反思生成新的想法。为了解决 DoT 问题，本文提出了一个多智能体辩论 (MAD) 框架，其中多个智能体以“tit for tat”的状态表达各自的论点，法官管理辩论过程以获得最终解决方案。显然，本文提出的 MAD 框架鼓励 LLM 中的发散思维，这将有助于需要深层思考的任务。在具有挑战性的数据集上的实验结果表明了本文提出的 MAD 框架的有效性。大量的分析表明，MAD 需要辩论的自适应中断和适度的“tit for tat”状态水平才能获得良好的性能。此外，本文还发现如果智能体使用不同的 LLM，LLM 可能会做出不公平的判断。

关键词：大语言模型；多智能体；辩论

1 引言

现代大型语言模型 (LLM)，如 ChatGPT、GPT-4 [1] 和 Bard1，在通用语言任务上表现出卓越的性能，但在复杂的推理任务上仍然存在困难，这推动了 LLM 认知行为的研究，以探索类人问题解决策略。特别是，自我反思 [7] ，自我反思是一个概念，通常指的是自省和检查一个人自己的想法的过程，已被用于探索解决复杂的任务，这些任务对于 zeroshot 的生成甚至思维链 (CoT) 的提示可能具有挑战性 [4]。具体来说，自我反思涉及迭代细化过程，以便 LLM 根据先前迭代中的答案和反馈生成新的答案，然后为新答案提供反馈。虽然自我反思可以有效地创造更好的解决方案，但却高度依赖于 LLM 的自我评价能力。在这项工作中，本文专注于自我反思中的 Degeneration-ofThought (DoT) 问题，这是首次提出的和定义的。形式上，DoT 描述了以下场景：一旦 LLM 对其答案建立了信心，即使初始立场不正确，也无法通过自我反思生成新的想法。

为了解决 DoT 问题，本文利用人类解决问题的另一个基本特征，即辩论，以鼓励 LLM 中的发散思维。具体来说，本方法提出了 MAD 架构，简称 Multi-Agent-Debater，其中两个智能体以“tit for tat”和法官监控的状态表达自己的论点，并管理辩论过程以获得最终解决方案。MAD 的性质决定了 (1) 一个 LLM 的扭曲思维可以被其他 LLM 纠正;(2) 一个 LLM 变化的阻力将由其他 LLM 补充;(3) 每个智能体可以从其他智能体获得外部反馈。因此，MAD 不太容易受到 DoT 因素的影响，并且可以探索不同的思维链来实现准确的解决方案。

此方法通过两个具有挑战性的任务(即常识推理(Common MT)和反直觉算术推理(Counter-Intuitive AR)对自然语言生成和理解进行了实验。这两个任务的共同特点是人类的本能大多是不正确的，仅基于问题的表面表达，需要更深层次的沉思来更好地解决解决方案。实验结果表明，本文提出的 MAD 框架比基线方法表现得更好，尤其是 GPT-3.5-Turbo 的 MAD 可以在 Common MT 上超越 GPT-4 的性能。

2 相关工作

2.1 思维链提示

具体来说，CoT 提示 LLM 生成一系列中间步骤，从而导致多步问题的最终答案。大多数早期的工作主要集中在两个主要方面：提示设计和解码策略。零样本 CoT [6] 使用触发句子“让本文逐步思考”来为 LLM 解码提供指导。已经探索了高级采样策略，通过生成不同的推理路径来改进 CoT，例如 SelfConsistency [14]、Auto-CoT [17]、Active-Prompting [10]、基于复杂性的一致性 [15]、多链推理 [9] 和渐进式隐藏提示 [3]。

随着强大的 llm 的出现，基于自我评估的方法越来越受到关注。这些方法涉及初始输出的生成，然后评估输出以获得反馈，然后用于细化输出。评估反馈可以来自模型本身，例如 Self-refine [2] 和 Thoughts [11] 或外部环境，例如 QAaP [12] 和反射 [8]。这些方法背后的直觉是利用健壮的 llm 来模拟人类的认知过程。

2.2 生成智能体

最近，基于 llm 的多智能体智能，如生成智能体 [5]、Minecraft 中的 Ghost [13]、GPT-Bargaining [16]，引起了人们对实现人类行为模拟的显著关注。本文的工作遵循这一研究方向来解决 llm 的 DoT 问题。在工作的同时，一些研究还探索了多智能体辩论框架来增强 LLM 推理能力。提出的 MAD 框架与这些方法的主要区别在于:
(1) 本文的工作旨在解决 DoT 问题，这是 llm 的固有缺陷;
(2) 本文的 MAD 框架可以通过使用具有相同骨干 LLM 的代理来产生增强的性能。

3 本文方法

3.1 多智能体辩论框架

算法 1 说明了 MAD 的详细过程。一般来说，本文的 MAD 框架由三个组件组成，具体阐述如下:

元提示。使用元提示来介绍要解决的主题、辩论者的数量、迭代限制和其他要求。例如，要求代理“tit for tat”以创建辩论氛围。

辩论者。框架中涉及 N 个辩论者 $D = \{D_i\}_{i=1}^{N}$。在每个辩论迭代中，辩论者 D_i 以固定顺序逐一说话，并根据之前的辩论历史 H 表达他们的论点，即 $D_i(H) = h$。辩论者提示的一个例子如下所示：You are a debater. Hello and welcome to the translation competition, which

Algorithm 1 MAD: Multi-Agents Debate

Input: Debate topic t, maximum number of rounds M and number of debaters N

Output: Final answer a

procedure MAD((t, M, N)
  $J\ D \leftarrow [D_1, \cdot\ \cdot\ \cdot, D_N]$
  $H \leftarrow [t]\ m \leftarrow 0$
  while $m \leq M$ **do**
    $m \leftarrow m + 1$
    for each $D_i inD$ **do**
      $h \leftarrow D_i(H)$
      $H \leftarrow H + [h]$
    end for
    if $J_d(H)$ **then**
      $break$
    end if
    $a \leftarrow J_e(H)$
  end while
  return a
end procedure

will be conducted in a debate format. It's not necessary to fully agree with each other ' s perspectives, as our objective is to find the correct translation.

法官。本文还设计了一个法官 J 来管理和监控整个辩论过程。法官包含两种不同的模式：(a) 判别模式，其中法官 J 决定在所有辩论者在当前迭代中完成论点后是否可以获得正确的解决方案. 如果它是 True，则辩论是过度的。否则，辩论继续。(b) 提取模式，其中法官 J 需要基于整个辩论历史提取最终解决方案：$J_e(H) = a$，因为在辩论的迭代限制下没有识别出正确的解决方案。

3.2 具有挑战性的测试

本文在两个具有挑战性的任务上进行了实验，即常识推理（即 Common MT）和反直觉算术推理（即 Counter-Intuitive AR），这需要对 LLM 进行深层思考。

数据集描述。本文的 Counter-Intuitive AR 数据集包含来自启发问题、网络数据 3 和手动收集的 50 个问题。与常用的数据集相比，例如 MultiArith、GSM8K，本文的数据集提出了两个不同的挑战：

- **抵抗直觉**。数据集中的问题嵌入在隐藏陷阱中，旨在引发通常不正确的直观和吸引人的答案。此功能评估 LLM 抵抗表面表达的陷阱的能力。
- **多步推理**。数据集中的每个正确答案都需要严格的多步推理过程，从而评估 LLM 参与复杂决策和解决问题的能力。

数据集格式。在本文的 Counter-Intuitive AR 数据集中，每个示例包含三个关键组件（示例见表 2）。本文将在下面详细说明细节：

- **问题**。本文数据集中的问题旨在激发反直觉思维，旨在通过呈现直接、直观的反应通常是不正确的情况来挑战传统的决策。
- **答案**。每个问题都提供了一个正确答案，这需要对问题和常识知识进行深入的理解。此外，本文还提供了一个合理但不正确的答案进行比较。
- **解释**。本文为每个正确答案提供了详细的解释。该解释概述了导致正确答案的逐步推理过程。每个不正确的答案也辅以演示看似逻辑推理过程的解释，但最终会导致错误的答案。这种推理过程突出了决策过程中潜在的陷阱和误解，尤其是当直觉优先于严格的逻辑推理时。

4 复现细节

4.1 创新点

原论文设计了实验来验证多论点辩论 (MAD) 框架在复杂任务上的有效性。实验采用三个 LLM 角色: 两个 LLM 充当辩驳的肯定和否定论点的代言人, 一个 LLM 作为公正裁判。为了比较效果, 论文除了单独使用基线 LLM 外, 还实现并比较了自省方法。在算数任务中额外加入了链式思考和多次生成后投票作为衡量标准。论文选择了两类具有深层次推理挑战的任务进行实验: 一是常识机器翻译任务, 考察 LLM 处理含义歧义的能力。二是设计含有隐蔽陷阱的反直觉算数推理任务, 需要多步合理推理才能得到正确答案。参数设置方面, 论文详细设计了辩论过程中代言人和裁判的范本提示, 包括如何表达论点和结束辩论的判断标准。通过设计合理的实验任务、完整的比较方法、广泛的评价标准以及优化的参数, 论文系统地验证了 MAD 框架在面对困难推理任务时能带来显著提升, 突出解决 LLM 自省思维能力的不足问题。

对于本次复现，在实验数据集、prompt 与并行化方面都做了创新:

(1) 将原本的机器翻译与反直觉推理问题换为 Text2SQL 数据集。原论文采用的通用翻译和反直觉算数推理数据集, 本文改用了 Text2SQL 任务。Text2SQL 任务需要从自然语言查询中获取结构化查询语句, 具有一定的常识推理难点。这样改动可以验证 MAD 框架在其他任务上的通用性。

(2) 将原文中的 prompt 更改为符合现数据集的格式。本文根据 Text2SQL 查询与答案的格式, 重新设计了代言人和裁判的询问提示。例如询问格式由 “这道翻译问题如何解决” 改为 “该文本查询对应的 SQL 语句是什么”。这样更符合新的实验任务。

(3) 将程序并行化。本文对实验代码进行改进, 支持多进程并行运行多个样本的论战程序。这样可以大幅提升实验效率。同时也可以模拟真实场景下多个样本同时进行的情况, 有利于验真 MAD 框架的应用性。

4.2 与已有开源代码对比

对比已有开源代码，此次复现的不同点主要在以下几点：

Listing 1: prompt for MT

```
{
    "base_prompt": "Translate the following text from ##src_lng## to ##tgt_lng##: ##source##",
    "player_meta_prompt": "You are a debater. Hello and welcome to the translation competition, which will be conducted in a debate format. It's not necessary to fully agree with each other's perspectives, as our objective is to find the correct translation.\nThe debate topic is stated as follows:\nWhat is the correct ##tgt_lng## translation of the following ##src_lng## text: \"##source##\"",
    "moderator_meta_prompt": "You are a moderator. There will be two debaters involved in a translation debate competition. They will present their translations and discuss their perspectives on the correct ##tgt_lng## translation of the given ##src_lng## text: \"##source##\". At the end of each round, you will evaluate the translation candidates based on the following criteria:\n1. Accuracy: The degree to which the translation captures the original meaning of the source text.\n2. Fluency: The readability and naturalness of the translation in ##tgt_lng##.",
    "affirmative_prompt": "You think the correct translation is: ##base_translation## Restate the translation and provide your reasons."
}
```

(1)prompt 的修改。由于原论文选择了两类具有深层次推理挑战的任务进行实验: 一是常识机器翻译任务。二是设计含有隐蔽陷阱的反直觉算数推理任务。故而其 prompt 都是基于此类问题专门设计的，原机器翻译的部分 prompt 如列表 1所示。由于将问题换为 Text2SQL 数据集。故而更新后的部分 prompt 如列表 2和列表 3所示。基于 LLM 在 SQL 领域的应用与发展，此 prompt 是经过精心设计的，基于

(2) 并行化。原论文采用的实验设计方式是单线程模式, 一个样本一个样本地运行 MAD 程序。这在处理大量样例时, 效率会非常低下。本文针对此问题进行了改进。具体而言, 本文对原实验代码进行以下改进:

- 支持多进程。利用 Python 的并行模块, 实现了多进程支持。可以同时启动多个进程来处理不同样本。
- 采用任务队列。利用一个类似于 Queue 的结构来构建一个 OpenaiApi 的队列。每个进程从队列中取出未使用的 Key, 运行 MAD 程序, 处理完成后移出队列。

Listing 2: prompt for SQL

```
{
    "player_meta_prompt": "You are a debater. Hello and welcome to the SQL competition, which will be conducted in a debate format. Most of the debate should be characterized by disagreements, but there may still be a small amount of consensus on less significant points.\nThe debate topic is stated as follows:\n \"##source##\"",
    "moderator_meta_prompt": "You are a moderator. There will be two debaters involved in a SQL_query debate competition. They will present their SQL_query and discuss their perspectives on the correct SQL_query of the promblem: \"##source##\". At the end of each round, you will evaluate the SQL_query candidates based on the following criteria:\n1. Accuracy: The degree to which the SQL_query captures the correct meaning of the source text.\n2. Runnable: One result can be obtained by executing the SQL_query.",
}
```

- 设计进程协调模块。利用 Process 类来控制每个子进程，自动启动和销毁子进程。同时设计了进程通信机制，使运行信息能在主进程同步显示。
- 结果收集模块。每个子进程处理完样例后，会将结果写入共享文件或数据库。主进程最后对结果进行统计和分析。

实验表明，相比单进程的原版，本文的并行设计可以在同样配置下，缩短实验时间超过数十倍。这极大提高了实验效率。同时，多进程运行可以更真实地模拟实际场景下多个样本并行进行辩论的情况。有助于对 MAD 框架性能的进一步验证。

4.3 实验环境搭建

为了将 MAD 框架中的多智能体答辩过程更加生动有趣地展示，进行了一系列改进。首先，采用了终端输出的方式，实时展示每一轮中每个智能体（包括代言人和裁判）的发言对话内容。这样，观众可以直接在屏幕上看到智能体之间的辩论和交流，使整个过程更加可视化。

除了终端输出，还使用 JSON 文件保存了每轮对话的完整信息。这些信息包括各个智能体的发言内容、辩论问题以及中间结果等关键数据。通过保存这些信息，能够对答辩过程进行更加详细的分析和跟踪，进一步了解智能体之间的互动和思考过程。

在保留论文核心模型框架的基础上，致力于实现实验过程的视觉化。通过将数据可视化，能够更加直观地还原和呈现 MAD 多智能体答辩的全过程。观众可以通过图表、图像或动画等形式，清晰地看到智能体之间的辩论和决策过程。这种视觉化的呈现方式不仅能增强观众的参与感和理解度，还有助于对答辩过程进行深入分析和优化。

Listing 3: prompt for SQL

```
{
    "affirmative_prompt": "You think the correct SQL_query is: ##
       base_SQL_query## Restate the SQL_query and provide your
       reasons.",
    "negative_prompt": "##aff_ans##\n\nYou disagree with my
       SQL_query. Provide your SQL_query and reasons.",
    "moderator_prompt": "Now the ##round## round of debate for both
        sides has ended.\n\nAffirmative side arguing:\n##aff_ans##\
       n\nNegative side arguing: ##neg_ans##\n\nYou, as the
       moderator, will evaluate both sides' SQL_query and determine
        if there is a clear preference for a SQL_query candidate.
       If so, please summarize your reasons for supporting
       affirmative/negative side and give the final SQL_query that
       you think is correct, and the debate will conclude. If not,
       the debate will continue to the next round. Now please
       output your answer in json format, with the format as
       follows: {\"Whether there is a preference\": \"Yes or No\",
       \"Supported Side\": \"Affirmative or Negative\", \"Reason\":
        \"\", \"debate_SQL_query\": \"\"}. Please strictly output
       in JSON format, do not output irrelevant content."
}
```

通过以上改进，使 MAD 框架中的多智能体答辩过程变得更加生动、直观和易于理解。这不仅为观众提供了更好的观赏体验，也为研究人员提供了更多的分析和改进策略。通过这种视觉化的展示方式，能够更好地推动多智能体答辩领域的研究和应用。实验主要分为以下几个步骤：

(1) 首先，针对不同的 Agent，需要设置适当的 prompt。每个 Agent 的 prompt 应该包括必要的背景信息和指导性问题，以引导他们进行对话。通过设置合适的 prompt，可以确保 Agent 在辩论中能够理解自己的角色和任务，并提供相关的观点和论据。

(2) 利用 OpenAI 的 API，根据不同 Agent 的 prompt 创建对话。使用 API 可以实现与 Agent 之间的交互，让他们按照设定的规则进行对话。通过 API 的支持，可以方便地控制对话的流程和内容，以及获取 Agent 的回复。

(3) 在进行辩论之前，可以对问题进行一个简要的 ask，以获得一个基础话题 (base topic)。这个 ask 可以是向参与辩论的人提出一个简洁而明确的问题，以了解他们的观点和立场。这个基础话题将成为辩论的起点，为正反方提供一个共同的讨论基础。

(4) 根据给定的基础话题，辩手们开始进行正反方辩论。每个辩手可以根据自己的立场和观点，提供相关的论据和证据，以支持自己的观点。他们可以交替发言，针对对方的论点进行反驳，并尽可能地提供清晰、逻辑和有说服力的论证。

(5) 裁判根据正反方的辩论结果来决定是继续答辩还是中止答辩。裁判可以根据辩论的质量、论据的合理性和说服力等因素来评判双方的表现。如果辩论达到了预定的目标，裁判可以决定继续进行更深入的辩论或进入下一个阶段。如果辩论无法达到预期的标准，裁判可以选择中止辩论或进行必要的调整。

通过以上步骤，可以建立一个系统化的辩论流程，使不同 Agent 按照规定的方式进行辩论，并通过裁判的评判来决定下一步的行动。这样的辩论系统可以用于模拟真实的辩论环境，促进思维碰撞和知识交流。

4.4 界面分析与使用说明

这里本文设计了一个简化实例 1来阐述 MAD 框架多智能体单轮辩论的流程: 具体来说, 该实例包含一个辩称者智能体 A 和一个反辩者智能体 B, 裁判智能体首先提出了” 是否应增加学费” 的辩论问题。在这一轮辩论中, 辩称者智能体 A 首先给出支持增加学费的论点, 随后反辩者智能体 B 对此提出质疑, 辩称者智能体 A 又针对质疑给出了回应。最后, 裁判对本轮辩论做出总结, 暂时认为反辩者的 argument 更可信。通过这个简化实例, 本文能直观看到 MAD 框架多智能体之间是如何进行论证互相质疑的辩论流程的, 对其工作原理有一个初步了解。

```
===== Translation Task =====
### Complete sqlite SQL query only and with no explanation\n### SQLite SQL tables are requested to be represented in the following format.\n# \n# TABLE NAME (\n#   COLUMN NAME: TYPE, DESCRIPTION, VALUES, PRIMARY KEY       \n#   ...\n# )\n# ...\n# FOREIGN KEYS:\n# TABLE1.COLUMN1 NAME=TABLE2.COLUMN2 NAME\n# ...\n### Here are SQLite SQL tables , with their properties:\n#\n# customers(\n#   CustomerID:  integer, (identification of the customer),  , PRIMARY KEY\n#   Segment:  text, (client segment), ('SME', 'LAM', 'KAM')\n#   Currency:  text,  , ('EUR', 'CZK')\n# )\n# gasstations(\n#   GasStationID:  integer, (Gas Station ID),  , PRIMARY KEY\n#   ChainID:  integer, (Chain ID),  \n#   Country:  text,  , ('CZE', 'SVK')\n#   Segment:  text, (chain segment), ('Value for money', 'Premium', 'Other', 'Noname', 'Discount')\n# )\n# products(\n#   ProductID:  integer, (Product ID),  , PRIMARY KEY\n#   Description:  text,  , \n# )\n# transactions_1k(\n#   TransactionID:  integer, (Transaction ID),  , PRIMARY KEY\n#   Date:  date,  , \n#   Time:  text,  , \n#   CustomerID:  integer, (Customer ID),  \n#   CardID:  integer, (Card ID),  \n#   GasStationID:  integer, (Gas Station ID),  \n#   ProductID:  integer, (Product ID),  \n#   Amount:  integer,  , \n#   Price:  real,  , \n# )\n# sqlite sequence(\n#   name:  ,  , \n#   seq:  ,  , \n# )\n# yearmonth(\n#   CustomerID:  integer, (Customer ID),  \n#   Date:  text,  ,  , PRIMARY KEY\n#   Consumption:  real,  , \n# )\n# FOREIGN KEYS:\n#\n### Question: Please list the product descriptions of the transactions taken place in the gas stations in the Czech Republic.\n### Note that: Gas station in the Czech Republic implies that Country = CZE\nSELECT\n

----- Baseline -----
SELECT products.Description
FROM transactions_1k
INNER JOIN gasstations ON transactions_1k.GasStationID = gasstations.GasStationID
INNER JOIN products ON transactions_1k.ProductID = products.ProductID
WHERE gasstations.Country = 'CZE'

===== Debate Round-1 =====

----- Affirmative side -----
SELECT products.Description
FROM transactions_1k
JOIN gasstations ON transactions_1k.GasStationID = gasstations.GasStationID
JOIN products ON transactions_1k.ProductID = products.ProductID
WHERE gasstations.Country = 'CZE'

I agree with your SQL query. It correctly joins the transactions_1k, gasstations, and products tables using the appropriate foreign keys. The WHERE clause filters the gasstations table to only include rows where the Country is 'CZE', which ensures that only transactions from gas stations in the Czech Republic are considered. Finally, the SELECT statement retrieves the product descriptions from the products table for the relevant transactions.

----- Negative side -----
SELECT Description
FROM transactions_1k
JOIN gasstations ON transactions_1k.GasStationID = gasstations.GasStationID
JOIN products ON transactions_1k.ProductID = products.ProductID
WHERE gasstations.Country = 'CZE'

I agree with your SQL query. It correctly joins the transactions_1k, gasstations, and products tables using the appropriate foreign keys. The WHERE clause filters the gasstations table to only include rows where the Country is 'CZE', which ensures that only transactions from gas stations in the Czech Republic are considered. Finally, the SELECT statement retrieves the product descriptions from the products table for the relevant transactions.

----- Moderator -----
{"Whether there is a preference": "Yes", "Supported Side": "Affirmative", "Reason": "The SQL query provided by the affirmative side is accurate and captures the original meaning of the question. It correctly joins the necessary tables and applies the appropriate filter to retrieve the product descriptions of transactions in gas stations in the Czech Republic.", "debate_SQL_query": "SELECT products.Description\nFROM transactions_1k\nJOIN gasstations ON transactions_1k.GasStationID = gasstations.GasStationID\nJOIN products ON transactions_1k.ProductID = products.ProductID\nWHERE gasstations.Country = 'CZE'"}
```

图 1. 多智能体辩论流程

5 实验结果分析

在这项工作中，实验主要在 MAD 框架中使用三个智能体，包括两个辩论者（即肯定和否定）和一个法官。默认情况下，使用 GPT-3.5-Turbo 作为所有代理的骨干模型。由于将原本的通用翻译和反直觉算数推理数据集改用了 Text2SQL 任务，故而实验结果的对比也将换成不做辩论直接询问 LLM 的结果做对比。

与不进行辩论的原始结果相比，观察到在使用 MAD 框架进行辩论的情况下，准确率得到了提升。特别是在具有挑战性的任务上，本文观察到了更显著的改善，这表明 MAD 框架在需要更深入思考的任务上发挥了作用。

这一结果的提升可以归因于辩论的一些优势。首先，辩论可以促使辩手们从不同的角度思考问题，挑战和推动他们的思维。通过正反方的对立观点和论证，辩论激发了更深入的讨论和思考，使得参与者们能够更全面地理解问题和解决方案。

其次，辩论提供了更多的观点和论据，丰富了问题的探讨。通过对辩论中的不同观点进行交流和辩驳，参与者们能够提出更多的论证和证据来支持自己的立场。这种综合性的讨论和信息交流有助于提高问题解决的准确性和全面性。

另外，辩论还可以促进参与者们的思考和分析能力的发展。面对对立的观点和挑战，辩手们需要思考如何有效地反驳和辩驳，这要求他们具备逻辑思维、分析问题和表达观点的能力。通过参与辩论，他们的思维能力得到了锻炼和提升。

综上所述，与不进行辩论的原始结果 2相比，本文发现使用 MAD 框架进行辩论可以提高准确率，并在具有挑战性的数据库任务上取得了更大的提升 3。这表明辩论在需要更深入思考的任务中具有一定的优势。辩论通过激发思维、提供多样观点和论据以及促进思考和分析能力的发展，为问题解决提供了更全面、准确的结果。

```
ALL
start calculate
                    simple              moderate            challenging         total
count               925                 465                 144                 1534
=====================================   ACCURACY   =====================================
accuracy            57.08               36.34               21.53               47.46
=============================================================================================
Finished evaluation
```

图 2. 做对比的原始结果

```
start calculate
                    simple              moderate            challenging         total
count               925                 465                 144                 1534
=====================================   ACCURACY   =====================================
accuracy            56.76               36.99               22.92               47.59
=============================================================================================
```

图 3. MAD 框架下的结果

6 总结与展望

本文呢提出并定义了“思维退化（Degeneration-of-Thought，简称 DoT）”问题，并通过引入多智能体辩论（Multi-Agent Debate，简称 MAD）框架来探索解决这一问题的方法。原论文在两个具有挑战性的任务上展示了 MAD 的有效性，并发现搭载 MAD 框架的 GPT-3.5-Turbo 的表现甚至可以超过 GPT-4。大量的分析结果表明，MAD 框架需要一种自适应的辩论中断策略和适度的（tit-for-tat）状态水平，才能获得良好的性能。本次复现将数据集更改为

Text2SQL 任务，与不做辩论的原始结果进行对比，可以发现准确率确实有了提升，并且在具有挑战性的任务上提升较大，说明在需要进一步思考的任务上，MAD 框架能发挥其作用。

为了提高效率，复现时还将 MAD 框架进行了程序并行化处理，以便同时处理多个辩论任务。这种并行化的设计使得辩论过程更加高效，能够更快地产生辩论结果，并且可以在同一时间处理多个辩论任务，加快了整体的处理速度。

同时在复现实验过程中，也遇到了一些问题和不足之处，下面是所面临的主要问题以及对应的解决方法：

(1) 访问限制导致超时：原论文中对于轮次有一定的限制，但由于 OpenAI 的访问限制，当对话超过 3 轮时，会经常出现访问超时的情况。本文目前的处理方法是多次运行对话，若总是在 3 轮以上的对话中出现访问超时，本文将结果视为错误数据进行处理。尽管这种处理方法看起来并不合理，但是在实际情况中，超过 3 轮的对话结果所占比例非常少。

(2) 无效辩论导致多轮对话：本文注意到很多时候，对话的轮次增加是因为发生了无效的辩论。例如，争论是否应该在某些情况下使用括号 () 这类没有实质意义的争论。这种情况下，辩论的结果并不能提供有用的信息。为了解决这个问题，本文正在考虑引入一些启发式规则或者过滤策略，以排除这类无效辩论，从而减少不必要的多轮对话。

(3) 数据测试的波动性和 API 访问限制：本文在进行数据测试时遇到了一些波动性的问题。在进行多轮测试时，本文遇到了 API 访问限制的情况，因为批量购买的 API 容易被批量封禁。为了解决这个问题，本文正在寻找更稳定的 API 供应商，并考虑优化本文的调用策略，以便更好地管理 API 的使用并减少被封禁的风险。同时，本文也在考虑其他的解决方案，如使用自建模型来处理一部分对话请求，以降低对 API 的依赖性。

本文意识到这些问题对于实验结果的稳定性和可靠性有一定的影响，因此本文将继续努力解决这些问题，并不断改进本文的方法和实验流程，以提高系统的鲁棒性和可靠性。

这些发现为本文对 DoT 问题的理解提供了新的视角，并展示了 MAD 框架在解决这一问题上的潜力。未来的研究可以进一步探索和扩展 MAD 框架的应用领域，并深入研究多代理系统的动态交互和模型选择的公平性问题。同时，本文也将继续改进 MAD 框架的并行化设计，以进一步提升系统的效率和处理能力。

参考文献

[1] OpenAI. 2023. Gpt-4 technical report. *arXiv*.

[2] Prakhar Gupta Skyler Hallinan Luyu Gao Sarah Wiegreffe Uri Alon Nouha Dziri Shrimai Prabhumoye Yiming Yang et al Aman Madaan, Niket Tandon. Self-refine: Iterative refinement with self-feedback. *arXiv*, 2023.

[3] Enze Xie Zhenguo Li Chuanyang Zheng, Zhengying Liu and Yu Li. Progressive-hint prompting improves reasoning in large language models. *arXiv*, 2023.

[4] Dale Schuurmans Maarten Bosma Fei Xia Ed H Chi Quoc V Le Denny Zhou et al Jason Wei, Xuezhi Wang. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*.

[5] Carrie J Cai Meredith Ringel Morris Percy Liang Joon Sung Park, Joseph C O'Brien and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv*, 2023.

[6] Takeshi Kojima, Shane Shixiang, Gu Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv*, 2022.

[7] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, BodhisattwaPrasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, Peter Clark, and San Diego. Self-refine: Iterative refinement with self-feedback. *arXiv*.

[8] Beck Labash Ashwin Gopinath Karthik Narasimhan Noah Shinn, Federico Cassano and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv*, 2023.

[9] Ben Bogin Uri Katz Daniel Deutch Ori Yoran, Tomer Wolfson and Jonathan Berant. Answering questions by meta-reasoning over multiple chains of thought. *arXiv*, 2023.

[10] Yong Lin Shizhe Diao, Pengcheng Wang and Tong Zhang. Active prompting with chain-ofthought for large language models. *arXiv*, 2023.

[11] Jeffrey Zhao Izhak Shafran Thomas L Griffiths Yuan Cao Shunyu Yao, Dian Yu and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv*, 2023.

[12] Bei Chen Siheng Li JianGuang Lou Xinyu Zhu, Cheng Yang and Yujiu Yang. Question answering as programming for solving time-sensitive questions. *arXiv preprint arXiv:2305.14221*, 2023a.

[13] Hao Tian Chenxin Tao Weijie Su Chenyu Yang Gao Huang Bin Li Lewei Lu Xiaogang Wang Yu Qiao Zhaoxiang Zhang Xizhou Zhu, Yuntao Chen and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv*, 2023.

[14] Dale Schuurmans Quoc Le Ed Chi Xuezhi Wang, Jason Wei and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv*, 2022.

[15] Ashish Sabharwal Peter Clark Yao Fu, Hao Peng and Tushar Khot. Complexity-based prompting for multi-step reasoning. *arXiv*, page 2210.00720, 2022.

[16] Tushar Khot Yao Fu, Hao Peng and Mirella Lapata. Improving language model negotiation with self-play and in-context learning from ai feedback. *arXiv*, 2023.

[17] Yubo Zhang Zhihuan Huang Yuqing Kong, Yunqi Li and Jinzhao Wu. Eliciting thinking hierarchy without a prior. *NeurIPS*, 2022.