Fact-Checking Complex Claims with Program-Guided Reasoning

Liangming Pan[1,2] Xiaobao Wu[3] Xinyuan Lu[4] Anh Tuan Luu[3]
William Yang Wang[1] Min-Yen Kan[4] Preslav Nakov[2]

[1] University of California, Santa Barbara [2] Mohamed bin Zayed University of Artificial Intelligence
[3] Nanyang Technological University [4] National University of Singapore

liangmingpan@ucsb.edu xiaobao002@e.ntu.edu.sg luxinyuan@u.nus.edu
anhtuan.luu@ntu.edu.sg william@cs.ucsb.edu
kanmy@comp.nus.edu.sg preslav.nakov@mbzuai.ac.ae

Abstract

事实核查往往需要收集多个证据并应用复杂的多步推理。本文引入了一种新颖的事实核查模型“Program-Guided Fact-Checking”(PFC)，该模型将复杂的主张分解成更简单的子任务，并使用一组专门函数的共享库来解决这些子任务。我们首先利用大型语言模型的上下文学习能力生成“推理程序”来指导核查过程。之后，我们通过将每个子任务委托给相应的子任务处理器来“执行”这个程序。这个过程使我们的模型具有解释性和数据效率，提供清晰的推理过程解释并且需要很少的训练数据。我们在两个具有挑战性的事实核查数据集上评估了PFC，并显示其在不同证据可用性设置下优于七个事实核查基线，同时输出明确的程序以便人类调试。[1]（代码和数据公开在https://github.com/mbzuai-nlp/ProgramFC）

1 Introduction

虚假信息（例如，在社交媒体上）的泛滥使得自动事实核查成为自然语言处理（NLP）的一个重要应用。给定一个主张，目标是找到证据，然后基于这些证据对主张的真实性作出裁决 (???)。

评估真实世界主张的真实性通常涉及收集多个证据并进行复杂的推理 (????)。例如，考虑主张“詹姆斯·卡梅隆和电影《星际穿越》的导演都出生在加拿大”。在网上找到直接证据来证实或反驳这个主张可能是具有挑战性的。相反，人工事实核查员需要分解主张，收集多个证据，并进行逐步的推理(?)，如图 1 所示。这使得验证复杂的主张比之前的研究中探索的典型设置更具挑战性，典型设置中一篇文章的信息足以支持/反驳主张 (??????)。

除了多步推理外，我们还必须考虑两个关键方面以开发可靠的事实核查系统：(*i*) 可解释性：模型不仅应该预测主张的真实性，还应该在推理过程中提供清晰的解释，以帮助用户理解和信任结果。(*ii*) 数据效率：人工标注通常耗时、昂贵并具有潜在偏见，这使得难以收集足够高质量的标记数据用于模型训练，尤其是对于复杂的主张。因此，构建一个能够在最小或无训练数据的情况下良好执行的模型是可取的。尽管已经提出了一些模型 (???)来促进事实核查中的多步推理，但它们要么缺乏推理过程的可解释性，要么需要大量的任务特定训练示例。

在本文中，我们提出了程序引导的事实核查 (PROGRAMFC)，这是一种既有解释性又具备数据效率的新型事实核查框架。图 1 展示了我们的方法。为了验证复杂的主张，PROGRAMFC将它们分解为更简单的子任务，可以使用专门的子任务函数库来解决。具体而言，PROGRAMFC首先为输入主张生成一个*推理程序*，它是一个以动作[参数]形式的子任务序列（例如图 1 中的 S1-S4），其中动作和参数分别定义了子任务的类型和内容。生成的推理程序作为逐步验证主张的指南。然后，我

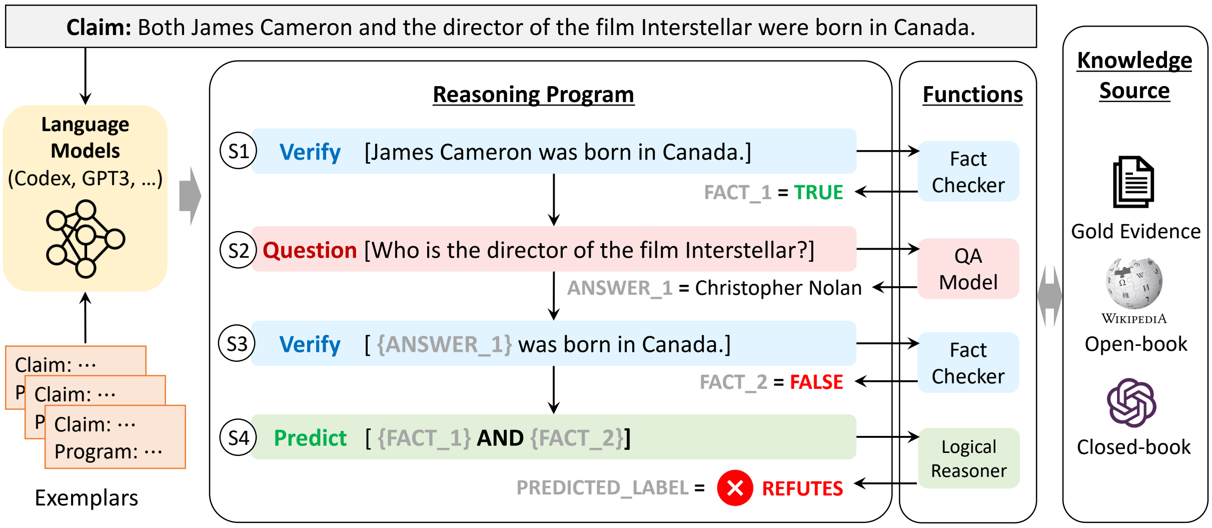

图 1: 我们*model*模型的概述，它包括两个模块：(*i*) 程序生成模块使用Codex带有上下文学习的方式为输入的命题生成一个推理程序，然后 (*ii*) 程序执行模块通过将每个步骤委托给对应的子任务函数来依次解释程序。

们通过将每个子任务顺序委派给相应的子任务处理程序来执行程序，如图 1 中的函数列所示。这些子任务可能包括回答问题、验证简单主张或进行逻辑推理。PROGRAMFC结合了可解释性和数据效率。它使用推理程序提供其推理过程的清晰解释。对于数据效率，仅凭少量示例作为提示，大型语言模型(LLMs)可以解决各种任务，例如在上下文学习中 (?)。我们利用LLMs的这种能力，通过向模型展示少量的(claim, program)对作为示范，生成给定主张的推理程序。PROGRAMFC也非常灵活，它允许在不影响系统其余部分的情况下轻松替换子任务函数的实现，以适应不同的事实核查设置。我们可以允许这些函数从外部源获取信息（在开放式问题设置中），或者要求它们仅根据LLM的内部参数化知识生成答案（在封闭式问题设置中）。

我们在两个专为复杂断言事实核查设计的具有挑战性的数据集上对PROGRAMFC进行评估：HOVER (?) 和 FEVEROUS (?)，并且我们展示它在两个数据集上比七个少样本事实核查基线模型表现更优 (§ 4.1)。当需要的推理深度增加时，程序引导推理策略变得越来越有效 (§ 4.1)。在开放领域设置中，我们发现推理程序可以增强来自知识源的相关证据检索 (§ 4.2)。即使我们使用弱模型作为子任务求解器，PROGRAMFC也能保持其鲁棒性 (§ 4.2)。我们还通过人工评估和错误分析来评估推理程序的可解释性 (§ 4.3)。

2 Related Work

事实核查。 近年来，自动事实核查在自然语言处理（NLP）研究界引起了相当大的关注，作为对抗错误信息和虚假信息的手段。已经提出了各种数据集，以便于开发和评估自动事实核查系统，其中最流行的基于人工构建的维基百科内容的声明 (???)和政治或科学领域中自然发生的声明 (???????)。值得注意的是，这些数据集中大多数是通过一种方式构建的，即用于支持或驳斥某个声明的证据可以在单个文件中找到。例如，在FEVER (?)中，超过87%的声明只需要来自单个维基百科文章的信息 (?)。

为了弥补这一差距，已经提出了用于研究需要多步推理的复杂声明的数据集 (??)。图形模型 (??????)用于促进对多个证据进行推理。尽管这些模型取得了相当大的性能提升，但它们缺乏可解释性并且过于依赖大量的训练数

据。为了解决上述问题，我们提出了一种可解释性强、灵活且数据效率高的模型，该模型生成推理图作为解释，并利用上下文学习实现少样本学习。

解释生成。 面对现实世界声明的复杂性，仅仅给出一个最终的真实性可能无法令人信服 (**?**)。以前的研究提出了各种方法来为模型预测提供事后解释，例如使用注意力权重突出显示证据的相关部分 (**????**)，基于知识图谱的逻辑系统生成证明 (**??**)，以及生成检索到的相关证据的摘要 (**???**)。相比之下，我们提出使用推理程序提供解释，这些解释由类似程序的自然语言描述的子任务组成。这具有几个优点：它允许解释不局限于证据，如注意力权重；它比基于逻辑的解释更灵活；它比自由形式的总结更简洁。

思维链推理。 此外，与生成事后解释的先前工作不同，我们还使用推理程序作为预测声明真实性的指导。这是受到最近链式思考提示（CoT） (**???**)的成功启发，该方法生成逐步自然语言推理步骤，以引导模型回答复杂问题。我们采用这个想法来检查复杂的声明。与原始CoT不同，原始CoT使用单个LLM进行分解和问答，而我们只使用语言模型生成推理程序作为解决问题的蓝图，并将每个子任务委托给专门的函数。这种方法减轻了语言模型的负担，使其更加灵活，可以集成必要的组件用于事实检查，例如证据检索器。程序引导推理的策略也符合最近工具增强语言模型的趋势 (**??**)，即为语言模型提供外部工具和资源的增强。

3 PROGRAMFC

首先，我们对事实检查问题进行了形式化描述，然后介绍了我们提出的程序引导的事实检查模型（PROGRAMFC）。

3.1 Problem Formulation

给定一个主张 C，一个事实核查模型 $\mathcal{F}$ 的目标是基于一个知识源 $\mathcal{K}$ 预测一个标签 Y 来评估该主张的真实性为TRUE （真）或者FALSE （假）。该模型还需要输出一个解释 E 来证明预测的真实性标签。我们总结了根据知识源 $\mathcal{K}$ 的类型，事实核查分为三种不同的设置。

- **黄金证据**：对于每一个主张，$\mathcal{K}$ 是支持或反驳该主张的黄金证据文档的集合。这种设置也被称为主张验证 (**??**)。
- **开放书设置**：$\mathcal{K}$ 是一个大规模的文本语料库，例如维基百科。模型首先从语料库中检索相关的证据，然后根据这些证据预测真实性标签 (**??**)。
- **封闭书设置**：模型无法访问任何外部知识源（$\mathcal{K} = \emptyset$）。它需要利用其参数中存储的知识（在预训练和微调期间获取）来验证主张。这个设置在应用大型语言模型进行事实检查的工作中被探索过 (**??**)。

3.2 Program-Guided Reasoning

我们的目标是对一个需要多步推理的复杂命题C进行事实核查。我们专注于小样本设定，即模型只能利用少量的领域内的示例进行训练。为了解决这个问题，PROGRAMFC采用程序生成和执行的范式，如图 1所示。

程序生成。 在这个阶段，给定输入命题C，规划器$\mathcal{P}$为其生成一个推理程序$P = [S_1, \cdots, S_n]$，其中包含n个按顺序排列的推理步骤S_i。

每个推理步骤$S_i \in P$是一个用自然语言控制的指令，指导S_i去执行系统提供的辅助子任务函数集合$\mathcal{F}$中的某个函数。具体来说，我们定义$S_i = (f_i, A_i, V_i)$，其中f_i指定子任务函数$f_i \in \mathcal{F}$，A_i是传递给函数f_i的参数，V_i是存储函数调用$f_i(A_i)$的返回结果的变量。对于一个有效的推理程序，最后一个推理步骤的返回值必须是一个布尔值，表示命题C的真实性标

签，即$V_n \in \{\textsc{True}, \textsc{False}\}$。

程序执行。 在执行阶段，推理程序P由一个解释器运行，以推导出命题C的真实性标签。解释器按顺序解析P中的推理步骤。对于每个步骤$S_i = (f_i, A_i, V_i)$，它调用相应的现成的子任务函数f_i并向其传递参数A_i。参数A_i可以是逻辑表达式或自然语言句子，例如问题或简单命题。函数调用的结果然后被存储在变量V_i中。由于后续步骤常常依赖于之前步骤的结果，我们允许参数A_i在前面的步骤中引用变量$V_1, \cdots, V_{i-1}$。例如，在图 1中，S_3中的参数是"*{ANSWER_1} was born in Canada.*"，它引用了S_2的返回变量*{ANSWER_1}*。在执行S_3时，该变量会被替换为其实际值，参数变为"*Christopher Nolan was born in Canada*"。执行完最后一步后，返回值即为命题C的预测真实性标签。

汇总推理路径。 请注意，可能有多个推理路径可以达到最终的真实性标签。因此，我们为输入命题生成了一个多样化的候选推理程序集合$\mathcal{P} = \{P_1, \cdots, P_N\}$。在执行$\mathcal{P}$中的所有程序后，我们根据所有$N$个预测标签的多数票作为最终标签。这种方法类似于人类在事实核查中依赖多种验证方法来增加信心。它还使得模型不容易受到个别推理程序中的错误的影响。

3.3 Reasoning Program Generation

我们的程序生成器基于*Codex* (?)，一个经过代码预训练的LLM，它可以将自然语言解析为符号表示，如SQL (?)或Python程序 (??)。

然而，推理程序的语法与编程语言的语法不同。我们利用Codex的少样本泛化能力，发现它可以从一小部分上下文示例$\mathcal{D} = \{d_1, \cdots, d_{|D|}\}$中有效学习。每个示例$d_i$包括一个主张和一个程序。该程序具有类似于Python的语法，每个推理步骤以$V_i = f_i(A_i)$的格式编写。在推理时，我们用任务的指导说明K个上下文示例和输入主张C来提示Codex。Codex然后尝试完成下列文本，并生成C的程序。提示的模板如图 2所示。我们使用$K = 20$来权衡推理类型的多样性和模型的最大输入容量。我们使用基于采样的解码（温度为0.7）来生成多次运行的不同推理程序。

3.4 Sub-Task Functions

我们为该模型实现了三个子任务函数，在程序执行过程中进行调用。

• **问题:** 该子任务函数是一个问答模块，它以问题 Q 作为输入参数，并返回该问题的答案 A。我们使用 `FLAN-T5` (?)，一种改进的 T5 模型 (?)，该模型使用了超过1.8K个任务进行预训练，并经过指导调整，在许多问答基准测试中取得了最先进的零/少样本性能。如图 3所示，我们根据第 3.1节中定义的设置与模型进行交互处理。对于闭卷设置，输入的提示为：

```
Q: QUESTION ? The answer is:
```

对于另外两种设置，输入提示为：

```
EVIDENCE Q: QUESTION ?
The answer is:
```

• **验证：** 这是一个事实验证模块，它以一个声明C作为输入参数，并返回 True 或 False 的标签。我们也使用 `FLAN-T5` 来进行这个模块的验证，通过以下的问答格式来提示模型。

```
EVIDENCE
Q: Is it true that CLAIM ?
True or False? The answer is:
```

• **预测:** 该模块以逻辑表达式作为输入，逻辑表达式对之前步骤中的变量进行与、或、非运算。其输出作为预测的真实性标签返回。

4 Experiments

数据集。 大多数事实核查数据集主要由可以通过一条证据证实的简单声明组成。然而，在这里，我们关注需要多步推理的复杂声明。鉴于这种背景，我们选择评估我们的模型在唯一两个符合这些标准的数据集上：HOVER (?) 和

```
'''Generate a python-like program that describes the reasoning steps
    required to verify the claim step-by-step. You can call three functions
    in the program: 1. Question() to answer a question; 2. Verify() to
    verify a simple claim; 3. Predict() to predict the veracity label.'''

# The claim is that Both James Cameron and the director of the film
    Interstellar were born in Canada.
def program():
    fact_1 = Verify("James Cameron was born in Canada.")
    Answer_1 = Question("Who is the director of the film Interstellar?")
    fact_2 = Verify("{Answer_1} was born in Canada.")
    label = Predict(fact_1 and fact_2)

(··· more in-context examples here ···)

# The claim is that <input_claim>
def program():
```

图 2: 用于生成推理程序的Codex提示模板，包括一个任务指令，上下文示例和一个<input_claim>的提示。完整的模板见附录 D。

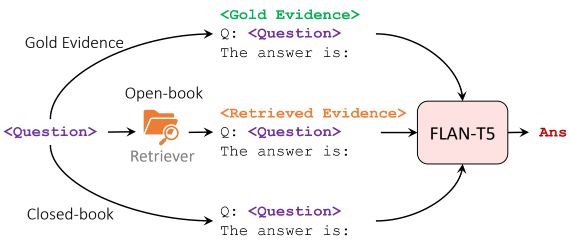

图 3: 对于三种不同设置，实施问题回答子任务函数。

FEVEROUS (?)。由于测试集未公开发布，我们使用验证集进行评估。HOVER包含需要对多个维基百科文章进行整合和推理的声明。我们将其验证集分为三个子集，根据验证声明所需的“跳数”：1,126个两跳声明、1,835个三跳声明和1,039个四跳声明。FEVEROUS主要关注对非结构化和结构化数据的复杂声明进行事实核查，其中每个声明都以句子和/或维基百科表格中的单元格形式的证据进行注释。由于我们专注于文本事实核查，我们仅选择了只需要句子证据的声明，共计2,962个声明。我们将这个子集称为FEVEROUS-S。

为了在开放式图书模式下进行评估，我们使用为这两个数据集构建的相应维基百科语料库作为知识来源。HOVER使用由 ?处理的2017年10月维基百科转储，包含有520万个维基百科页面的简介部分。FEVEROUS使用2020年12月维基百科转储，包含有540万个完整的维基百科文章。

基准模型。 我们将PROGRAMFC与七种基准模型进行比较，分为三组。(*i*) 预训练模型：BERT-FC (?) 和 LisT5 (?) 分别利用BERT和T5进行事实核实。(*ii*) *FC/NLI*微调模型：我们选择了在其他事实核查数据集或自然语言推理（NLI）数据集上进行微调的三个预训练模型。RoBERTa-NLI (?) 使用经过微调的RoBERTa-large模型在四个NLI数据集上进行微调；DeBERTaV3-NLI (?) 在FEVER和四个NLI数据集的885,242个（声明、证据、标签）注释上微调DeBERTaV3模型；MULTIVERS (?) 是在FEVER上微调的LongFormer (?)模型。(*iii*) 上下文学习模型：其中一个基线是直接使用我们VERIFY模块中的FLAN-T5模型进行事实核查。另一个基线使用Codex的上下文学习进行少样本事实核查。关于实现细节，请参见附录 A。

少样本学习。 我们研究只有少量领域内样本可用的少样本学习。因此，为了公

Few-shot learning models		HOVER (2-hop)		HOVER (3-hop)		HOVER (4-hop)		FEVEROUS-S	
		Gold	Open	Gold	Open	Gold	Open	Gold	Open
I	BERT-FC (?)	53.40	50.68	50.90	49.86	50.86	48.57	74.71	51.67
	LisT5 (?)	56.15	52.56	53.76	51.89	51.67	50.46	77.88	54.15
II	RoBERTa-NLI (?)	74.62	63.62	62.23	53.99	57.98	52.40	88.28	57.80
	DeBERTaV3-NLI (?)	**77.22**	68.72	65.98	60.76	60.49	56.00	91.98	58.81
	MULTIVERS (?)	68.86	60.17	59.87	52.55	55.67	51.86	86.03	56.61
III	Codex (?)	70.63	65.07	66.46	56.63	63.49	57.27	89.77	62.58
	FLAN-T5 (?)	73.69	69.02	65.66	60.23	58.08	55.42	90.81	63.73
IV	ProgramFC (N=1)	74.10	69.36	66.13	60.63	65.69	**59.16**	91.77	67.80
	ProgramFC (N=5)	75.65	**70.30**	**68.48**	**63.43**	**66.75**	57.74	**92.69**	**68.06**

表 1: PROGRAMFC（IV）和基线模型（I-III）在HOVER和FEVEROUS-S的评估集上的宏F1分数。*Gold*和*Open*分别表示黄金证据设置和开放式图书设置。 I：预训练的Transformer模型；II：FC/NLI微调模型；III：上下文学习模型。

平比较，我们限制所有模型只能访问来自HOVER或FEVEROUS-S的20个样本。

我们会将这些样本用于对预训练模型（BERT-FC和LisT5）进行微调，对FC/NLI微调模型进行连续微调，或对FLAN-T5和Codex进行上下文示例。对于PROGRAMFC，我们将它们用作推理程序生成的上下文示例。

我们评估黄金证据设置和开放式图书模式的表现。基线模型在这两种设置下是相同的。然而，进行开放式图书模式的测试时，模型会接收到检索到的证据而不是真实的证据。

我们使用使用Pyserini工具包 (?)实现的BM25 (?)作为PROGRAMFC和基线模型的检索器。我们使用从知识语料库中检索的前10个段落作为证据。

4.1 Main Results

我们在表1中报告了PROGRAMFC和少样本事实检查基线的总体结果。在8次评估中，PROGRAMFC在其中7次中取得了最佳表现，证明了其有效性。我们还有三个更具体的观察结果。

ProgramFC对于较深层次的论证效果更好。

在HOVER数据集上，ProgramFC（N=5）平均比基准线模型表现高出10.38%，11.37%，和14.77%分别在两跳、三跳和四跳论证上。这表明随着需要进行的推理深度增加，ProgramFC变得越来越有效。在基准线模型中，DeBERTaV3-NLI在两跳论证上执行得与ProgramFC相当，这表明对较简单论证的大规模预训练有助于模型泛化到更复杂的论证中。

然而，随着论证的复杂性增加，这种泛化变得更具挑战性。在HOVER数据集上，DeBERTaV3-NLI的F1分数从两跳论证的77.22降至四跳论证的60.49，减少了21.7%。相比之下，使用计划引导推理策略的ProgramFC的性能下降较小，仅为11.7%。

使用与子任务函数相同的FLAN-T5模型的ProgramFC模型，在所有四个数据集上表现优于直接使用FLAN-T5验证论证的基准线。平均而言，黄金证据设置下有6.0%的改进，开放式设置下有4.5%的改进。这表明使用程序将复杂的论证分解为简单的步骤可以促进更准确的推理。当所需推理复杂时，这尤为明显：4跳论证的黄金证据设置下有14.9%的改进，开放

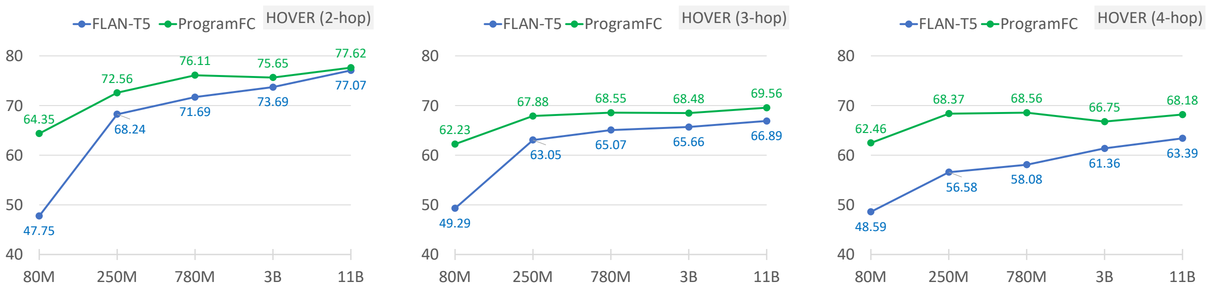

图 4: 使用黄金证据对FLAN-T5 （蓝线）和PROGRAMFC（绿线）进行事实核查的F1分数，其中语言模型的规模逐渐增大：FLAN-T5-small （80M）、FLAN-T5-base （250M）、FLAN-large （780M）、FLAN-T5-XL （3B）和FLAN-T5-XXL （11B），在HOVER 2跳（左）、3跳（中）和4跳（右）上。

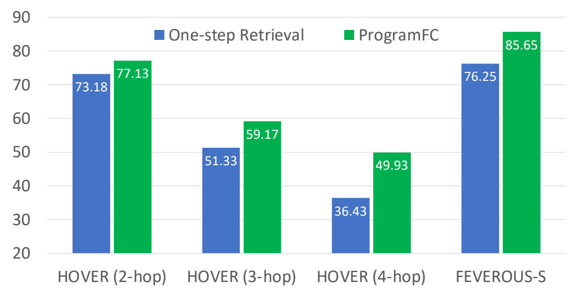

图 5: 在 PROGRAMFC中，一步检索和迭代检索的检索召回率@10。

式设置下有6.7%的改进。

聚合推理程序是有益的。

我们发现，通过将5个推理程序的预测聚合，相较于仅使用单个程序，平均性能提升了1.5%。这与 ?的研究结果一致，他们在问答任务中采用了这个方法：如果有多种不同的思路得出相同的答案，我们可以更有信心地认为最终答案是正确的。这个直觉同样适用于事实核查，因为每个程序代表了一条独特的推理链来验证主张。

4.2 How Does the Reasoning Program Help?

为了进一步了解推理程序如何促进事实核查，我们将PROGRAMFC与FLAN-T5在不同的语言模型大小（small、base、large、XL 和 XXL）上的性能进行了比较。结果如图 4 所示，表明当模型尺寸较小时，程序引导的推理特别有效。由于较小的模型容量较小，端到端的FLAN-T5模型的性能随模型尺寸的减小而显著下降。然而，对于 PROGRAMFC来说，这种趋势不太明显。推理程序提供的高层次推理计划大大减轻了对随后子任务解决器的需求。我们的结果表明，使用FLAN-T5-small（80M参数）作为子任务解决器的程序引导模型可以在4跳断言的端到端推理上实现与尺寸为11B的FLAN-T5-XXL模型相媲美的性能。

在开放域设置中，我们发现推理程序可以增强从知识源中检索相关证据的能力。图 5 比较了基线中使用单步BM25检索器和PROGRAMFC中迭代的逐步BM25检索器的检索性能。

我们衡量了前10个检索段落的召回率（recall@10）。对于 PROGRAMFC，我们组合了所有步骤的检索段落，并考虑前10个结果。从图 5 可以看出，PROGRAMFC在所有数据集上的性能均优于一步检索，其中在 HOVER 4-hop 数据集上改进最大，达到37.1%。这是因为一些信息可能在原始主张中无法找到，只能在推理过程中揭示出来（如图 1 中的“Christopher Nolan”）。因此，由推理程序引导的迭代检索能够获得更好的结果。

4.3 Interpretability of Reasoning Programs

PROGRAMFC的一个优势在于它相比端到端模型提高了事实核查的可解释性，因为明确的程序可以帮助人类理解和调试。附录 B的图7展示了生成的推理程序的示例。为了评估生成的推理程序的质量，我们从HOVER的2跳、3跳和4跳数据集中随机选择了300个错误预测

Error Type	Proportion (%)		
	2-hop	3-hop	4-hop
Syntax error	**0%**	**0%**	**0%**
Semantic error	**29%**	**38%**	**77%**
Token	8%	20%	18%
Structure	19%	13%	57%
Subtask	2%	5%	2%
Incorrect execution	**71%**	**62%**	**23%**

表 2: 对于HOVER中每个跳步长度的错误预测样本进行推理程序评估。

的真实标签的案例，每个数据集选择100个样例。我们请人类注释者分析错误类型，并将结果分类为三个类别：*(i)* 语法错误，程序不符合定义的语法且无法被解析；*(ii)* 语义错误，包括错误或缺失的参数/变量(*Token*)，错误的程序结构(*Structure*)，和错误的子任务调用(*Subtask*)；以及*(iii)* 执行错误，程序本身是正确的，但错误的预测结果是由于其执行引起的。

我们在表格 2中展示了错误分析。首先，在我们的样本中没有发现语法错误，这表明Codex通过少量的上下文学习有效地生成可执行程序。

其次，对于2跳的主张，我们发现71%的程序是正确的，大多数错误是由于程序执行错误导致的，其中问答或事实检验模块未能返回正确答案。

第三，随着主张的复杂性增加，程序中语义错误的比例增加，其中结构错误尤为突出。这突显了为需要长链推理的主张生成适当的逐步推理策略的困难。图 6展示了一个结构错误的例子，模型无法将主张的第二句正确解析为程序指令。附录 C中可以找到其他错误例子。

4.4 Closed-Book Fact-Checking

最后，我们评估了闭卷设置，即模型无法访问任何知识源，需要仅依赖其参数化知识。表 1 中的 I 组和 II 组的基准模型是使用（证据，主张）对训练的，因此在此设置下不适用。我们将我们的方法与使用大型语言模型进行上下文学习的基准方法进行比较，包括表 1 中的 Codex (`code-davinci-002`) 和 FLAN-T5。

Model	HOVER			FEVEROUS
	2-hop	3-hop	4-hop	
`InstructGPT`				
`- Direct`	56.51	51.75	49.68	60.13
`- ZS-CoT`	50.30	52.30	51.58	54.78
`- CoT`	**57.20**	53.66	51.83	**61.05**
`- Self-Ask`	51.54	51.47	52.45	56.82
`Codex`	55.57	53.42	45.59	57.85
`FLAN-T5`	48.27	52.11	51.13	55.16
`ProgramFC`	54.27	**54.18**	**52.88**	59.66

表 3: 闭卷考试环境下，PROGRAMFC和基准模型的宏F1得分。

我们还包括使用了 175B 参数的 InstructGPT (`text-davinci-002`) (?)，并使用了四个不同的提示：（*i*）用主张进行直接提示，（*ii*）使用 CoT (?) 或演示进行思路链提示，（*iii*）使用 ZS-CoT (?) 或零样本思路链和提示 “让我们一步一步思考”，以及（*iv*）使用 Self-Ask (?)，它是 CoT 的一种变体，通过提问一系列问题来引导模型的推理。详细的提示模板在附录 E 中给出。

我们的结果如表 3所示，在HOVER数据集上，大多数模型的宏F1得分仅略高于随机猜测，表明仅仅依赖大型语言模型的参数化知识来核查复杂的声明是困难的。与 4.1节中的观察结果类似，我们注意到随着所需推理步数的增加，性能有所提高的趋势。思路链提示的得分平均比直接提示高2.7分，强调了逐步推理对于复杂事实核查的重要性。它在HOVER 2-hop和FEVEROUS上优于我们的PROGRAMFC，但在HOVER 3-hop和4-hop上表现较差。

这可能是因为思路链生成自由形式的解释，可能导致长推理链中的不可预测错误。相比之下，我们的程序生成和执行策略对于更长的推理链更加稳定。

```
Claim:
Emery, located in the same state as Edison Local School District, is a ghost town. It is near the
city that lies close to the Ohio Turnpike, a 241.26 mi highway.

Predicted Program:
answer_1 = Question("Which state is Emery located in?")
answer_2 = Question("Which state is Edison Local School District located in?")
fact_1 = Verify("{answer_1} and {answer_2} are the same state.")
fact_2 = Verify("Emery is a ghost town.")
answer_3 = Question("Which city is near Emery?")
answer_4 = Question("Which city lies close to the Ohio Turnpike, a 241.26 mi highway?")
fact_3 = Verify("{answer_3} is near {answer_4}.") ⟶ fact_3 = Verify("Emery is near {answer_4}.")
label = Predict(fact_1 and fact_2 and fact_3)
```

图 6: 在HOVER 4-hop数据集中出现了一个错误示例，其中生成的推理程序具有错误的程序结构。错误的部分用红色标记，正确的修订部分用绿色标记。

5 Conclusion and Future Work

我们提出了PROGRAMFC，一种针对事实核实的少样本神经符号模型，该模型学习将输入的声明映射到一个推理程序，该程序由一系列子任务函数调用组成，用于回答问题、核查简单断言以及计算逻辑表达式。然后通过执行该程序进行事实核查。PROGRAMFC结合了符号程序的优点，如可解释性，与端到端神经模型的灵活性。使用Codex作为程序生成器，PROGRAMFC在HOVER和FEVEROUS上只通过少量的上下文演示并且无需额外训练就展现了有希望的性能。我们还调查了模型大小和检索程序的益处，分析了错误。结果表明，PROGRAMFC有效地平衡了模型能力、学习效率和可解释性。

在未来的工作中，我们希望通过先进的推理程序设计和子任务功能，将应用于更多的真实世界事实核实场景，如假新闻检测和多模态事实核查。

Limitations

我们确定了PROGRAMFC的两个主要局限性。首先，尽管HOVER和FEVEROUS数据集中的论断在表面形式上是复杂的，但大多数情况下只需要进行显式的多步推理，即可以从论断的句法结构或论断的框架中得出分解。这降低了生成推理程序的难度。然而，对于许多现实世界的复杂论断，推理通常是隐含的。例如，对于论断“亚里士多德不可能使用笔记本电脑”，推理程序如下所示：

answer_1 = Question(“亚里士多德什么时候活着？”);

answer_2 = Question(“笔记本电脑是什么时候发明的？”);

fact_1 = 验证(“answer_1 在 answer_2 之前。”);

标签 = 预测（事实_1）

生成此类隐含复杂主张的推理程序需要对主张有更深入的理解，同时也需要世界和常识知识的支持。我们对这些类型的主张进行了初步实验，但我们发现我们基于Codex的生成器难以产生一个正确的推理程序。这凸显了将我们的模型应用于事实核查真实世界主张的差距。解决这些挑战是未来工作的一个重要方向。

其次，我们发现相较于基准的端到端事实核查模型，模型的计算成本更高。它需要调用大型语言模型进行程序生成，还需要进一步调用多个子任务模型。这导致实际计算时间比端到端 FLAN-T5 模型高约4-5倍。因此，开发更高效的程序生成和执行方法是未来工作的一个重要方向。

Ethics Statement

偏见。 我们注意到训练LLMs所使用的数据可能存在一些偏见，以及一些事实判断可能也存在偏见。这两者都超出了我们的控制范围。

预期的使用和滥用潜力。 我们的模型可能对大众有兴趣，同时也可以节省人工事实核查员大量的时间。然而，它们也有可能被恶意行

为者滥用。因此，我们要求研究人员要谨慎使用。

环境影响。 我们想要提醒大家，使用大型语言模型需要大量的计算资源，并且在训练过程中使用了GPU/TPU，这会增加全球变暖的程度。在我们这种情况下，这个问题相对较小，因为我们并不是从零开始训练这样的模型；相反，我们进行了少量的上下文学习。然而，我们所调用的大型语言模型（Codex）很可能在GPU上运行。

Acknowledgements

本研究获得了美国国家科学基金会奖励号码2048122的支持。所表达的观点是作者个人的，不代表美国政府的官方政策或立场。感谢美国加州大学圣塔芭芭拉分校自然语言处理小组的Alex Mei、Xinyi Wang、Danqing Wang、Sharon Levy、Gyuwan Kim和其他成员给予的宝贵意见。

References

A Implementation Details about the Baselines

在本节中，我们将介绍我们在研究中使用的七个基线模型的实现细节。使用大型语言模型来进行少样本事实检验的典型方法包括微调和上下文学习。因此，我们将这些基线模型分为三个类别。

A.1 Pre-trained Models

预训练模型使用预训练的Transformer (?)，例如BERT (?)和T5 (?)，进行事实核查。对于少样本学习，我们使用HOVER或FEVEROUS中随机抽样的20个训练示例进行微调。我们使用不同的随机种子进行10次训练，并报告验证集上的平均性能。我们选择了两个模型：

- `BERT-FC`：它使用BERT进行主张验证。主张和证据被连接起来（`[CLS] claim [SEP] evidence`），并作为输入用于二分类任务，以预测主张的真实性标签。我们使用HuggingFace提供的`bert-large-uncased`（345M参数）模型。.[1]

- `LisT5` (?)：这是一个使用预训练的序列到序列转换模型T5 (?)作为其基础架构的事实核查框架。我们采用论文中提出的“列表连接”方法进行标签预测，这将所有候选证据句子连接成一个输入，我们训练`t5-large`模型直接对声明进行分类，判断为支持或反驳。我们使用了该模型的原始实现。[2]

A.2 FC/NLI Fine-Tuned Models

这些模型是在单跳事实检查数据集（例如FEVER）或自然语言推理（NLI）数据集上进行了特定微调的预训练Transformer模型。这种额外训练使得这些模型在事实检查简单声明方面表现出色，因此它们在需要进一步少量样本微调期间的多跳推理时可以得到更好的概括。在这个类别中，我们选择了以下三个微调模型：

- `RoBERTa-NLI` (?) 在四个知名的自然语言推理数据集（SNLI (?)、MNLI (?)、FEVER-NLI (?)、ANLI（R1，R2，R3）(?)）的组合上进行了对 RoBERTa-large (?) 的微调。我们使用了 HuggingFace 提供的公开模型检查点[3]，并通过 HOVER/FEVEROUS 随机选择了20个示例进一步进行了微调。

- `DeBERTaV3-NLI` (?) 是在885,242个来自FEVER数据集的NLI假设–前提配对和四个NLI数据集（MNLI, ANLI, LingNLI (?)和WANLI (?)）上对`DeBERTaV3-large`模型进行微调的。截至2022年06月06日，这是HuggingFace上表现最佳的NLI模型。[4]

- `MULTIVERS` (?)，曾被称为`LongChecker`，使用LongFormer (?)对于长文本证据问题进行主张验证。我们使用在FEVER上进行微调的模型检查点。[1]

A.3 In-Context Learning Models

这些模型最近在各种自然语言处理任务中表现出强大的小样本学习能力。通过用几个上下文的示例来激活一个大型语言模型，该模型能够迅速从演示中学习一个任务。为了与我们的模型进行公平比较，我们选择了两个上下文学习基线模型，如下所示。

- 我们的模型使用`Codex` (?)来生成推理程序。一个直接的基准线使用`Codex`进行事实核查。为此，我们以以下方式提示`Codex`（`code-davinci-002`）：“`<Evidence>`

[1] `https://huggingface.co/`

[2] `https://github.com/castorini/pygaggle/tree/master/experiments/list5`

[3] `https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli`

[4] `https://huggingface.co/MoritzLaurer/DeBERTa-v3-large-mnli-fever-anli-ling-wanli`

根据上述信息,<Claim>是否为真？真或假？答案是:”。在提示之前，我们使用相同的20个上下文示例对我们的模型进行前缀演示。

- FLAN-T5 (?) 是T5的改进版本，它在1.8K个以指令形式表达的任务上进行了微调，包括有和没有示例的情况，即零样本学习和少样本学习。该模型在各种上下文中的少样本学习自然语言处理任务中表现出强大的性能，如推理和问答。我们以与第3.4节中使用的相同格式提示模型：“<Evidence> Q: <Claim> 是否真实地 <Claim>? 是或否? 答案是:”，前缀为相同的20个上下文示例。我们还使用与我们的模型相同的模型大小（FLAN-T5-XXL 3B），以进行公正比较。

B Examples of Generated Reasoning Programs

图 7 显示了由 ProgramFC生成的六个不同类型推理链的推理程序示例。

C Error Analysis for Reasoning Programs

图 8展示了五个错误案例，其中生成的推理程序是不正确的。我们对每个错误案例都提供了解释：

案例1 生成了一个错误的逻辑推理运算符作为最后一步。正确的逻辑应该是“not (fact_1 and fact_2)”，而不是“fact_1 and fact_2”。

案例2 在第三和第四个推理步骤中，没有执行共指消解。应该将“这张专辑”替换为“这张蓝草专辑”，以使得子任务与上下文无关。“这部音乐片”应该替换为第一步中的变量“answer_1”。

案例3 未能为声明创建一个有意义的问题分解。它生成了一个简单地重复原始声明的程序。

案例4 未能为输入声明生成一个细粒度的推理结构。它也生成了一个简单地将声明分隔成句子的程序。

案例5 生成了一个多余的推理步骤“Question("音乐家是何时出生的？")”，这个步骤没有为推理链中添加任何新的信息。

D Program Generation Prompts

我们对HOVER和FEVEROUS-S数据集手动编写的提示在代码清单 1和 2中给出。

E Prompts for Closed-Book Fact-Checking

以下我们展示了在第 4.4 节中用于 InstructGPT 封闭书籍事实核查设置的四种提示方式的模板。

直接提示

```
# Answer the following true/false questions:

Is it true that The woman the story behind Girl Crazy
is credited to is older than Ted Kotcheff?
The answer is: False

(··· more in-context examples here ···)

Is it true that <input_claim>?
The answer is:
```

ZS-CoT Prompting

```
# Answer the following true/false question:

Is it true that <input_claim>? True or False?
Let us think step-by-step. The answer is:
```

CoT Prompting

```
# Answer the following true/false questions:

Is it true that The woman the story behind Girl Crazy
is credited to is older than Ted Kotcheff?
Let's think step by step.
Girl Crazy's story is credited to Hampton Del Ruth.
Hampton Del Ruth was born on September 7, 1879.
Ted Kotcheff was born on April 7, 1931.
Therefore, the answer is: False.

(··· more in-context examples here ···)

Is it true that <input_claim>?
Let's think step by step.
```

Self-Ask Prompting

```
Claim:
Tritonia and Phyteuma are both names for a plant genus.

Predicted Program:
fact_1 = Verify("Tritonia is a name for a plant genus.")
fact_2 = Verify("Phyteuma is a name for a plant genus.")
label = Predict(fact_1 and fact_2)
```

```
Claim:
The country that Fujairah College is located in had a 2013 population of 9.2 million until it was
hit by the plague in 1483 when the population was halved.

Predicted Program:
answer_1 = Question("Which country is Fujairah College located in?")
fact_1 = Verify("{answer_1} had a 2013 population of 9.2 million.")
fact_2 = Verify("{answer_1} was hit by the plague in 1483.")
fact_3 = Verify("The population of {answer_1} was halved in 1483.")
label = Predict(fact_1 and fact_2 and fact_3)
```

```
Claim:
The first female board member for the Liberal Party, she was born in Vestfold county in Norway.

Predicted Program:
answer_1 = Question("Who was the first female board member for the Liberal Party?")
fact_1 = Verify("{answer_1} was born in Vestfold county in Norway.")
label = Predict(fact_1)
```

```
Claim:
The solicitor who won the show  Back to Reality ahead of Maureen Rees and Craig Phillips is
English. The solicitor that was a chair of Global Witness is also English.

Predicted Program:
answer_1 = Question("Which solicitor won the show Back to Reality ahead of Maureen Rees and Craig
Phillips?")
answer_2 = Question("Which solicitor was a chair of Global Witness?")
fact_1 = Verify("{answer_1} is English.")
fact_2 = Verify("{answer_2} is English.")
label = Predict(fact_1 and fact_2)
```

```
Claim:
Anthony Burgess addressed the novelist and essayist, the author of Grimus, in a lengthy love
letter. The author is of the same nationality as Raj Koothrappali.

Predicted Program:
answer_1 = Question("Which novelist and essayist did Anthony Burgess address in a lengthy love
letter?")
fact_1 = Verify("{answer_1} is the author of Grimus.")
answer_2 = Question("What is the nationality of Raj Koothrappali?")
fact_2 = Verify("{answer_1} is of the same nationality as {answer_2}.")
label = Predict(fact_1 and fact_2)
```

```
Claim:
The critically acclaimed film, that Buddy Baker scored in 1975, is a Walt Disney film. It was
produced first before the film that featured Bruce M. Fischer as Mr. Coogar.

Predicted Program:
answer_1 = Question("Which critically acclaimed film did Buddy Baker score in 1975?")
fact_1 = Verify("{answer_1} is a Walt Disney film.")
answer_2 = Question("Which film featured Bruce M. Fischer as Mr. Coogar?")
fact_2 = Verify("{answer_1} was produced first before {answer_2}.")
label = Predict(fact_1 and fact_2)
```

图 7: 由 PROGRAMFC生成的推理程序示例如下:

```
# Answer the following true/false questions:

Is it true that The woman the story behind Girl Crazy
is credited to is older than Ted Kotcheff?
Q: The story behind Girl Crazy is credited to whom?
A: Hampton Del Ruth
Q: Is Hampton Del Ruth older than Ted Kotcheff?
A: No
So the final answer is: False.

(··· more in-context examples here ···)

Is it true that <input_claim>?
```

Semantic Error — Token: incorrect or missing arguments/variables

Example 1:
Bitter Jester and The Future of Food are not both documentaries about food.

Predicted Program:
```
fact_1 = Verify("Bitter Jester is a documentary about food.")
fact_2 = Verify("The Future of Food is a documentary about food.")
label = Predict(fact_1 and fact_2)→ label = Predict(not (fact_1 and fact_2))
```

Example 2:
The record producer that produced the bluegrass album was born on 22 June, 1944. This album inspired a Tony award winning musical. This musical had a character that was originated by Carmen Cusack.

Predicted Program:
```
answer_1 = Question("Which record producer produced the bluegrass album?")
fact_1 = Verify("{answer_1} was born on 22 June, 1944.")
fact_2 = Verify("This album inspired a Tony award winning musical.")
fact_3 = Verify("This musical had a character that was originated by Carmen Cusack.")
label = Predict(fact_1 and fact_2 and fact_3)

→

answer_1 = Question("Which record producer produced the bluegrass album?")
fact_1 = Verify("{answer_1} was born on 22 June, 1944.")
fact_2 = Verify("The bluegrass album inspired a Tony award winning musical.")
fact_3 = Verify("{answer_1} had a character that was originated by Carmen Cusack.")
label = Predict(fact_1 and fact_2 and fact_3)
```

Semantic Error — Structure: incorrect program structure

Example 3:
Richard Strauss, German composer of Die Nacht, composed another well-established opera called Der Rosenkavalier.

Predicted Program:
```
fact_1 = Verify("Richard Strauss, German composer of Die Nacht, composed another well-established
                 opera called Der Rosenkavalier.")
label = Predict(fact_1)

→

fact_1 = Verify("Richard Strauss is a German composer of Die Nacht.”)
fact_2 = Verify("Richard Strauss composed a well-established opera called Der Rosenkavalier.")
label = Predict(fact_1 and fact_2)
```

Example 4:
The film Deanna Oliver produced in 1999 grossed $36.8 million domestically. The musical film based on coach Herman Boone, did not.

Predicted Program:
```
fact_1 = Verify("The film Deanna Oliver produced in 1999 grossed $36.8 million domestically.")
fact_2 = Verify("The musical film based on coach Herman Boone, did not.")
label = Predict(fact_1 and fact_2)

→

answer_1 = Question("What film did Deanna Oliver produce in 1999?")
fact_1 = Verify("{answer_1} grossed $36.8 million domestically.")
answer_2 = Question("What musical film was based on coach Herman Boone?")
fact_2 = Verify("{answer_2} grossed less than $36.8 million domestically.")
label = Predict(fact_1 and fact_2)
```

Semantic Error — Subtask: missing / redundant / incorrect sub-task calls

Example 5:
The musician, who founded Morningwood with Max Green, is older than Max Green.

Predicted Program:
```
answer_1 = Question("Who founded Morningwood with Max Green?")
answer_2 = Question("When was Max Green born?")
```
~~answer_3 = Question("When was the musician born?")~~
```
fact_1 = Verify("{answer_3} is older than {answer_2}.")→ {answer_1} is older than {answer_2}.
label = Verify(fact_1)
```

图 8: 生成的推理程序出现错误的示例情况。错误的部分以红色标出，正确的修订以绿色标出。

```python
'''Generate a python-like program that describes the reasoning steps required to
    verify the claim step-by-step. You can call three functions in the program: 1.
    Question () to answer a question; 2. Verify () to verify a simple claim; 3.
    Predict() to predict the veracity label.'''

# The claim is that Howard University Hospital and Providence Hospital are both
    located in Washington, D.C.
def program():
    fact_1 = Verify("Howard University Hospital is located in Washington, D.C.")
    fact_2 = Verify("Providence Hospital is located in Washington, D.C.")
    label = Predict(fact_1 and fact_2)

# The claim is that WWE Super Tuesday took place at an arena that currently goes by
    the name TD Garden.
def program():
    answer_1 = Question("Which arena the WWE Super Tuesday took place?")
    fact_1 = Verify(f"{answer_1} currently goes by the name TD Garden.")
    label = Predict(fact_1)

# The claim is that Talking Heads, an American rock band that was "one of the most
    critically acclaimed bands of the 80's" is featured in KSPN's AAA format.
def program():
    fact_1 = Verify("Talking Heads is an American rock band that was 'one of the
    most critically acclaimed bands of the 80's'.")
    fact_2 = Verify("Talking Heads is featured in KSPN's AAA format.")
    label = Predict(fact_1 and fact_2)

# The claim is that An IndyCar race driver drove a Formula 1 car designed by Peter
    McCool during the 2007 Formula One season.
def program():
    answer_1 = Question("Which Formula 1 car was designed by Peter McCool during the
     2007 Formula One season?")
    fact_1 = Verify(f"An IndyCar race driver drove the car {answer_1}.")
    label = Predict(fact_1)

# The claim is that Gina Bramhill was born in a village. The 2011 population of the
    area that includes this village was 167,446.
def program():
    answer_1 = Question("Which village was Gina Bramhill born in?")
    fact_1 = Verify(f"The 2011 population of the area that includes {answer_1} was
    167,446.")
    label = Predict(fact_1)

# The claim is that Don Ashley Turlington graduated from Saint Joseph's College, a
    private Catholic liberal arts college in Standish.
def program():
    fact_1 = Verify("Saint Joseph's College is a private Catholic liberal arts
    college is located in Standish.")
    fact_2 = Verify(f"Don Ashley Turlington graduated from Saint Joseph's College.")
    label = Predict(fact_1 and fact_2)

# The claim is that Gael and Fitness are not published in the same country.
def program():
```

```
    answer_1 = Question("Which country was Gael published in?")
    answer_2 = Question("Which country was Fitness published in?")
    fact_1 = Verify(f"{answer_1} and {answer_2} are not the same country.")
    label = Predict(fact_1)

# The claim is that Blackstar is the name of the album released by David Bowie that
    was recorded in secret.
def program():
    fact_1 = Verify("David Bowie released an album called Blackstar.")
    fact_2 = Verify("David Bowie recorded an album in secret.")
    label = Predict(fact_1 and fact_2)

# The claim is that In the 2004 Hockey film produced by a former major league
    baseball pitcher Kurt Russell played the USA coach.
def program():
    answer_1 = Question("Which 2004 Hockey film was produced a former major league
    baseball pitcher?")
    fact_1 = Verify("Kurt Russell played the USA coach in the film {answer_1}.")
    label = Predict(fact_1)

# The claim is that Along with the New York Islanders and the New York Rangers, the
    New Jersey Devils NFL franchise is popular in the New York metropolitan area.
def program():
    fact_1 = Verify("The New York Islanders and the New York Rangers are popular in
    the New York metropolitan area.")
    fact_2 = Verify("The New Jersey Devils NFL franchise is popular in the New York
    metropolitan area.")
    label = Predict(fact_1 and fact_2)

# The claim is that Jack McFarland is the best known role of the host of the 64th
    Annual Tony Awards.
def program():
    answer_1 = Question("Who is the host of the 64th Annual Tony Awards?")
    fact_1 = Verify(f\"Jack McFarland is the best known role of {answer_1}.")
    label = Predict(fact_1)

# The claim is that The song recorded by Fergie that was produced by Polow da Don
    and was followed by Life Goes On was M.I.L.F.$.
def program():
    fact_1 = Verify("M.I.L.F.$ was recorded by Fergie that was produced by Polow da
    Don.")
    fact_2 = Verify("M.I.L.F.$ was was followed by Life Goes On.")
    label = Predict(fact_1 and fact_2)

# The claim is that Eatza Pizza and Your Pie were not founded in the same state.
def program():
    answer_1 = Question("Which state was Eatza Pizza founded in?")
    answer_2 = Question("Which state was Your Pie founded in?")
    fact_1 = Verify(f"{answer_1} and {answer_2} are not the same state.")
    label = Predict(fact_1)

# The claim is that Gregg Rolie and Rob Tyner, are not a keyboardist.
def program():
    fact_1 = Verify("Gregg Rolie is not a keyboardist.")
```

```
    fact_2 = Verify("Rob Tyner is not a keyboardist.")
    label = Predict(fact_1 and fact_2)

# The claim is that Maria Esther Andion Bueno, not Jimmy Connors, is the player that
     is from Brazil.
def program():
    fact_1 = Verify("Maria Esther Andion Bueno is from Brazil.")
    fact_2 = Verify("Jimmy Connors is not from Brazil.")
    label = Predict(fact_1 and fact_2)

# The claim is that Vladimir Igorevich Arnold died after Georg Cantor.
def program():
    answer_1 = Question("When did Vladimir Igorevich Arnold die?")
    answer_2 = Question("When did Georg Cantor die?")
    fact_1 = Verify(f"{answer_1} is after {answer_2}.")
    label = Predict(fact_1)

# The claim is that Barton Mine was halted by a natural disaster not Camlaren Mine.
def program():
    fact_1 = Verify("Barton Mine was halted by a natural disaster.")
    fact_2 = Verify("Camlaren Mine was not halted by a natural disaster.")
    label = Predict(fact_1 and fact_2)

# The claim is that John O'Hara and Rabindranath Tagore are not the same nationality
    .
def program():
    answer_1 = Question("What is the nationality of John O'Hara?")
    answer_2 = Question("What is the nationality of Rabindranath Tagore?")
    fact_1 = Verify(f"{answer_1} and {answer_2} are not the same nationality.")
    label = Predict(fact_1)


# The claim is that Thomas Loren Friedman has won more Pulitzer Prizes than Colson
    Whitehead.
def program():
    answer_1 = Question("How many Pulitzer Prizes has Thomas Loren Friedman won?")
    answer_2 = Question("How many Pulitzer Prizes has Colson Whitehead won?")
    fact_1 = Verify(f"{answer_1} is more than {answer_2}.")
    label = Predict(fact_1)

# The claim is that The model of car Trevor Bayne drives was introduced for model
    year 2006. The Rookie of The Year in the 1997 CART season drives it in the
    NASCAR Sprint Cup.
def program():
    answer_1 = Question("Which model of car is drived by Trevor Bayne?")
    fact_1 = Verify(f"{answer_1} was introduced for model year 2006.")
    answer_2 = Question("Who is the Rookie of The Year in the 1997 CART season?")
    fact_2 = Verify(f"{answer_2} drives the model of car Trevor Bayne drives in the
    NASCAR Sprint Cup.")
    label = predict(fact_1 and fact_2)

# The claim is that <input_claim>
def program():
```

Listing 1: The prompt used for Program Generation for HOVER.

```
'''Generate a python-like program that describes the reasoning steps required to
    verify the claim step-by-step. You can call three functions in the program: 1.
    Question () to answer a question; 2. Verify () to verify a simple claim; 3.
    Predict() to predict the veracity label.'''

# The claim is that In 1959, former Chilean boxer Alfredo Cornejo Cuevas (born June
    6, 1933) won the gold medal in the welterweight division at the Pan American
    Games (held in Chicago, United States, from August 27 to September 7) in Chicago
    , United States, and the world amateur welterweight title in Mexico City.
def program():
    fact_1 = Verify("Alfredo Cornejo Cuevas was born in June 6, 1933.")
    fact_2 = Verify("Alfredo Cornejo Cuevas won the gold medal in the welterweight
    division at the Pan American Games in 1959.")
    fact_3 = Verify("The Pan American Games in 1959 was held in Chicago, United
    States, from August 27 to September 7.")
    fact_4 = Verify("Alfredo Cornejo Cuevas won the world amateur welterweight title
     in Mexico City.")
    label = Predict(fact_1 and fact_2 and fact_3 and fact_4)

# The claim is that The Footwork FA12, which was intended to start the season,
    finally debuted at the San Marino Grand Prix, a Formula One motor race held at
    Imola on 28 April 1991.
def program():
    fact_1 = Verify("The Footwork FA12, which was intended to start the season.")
    fact_2 = Verify("The Footwork FA12 finally debuted at the San Marino Grand Prix.
    ")
    fact_3 = Verify("The San Marino Grand Prix was a Formula One motor race held at
    Imola on 28 April 1991.")
    label = Predict(fact_1 and fact_2 and fact_3)

# The claim is that SkyHigh Mount Dandenong (formerly Mount Dandenong Observatory)
    is a restaurant located on top of Mount Dandenong, Victoria, Australia.
def program():
    fact_1 = Verify("SkyHigh Mount Dandenong is a restaurant located on top of Mount
     Dandenong, Victoria, Australia.")
    fact_2 = Verify("SkyHigh Mount Dandenong is formerly known as Mount Dandenong
    Observatory.")
    label = Predict(fact_1 and fact_2)

# The claim is that Before the first Europeans arrived or copra companies leased it,
     Maupihaa was home to Inca's in ancient times.
def program():
    fact_1 = Verify("Maupihaa was home to Inca's in ancient times.")
    fact_2 = Verify("Maupihaa was home to Inca's before the first Europeans arrived
    or copra companies leased it.")
    label = Predict(fact_1 and fact_2)

# The claim is that Shulin, a 33.1288 km (12.7911 sq mi) land located in New Taipei
    City, China, a country in East Asia, has a total population of 183,946 in
    December 2018.
def program():
    fact_1 = Verify("Shulin is a 33.1288 km (12.7911 sq mi) land located in New
```

```python
    Taipei City, China.")
    fact_2 = Verify("Shulin has a total population of 183,946 in December 2018.")
    label = Predict(fact_1 and fact_2)

# The claim is that Sumo wrestler Toyozakura Toshiaki committed match-fixing, ending
     his career in 2011 that started in 1989.
def program():
    fact_1 = Verify("Toyozakura Toshiaki ended his career in 2011 that started in
    1989.")
    fact_2 = Verify("Toyozakura Toshiaki is a Sumo wrestler.")
    fact_3 = Verify("Toyozakura Toshiaki committed match-fixing.")
    label = Predict(fact_1 and fact_2 and fact_3)

# The claim is that In 1959, former Chilean boxer Alfredo Cornejo Cuevas (born June
    6, 1933) won the gold medal in the welterweight division at the Pan American
    Games (held in Chicago, United States, from August 27 to September 7) in Chicago
    , United States, and the world amateur welterweight title in Mexico City.
def program():
    fact_1 = Verify("Alfredo Cornejo Cuevas is a former Chilean boxer.")
    fact_2 = Verify("Alfredo Cornejo won the gold medal in the welterweight division
     at the Pan American Games.")
    fact_3 = Verify("The Pan American Games was held in Chicago, United States, from
     August 27 to September 7.")
    fact_4 = Verify("Alfredo Cornejo won the world amateur welterweight title in
    Mexico City.")
    label = Predict(fact_1 and fact_2 and fact_3 and fact_4)

# The claim is that Adductor hiatus is associated with nine structures, seven of
    which enter and leave through hiatus.
def program():
    fact_1 = Verify("Adductor hiatus is associated with nine structures.")
    fact_2 = Verify("Seven of the nine structures associated with Adductor hiatus
    enter and leave through hiatus.")
    label = Predict(fact_1 and fact_2)

# The claim is that Ifor Bowen Lloyd was educated at Winchester (an independent
    boarding school for boys in the British public school tradition) and Exeter
    College, Oxford where he was a member of the Library Committee of the Oxford
    Union Society, as well as, received a BA in Modern History in 1924.
def program():
    fact_1 = Verify("Ifor Bowen Lloyd was educated at Winchester and Exeter College,
     Oxford.")
    fact_2 = Verify("Winchester is an independent boarding school for boys in the
    British public school tradition.")
    fact_3 = Verify("While at Oxford, Ifor Bowen Lloyd was a member of the Library
    Committee of the Oxford Union Society.")
    fact_4 = Verify("Ifor Bowen Lloyd received a BA in Modern History in 1924 at
    Oxford.")
    label = Predict(fact_1 and fact_2 and fact_3 and fact_4)

# The claim is that In the 2001 Stanley Cup playoffs Eastern Conference Semifinals
    Devils' Elias scored and Maple Leafs' left Devils player Scott Neidermayer hurt.
def program():
    fact_1 = Verify("In the 2001 Stanley Cup playoffs Eastern Conference Semifinals
```

```
    Devils' Elias scored.")
    fact_2 = Verify("Maple Leafs' left Devils player Scott Neidermayer hurt.")
    label = Predict(fact_1 and fact_2)

# The claim is that Teldenia helena is a moth first described in 1967 by Wilkinson.
def program():
    fact_1 = Verify("Teldenia helena is a moth.")
    fact_2 = Verify("Teldenia helena was first described by Wilkinson in 1967.")
    label = Predict(fact_1 and fact_2)

# The claim is that Born December 30, 1974, William Frick was a dark horse candidate
     in the Maryland House of Delegates appointment process.
def program():
    fact_1 = Verify("William Frick was born in December 30, 1974.")
    fact_2 = Verify("William Frick was a dark horse candidate in the Maryland House
    of Delegates appointment process.")
    label = Predict(fact_1 and fact_2)

# The claim is that <input_claim>
def program():
```

Listing 2: The prompt used for Program Generation for FEVEROUS-S.