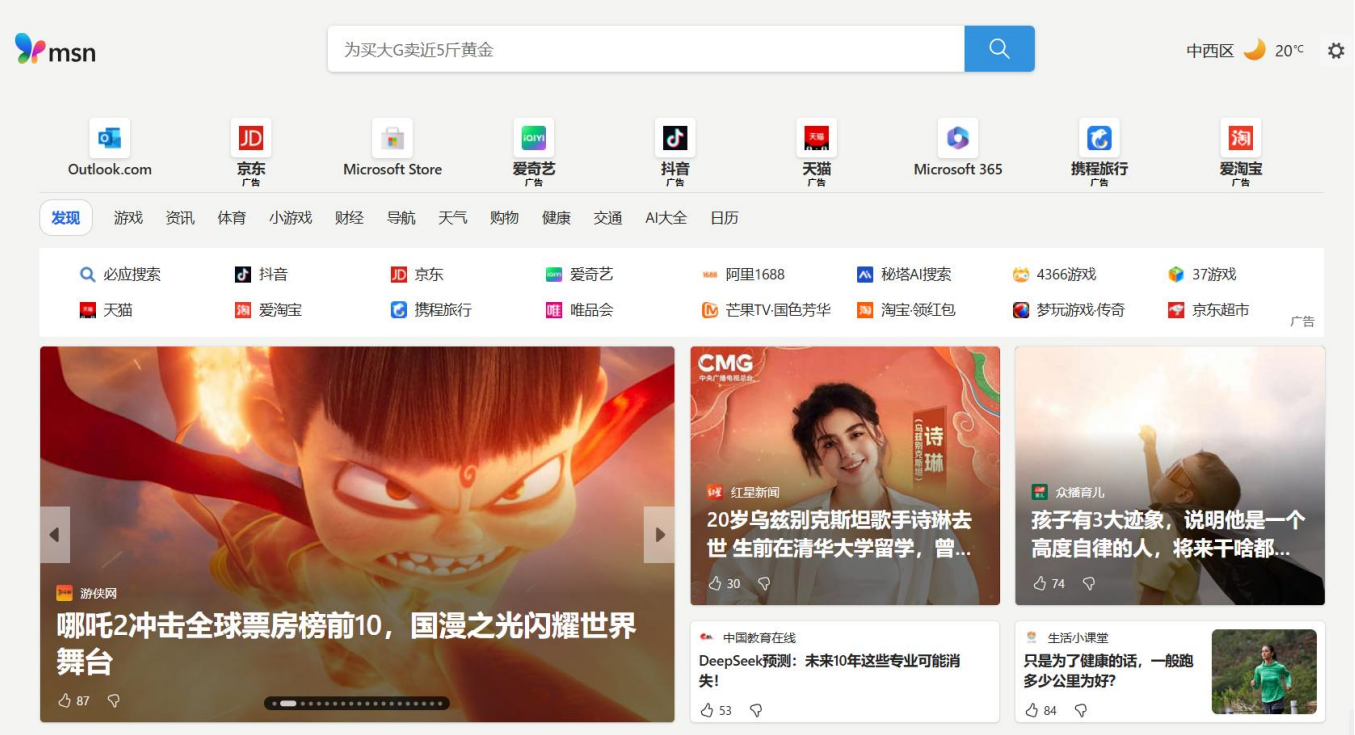
# Exploring the Applications of LLMs in Fake News Detection

Microsoft Internship Summary
Xin-Shuang Zhang

# Background

https://www.msn.com/zh-cn




MSN (Microsoft Network) News

**Fake News Detection:**
Improve News Recommendation Service

Limitations:

- No ground truth in MSN
- No fine-tuning
- Exploratory, single-person team

# Frontier Advances

- A NLP task, ACL and EMNLP, Prompt Engineering
- Lack of high-quality datasets
- Mainly focused on commonsense reasoning
- Problem formulation is not well done
- Lack of interpretability

# What is fake news

**Definition 1 (Fake News [3]).** *Fake news is a news article that is intentionally and verifiably false.*

Specifically, there are three key features of this definition [3, 4]:

1. **Authenticity**. Fake news includes false information that can be verified as such.
2. **Intent**. Fake news is created with dishonest intention to mislead consumers.
3. **Whether the information constitutes news**.

| Concept | Non-factual? | Intent to deceive? | News? |
|---|---|---|---|
| Fake news [2, 3, 4] | ✔ | ✔ | ✔ |
| False news [6] | ✔ | | ✔ |
| Satire news [7] | | ✗ | ✔ |
| Disinformation [8] | ✔ | ✔ | |
| Misinformation [9] | ✔ | | |
| Cherry-picking [10] | ✗ | ✔ | |
| Clickbait [11] | | ✔ | |

Table 1: A Comparison between Concepts related to Fake News

# Is the LLM a Good Detector?

The performance of LLMs still falls slightly short of BERT.

| Model | Usage | macF1 | Accuracy | $F1_{real}$ | $F1_{fake}$ |
|---|---|---|---|---|---|
| **GPT-3.5-turbo** | Few-Shot CoT | 0.702 | 0.813 | 0.884 | 0.519 |
| **BERT** | Fine-tuning | **0.765** | **0.862** | **0.916** | **0.615** |
| **Llama-3.1-8b** | Zero-Shot | 0.621 | 0.708 | 0.802 | 0.440 |
| | Few-Shot CoT | 0.627 | 0.715 | 0.807 | 0.445 |
| **Llama-3.1-405b** | Zero-Shot | 0.658 | 0.852 | 0.915 | 0.4 |
| | Few-Shot CoT | 0.688 | 0.852 | 0.915 | 0.461 |

Table 3: Comparison of Model Performance on GossipCop dataset

# How do LLMs Perform in MSN News

- Sample Size: about 10k
- Time Period: 2024.10~11
- Modality: text
- Label: no ground truth

| Domain | Proportion |
|---|---|
| News | 33.5% |
| Sports | 31.5% |
| Money | 16.5% |
| Lifestyle | 4.0% |
| Health | 3.7% |
| Entertainment | 3.2% |
| Other | 7.6% |

| Judgment | Description | Proportion |
|---|---|---|
| **Incorrect Facts** | Facts that cannot be verified, lack credible sources, conflict with existing facts, appear fabricated, or lack detail. | **44.4%** |
| **Misleading linguistic style** | Such as sensationalism, exaggeration, subjectivity, bias, emotionality, or provocation. | **13.0%** |
| **Combination** | Combination of incorrect facts and misleading linguistic style | **22.7%** |
| **Marketing Intent** | Such as product promotions or advertisements. | **11.2%** |
| **Lack of Factual Reporting** | Such as horoscopes, essays, or lifestyle tips. | **5.5%** |
| **Other** | Such as issues related to overly brief reporting, grammatical inaccuracies, logical inconsistencies, or data preprocessing errors. | **3.2%** |

Table 5: Key Criteria Used by LLMs for Fake News Detection

# Experimental Conclusions

**Insight 1 .** *Since most LLMs are trained only once and never updated, they lack the capability to verify the authenticity of recent news events.*

**Insight 2 .** *Leveraging large-scale corpus training and advanced natural language understanding, LLMs can effectively detect biases in the linguistic style of news articles.*

**Insight 3 .** *LLMs primarily evaluate three key features of news articles (see Definition 1), with fact-checking being the most critical.*

**Case Study:**

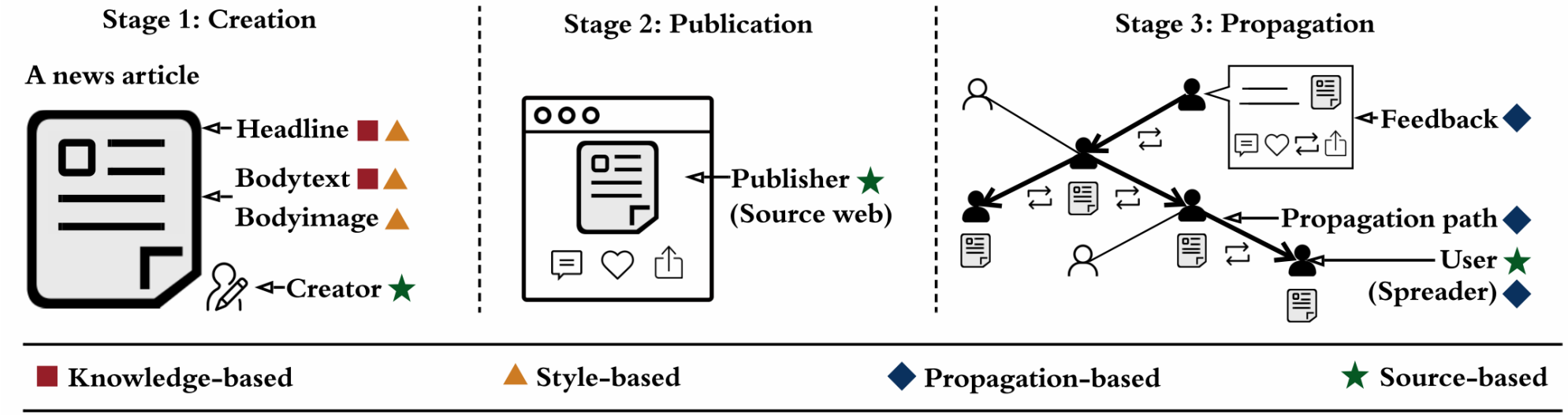
2.**核实信息来源**：文章提到了政府、财政大臣雷Rachel Reeves，这表明这些信息是基于官方公告。然而，我找不到任何关于一位名叫Rachel Reeves的财政大臣的信息，这引发了人们对这篇文章准确性的一些担忧。

**第二步：评估内容的事实准确性**
阅读内容后，我注意到文章提到卡马拉·哈里斯是民主党候选人，这似乎是有道理的。然而，我也注意到，文章提到乔·拜登因担心他的健康而下台，我无法通过可靠的消息来源证实这一点。此外，文章提到蒂姆·沃尔兹是哈里斯的竞选伙伴，这一点我无法证实。

基于这一分析，虽然这篇文章的来源是可靠的，而且似乎经过了充分的研究，但文章中的一些细节引起了注意。具体地说，文章中提到的未经证实的事件，如拜登总统放弃竞选连任，以及对特朗普的暗杀企图和重罪定罪，表明这篇文章可能是猜测或虚构的。

6.**日期和时间**：这篇文章的日期是2024年11月5日，这是未来的时间(截至我2023年的知识截止日期)。这可能表明这篇文章要么是草稿，要么是假文章

# Fake News Detection




1. **Knowledge-based** methods, which detect fake news by verifying if the knowledge within the news content is consistent with facts.
2. **Style-based** methods are concerned with how fake news is written (e.g., if it is written with extreme emotions).
3. **Propagation-based** methods, where they detect fake news based on how it spreads online.
4. **Source-based** methods detect fake news by investigating the credibility of news sources at various stages (being created, published online, and spread on social media).

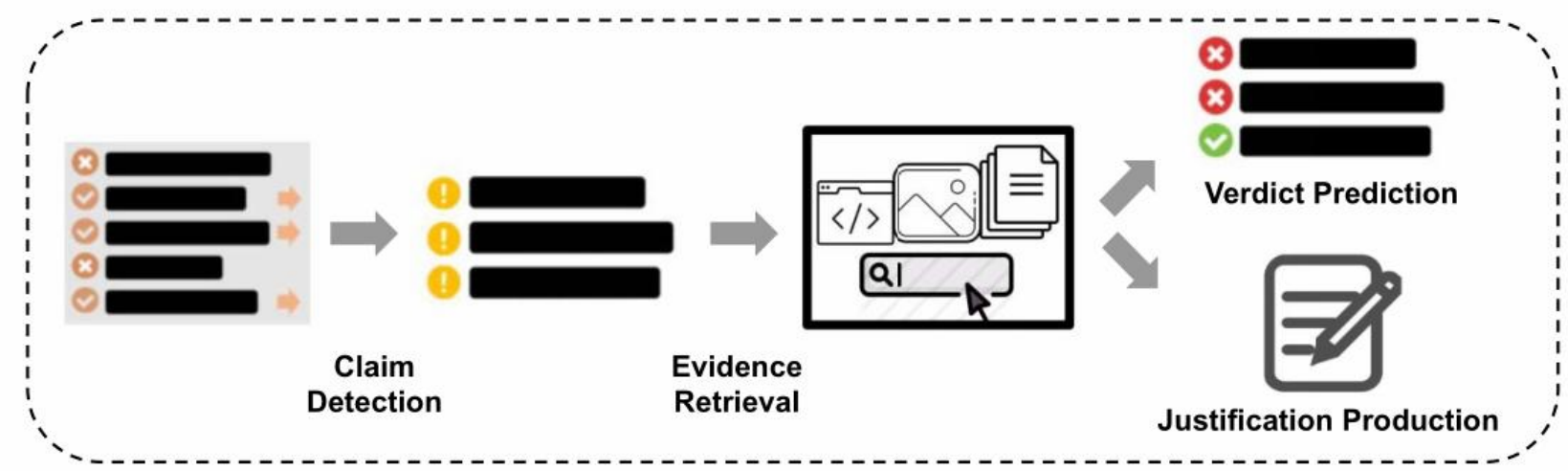
**fact-checking**



A survey of fake news: Fundamental theories, detection methods, and opportunities (CSUR, 2020)

# Fact-checking

**Definition 3 (fact-checking).** *Fact-checking is the assignment of a truth value to a claim made in a particular context based on retrieved evidence.*



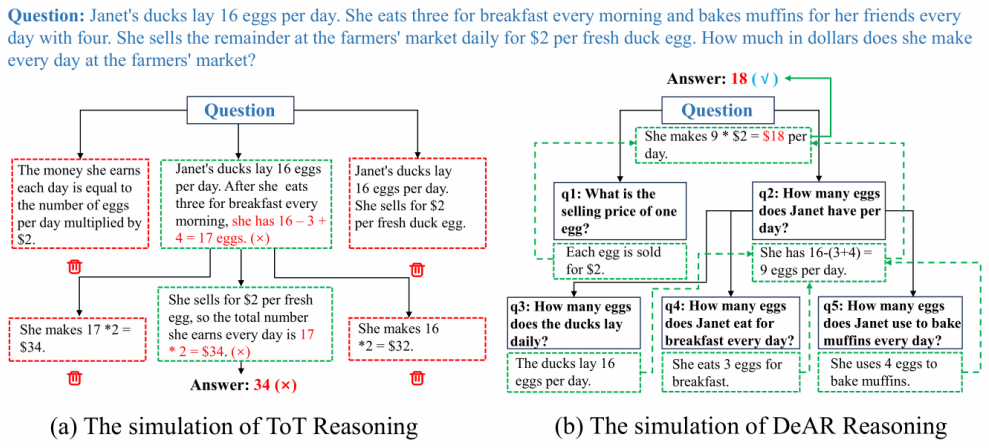

A NLP Framework for Fact-checking

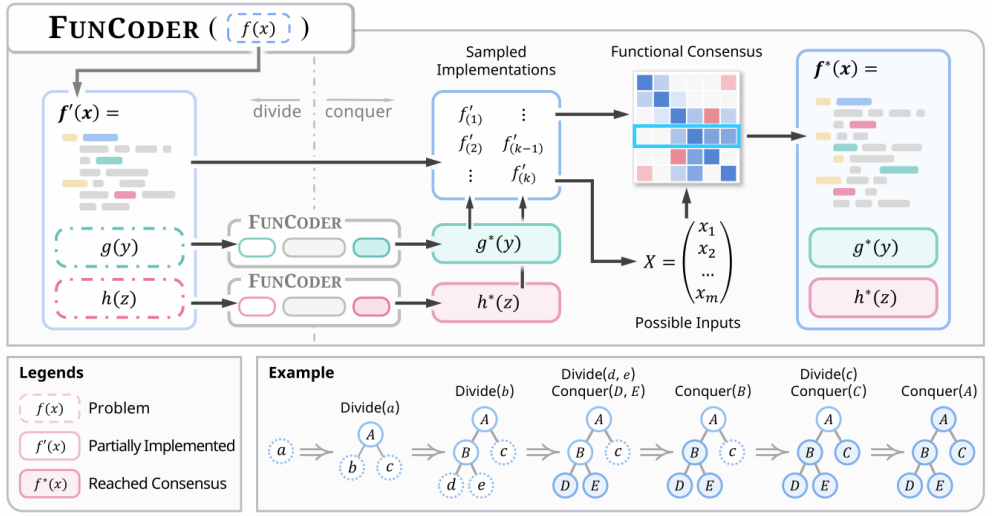
Natural language inference?
Question answering?

# Dataset

| Dataset | Input | Sources | Multi-class | Evidence | Count |
|---|---|---|---|---|---|
| **LIAR** (2017) [32] | Claim | PolitiFact | 6 | Missing | 12K |
| **FEVER** (2018) [26] | Claim | wikipedia | 3 | Sentence ID | 185K |
| **HOVER** (2020) [30] | Claim | wikipedia | 3 | Sentence ID | 26K |
| **FakeNewsNet** (2020) [28] | Article | PolitiFact<br>GossipCop | 2 | Missing | 23K |
| **FEVEROUS** (2021) [39] | Claim | wikipedia | 3 | Sentence ID | 87K |
| **RAWFC** (2022) [29] | Cliam | Snopes | 3 | Raw report | 2K |
| **LIAR-RAW** (2022) [29] | Cliam | PolitiFact | 6 | Raw report | 12K |
| **EX-FEVER** (2024)[25] | Cliam | wikipedia | 3 | Explanation | 60K |
| **LIAR2** (2024) [12] | Cliam | PolitiFact | 6 | Explanation | 23K |
| **AVeriTeC** (2024) [40] | Cliam | ClaimReview | 4 | Explanation | 4.5K |

# Decomposition Method




**Decompose, Analyze and Rethink: Solving Intricate Problems with Human-like Reasoning Cycle**

Question Answer

**NeurIPS 2024, Oral (Top 1.5%) Prompt Engineering!**

**Divide-and-Conquer Meets Consensus: Unleashing the Power of Functions in Code Generation**

Code Generation

# Least-to-most Prompting

(1) query the LLM to decompose the problem into subproblems;
(2) query the LLM to sequentially solve the subproblems. The answer to the second subproblem is built on the answer to the first subproblem.

Q: "think, machine, learning"
A: "think", "think, machine", "think, machine, learning"

**Stage1. Decomposition**

Q: "think, machine"
A: The last letter of "think" is "k". The last letter of "machine" is "e". Concatenating "k", "e" leads to "ke". So, "think, machine" outputs "ke".

**Stage2. Subproblem solving**

Q: "think, machine, learning"
A: "think, machine" outputs "ke". The last letter of "learning" is "g". Concatenating "ke", "g" leads to "keg". So, "think, machine, learning" outputs "keg".

**Least-to-most Prompting**

Q: "think, machine"
A: The last letter of "think" is "k". The last letter of "machine" is "e". Concatenating "k", "e" leads to "ke". So, "think, machine" outputs "ke".

Q: "think, machine, learning"
A: The last letter of "think" is "k". The last letter of "machine" is "e". The last letter of "learning" is "g". Concatenating "k", "e", "g" leads to "keg". So, "think, machine, learning" outputs "keg".

**Chain-of-thought prompting**

# Self-Ask

## Direct Prompting

GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Answer: Harry Vaughan Watkins.

Question: Who was president of the U.S. when superconductivity was discovered?
Answer: Franklin D. Roosevelt

## Chain of Thought

GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Answer: Theodor Haecker was 65 years old when he died. Harry Vaughan Watkins was 69 years old when he died.
So the final answer (the name of the person) is: Harry Vaughan Watkins.

Question: Who was president of the U.S. when superconductivity was discovered?
Answer: Superconductivity was discovered in 1911 by Heike Kamerlingh Onnes. Woodrow Wilson was president of the United States from 1913 to 1921. So the final answer (the name of the president) is: Woodrow Wilson.

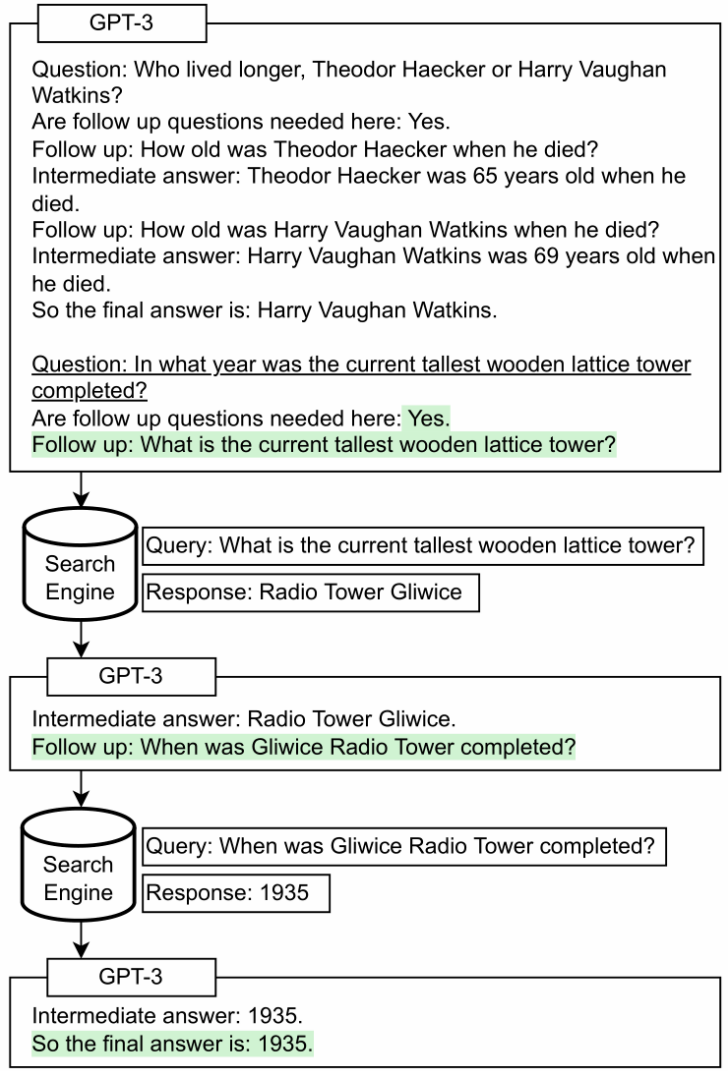
## Self-Ask

GPT-3

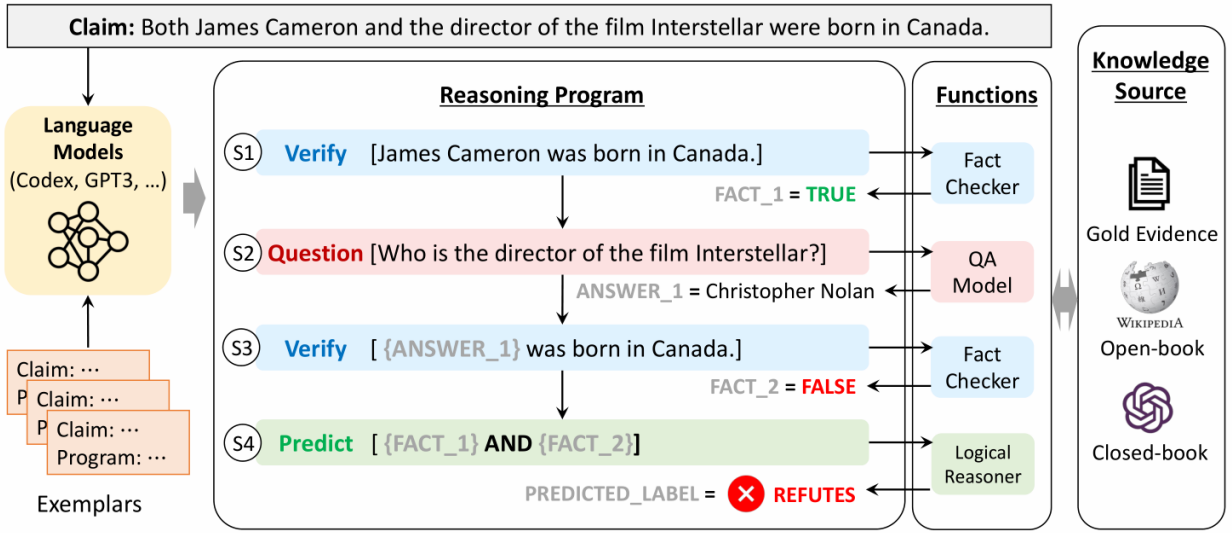
Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Are follow up questions needed here: Yes.
Follow up: How old was Theodor Haecker when he died?
Intermediate answer: Theodor Haecker was 65 years old when he died.
Follow up: How old was Harry Vaughan Watkins when he died?
Intermediate answer: Harry Vaughan Watkins was 69 years old when he died.
So the final answer is: Harry Vaughan Watkins

Question: Who was president of the U.S. when superconductivity was discovered?
Are follow up questions needed here: Yes.
Follow up: When was superconductivity discovered?
Intermediate answer: Superconductivity was discovered in 1911.
Follow up: Who was president of the U.S. in 1911?
Intermediate answer: William Howard Taft.
So the final answer is: William Howard Taft.

| | 2Wiki. | | Musique | |
|---|---|---|---|---|
| | Acc. ↑ | # Toks ↓ | Acc. ↑ | # Toks ↓ |
| Least-to-Most | 29.0 | 844 | **16.8** | 1020 |
| Self-ask | **35.5** | **569** | **16.3** | **663** |



# ProgramFC




```
'''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():
```

# FOLK

**Input: a textual claim**

- Decompose the claim into predicates(?)
- Combine SerpAPI to generate question-and-answer pairs
- reason about these question-and-answer pairs

**Output: prediction and explanation**

| | HoVER | | | FEVEROUS | | | SciFact-Open |
|---|---|---|---|---|---|---|---|
| | 2-Hop | 3-Hop | 4-Hop | Numerical | Multi-hop | Text and Table | |
| Direct | 57.11 | 44.95 | 55.91 | 48.52 | 50.18 | 59.07 | 49.70 |
| CoT | 53.98 | 46.57 | 47.99 | 49.56 | 60.90 | 61.76 | 63.39 |
| Self-Ask | 54.23 | 48.87 | 51.76 | 55.33 | 61.16 | 54.23 | 60.94 |
| ProgramFC | 71.00 | 51.04 | 52.92 | 54.78 | 59.84 | 51.69 | - |
| FOLK | 66.26 | 54.80 | 60.35 | 59.49 | 67.01 | 63.42 | 67.59 |

Macro-F1 scores

**Claim:** Lubabalo Kondlo won a silver medal in the 2012 SportAccord World Mind Games inaugurated in July 2011 in Beijing.

**Label:** *[NOT_SUPPORTED]*

**Predicates:**
Won(Lubabalo Kondlo, a silver medal) ::: Verify Lubabalo Kondlo won a silver medal
Inaugurated(the 2012 SportAccord World Mind Games, July 2011, Beijing) ::: Verify the 2012 SportAccord World Mind Games was inaugurated in July 2011 in Beijing.

**Follow-up Question:** What did Lubabalo Kondlo win in the 2012 SportAccord World Mind Games?
**Grounded Answer:** In 2012 he won the silver medal, ... in Beijing, China.

**Follow-up Question:** When and where was the 2012 SportAccord World Mind Games inaugurated?
**Grounded Answer:** The International Mind Sports Association (IMSA) inaugurated the SportAccord World Mind Games December 2011 in Beijing ...

**Prediction:**
Won(Lubabalo Kondlo, a silver medal) is **True** because In 2012 he won the silver medal at the SportAccord World Mind Games in Beijing, China.
Inaugurated(the 2012 SportAccord World Mind Games, July 2011, Beijing) is **False** because The International Mind Sports Association (IMSA) inaugurated the SportAccord World Mind Games December 2011 in Beijing.
Won(Lubabalo Kondlo, a silver medal) && Inaugurated(the 2012 SportAccord World Mind Games, July 2011, Beijing) is **False**.
The claim is *[NOT_SUPPORTED]*.

**Explanation:**
Lubabalo Kondlo won a silver medal in the 2012 SportAccord World Mind Games. However, the event was inaugurated in December 2012, not July 2011, in Beijing.

# Motivation

In some cases, the claim decomposition method **doesn't work** effectively because sub-claims may be **interdependent**.

Consider the claim: **The film in which Corey Jantzen played a wrestler on Team Foxcatcher was based on the life of a World and Olympic medalist.**

If we break it down into two sub-claims at once:

1. *Corey Jantzen played a wrestler on Team Foxcatcher in a certain film.*
2. *That film was based on the life of a World and Olympic medalist.*

**we don't know which film!**

# Basic Concepts

**Definition 4 (unit claim).** *A unit claim is a fundamental, indivisible proposition that can be independently substantiated by a single piece of knowledge.*

**Definition 5 (single-step question).** *A single-step question is a question that directly probes a unit claim, seeking a clear, verifiable answer without requiring multiple steps or reasoning.*

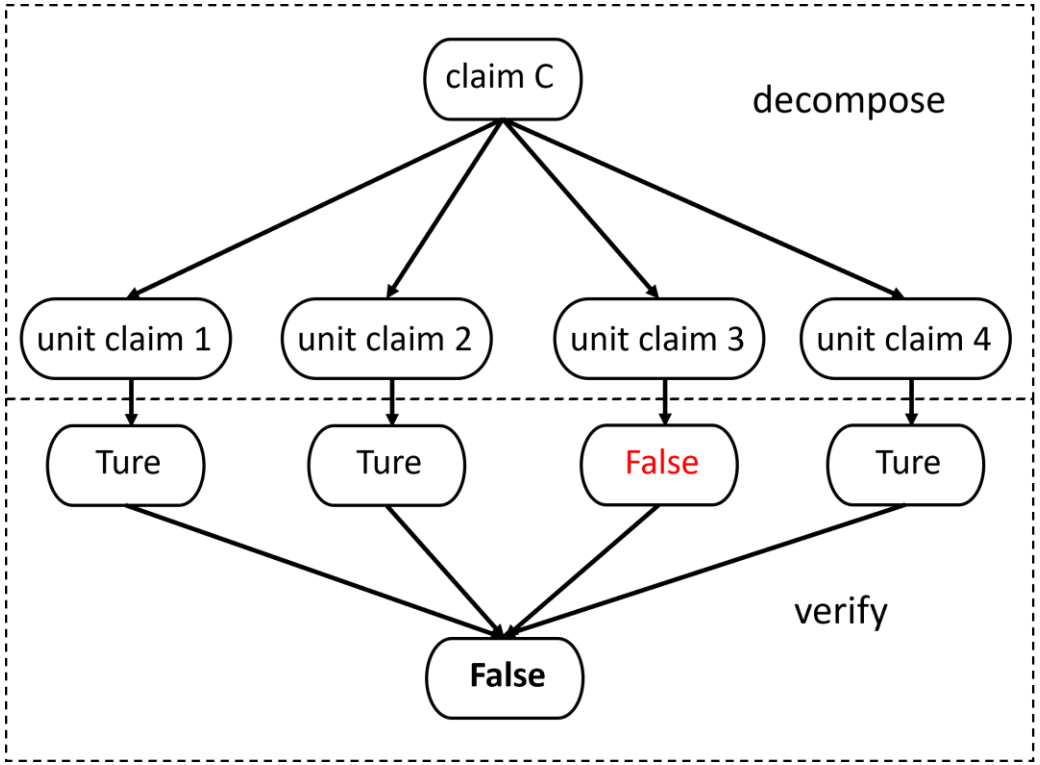
**Definition 6 (decomposition condition).** *A claim C is considered to satisfy decomposition condition if it either constitutes a unit claim or can be systematically decomposed into multiple unit claims.*

Based on the decomposition condition, we identify two common structures within the Decompose-then-Verify Framework:

- **Tree structure**
- **Sequential structure**

# Tree Structure

**Definition 7 (decomposition tree structure).** *Given a complex claim C, assume it can be decomposed into a set of sub-claims using a function* $\varepsilon : C \rightarrow \{c_1, \ldots, c_m\}$, *which satisfies decomposition condition. For each* $c_i$, *there exists a function* $\phi : c_i \rightarrow \{true, false\}$ *such that* $\phi(C) = \bigcap_{i=1}^{m} \phi(c_i)$.

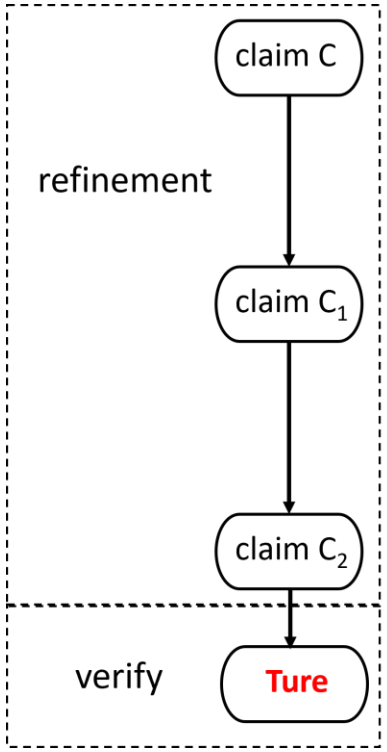



claim *C*: "Sumo wrestler Toyozakura Toshiaki committed match-fixing, ending his career in 2015 that started in 1989."

1. Toyozakura Toshiaki was a sumo wrestler. (*True*)
2. Toyozakura Toshiaki's career started in 1989. (*True*)
3. Toyozakura Toshiaki's career ended in 2015. (*False*)
4. Toyozakura Toshiaki committed match-fixing. (*True*)

# Sequential Structure

**Definition 8 (decomposition sequential structure).** *Given a complex claim C that does not satisfy the decomposition condition, let $C_0 = C$. At the i-th step, the model generates a single-step question-answer pair to refine $C_i$, transforming it into a clearer semantic representation $C_{i+1}$. After several iterations of refinement, the process terminates when $C_n$ either satisfies the decomposition condition or becomes a unit claim.*




**claim *C***: "The film in which Corey Jantzen played a wrestler on Team Foxcatcher was based on the life of a World and Olympic medalist, who was born on June 6, 1959."

**Single-step Question**: What is the title of the film in which Corey Jantzen played a wrestler on Team Foxcatcher?
**Answer**: *Foxcatcher*

**claim $C_1$**: " *Foxcatcher* was based on the life of a World and Olympic medalist, who was born on June 6, 1959."

**Single-step Question**: Who was the World and Olympic medalist that *Foxcatcher* was based on?
**Answer**: *Dave Schultz*

**claim $C_2$**: *Dave Schultz* was born on June 6, 1959.

**Single-step Question**: Was Dave Schultz born on June 6, 1959?
**Answer**: Yes

# Our method

**Algorithm 1:** Decompose-then-Verify Framework

**Input:** a claim $C$
**Output:** truth value of $C$, verification trace of $C$

```
1  while C does not satisfy decomposition condition do
2      question ← query(C);
3      answer ← retrieve(question);
4      if answer is none then
5          return False
6      C ← refine(C, question, answer)
7  c_1, c_2, ..., c_n ← decompose(C);
8  for each c in c_1, c_2, ..., c_n do
9      question ← query(c);
10     answer ← retrieve(question);
11     if verify(C, question, answer) is false then
12         return False
13 return True
```

Sequential Structure

Tree Structure

# Experiment

HOVER is a multi-hop fact verification dataset designed to challenge models to verify complex claims by retrieving and reasoning over multiple information sources, or “hops.”

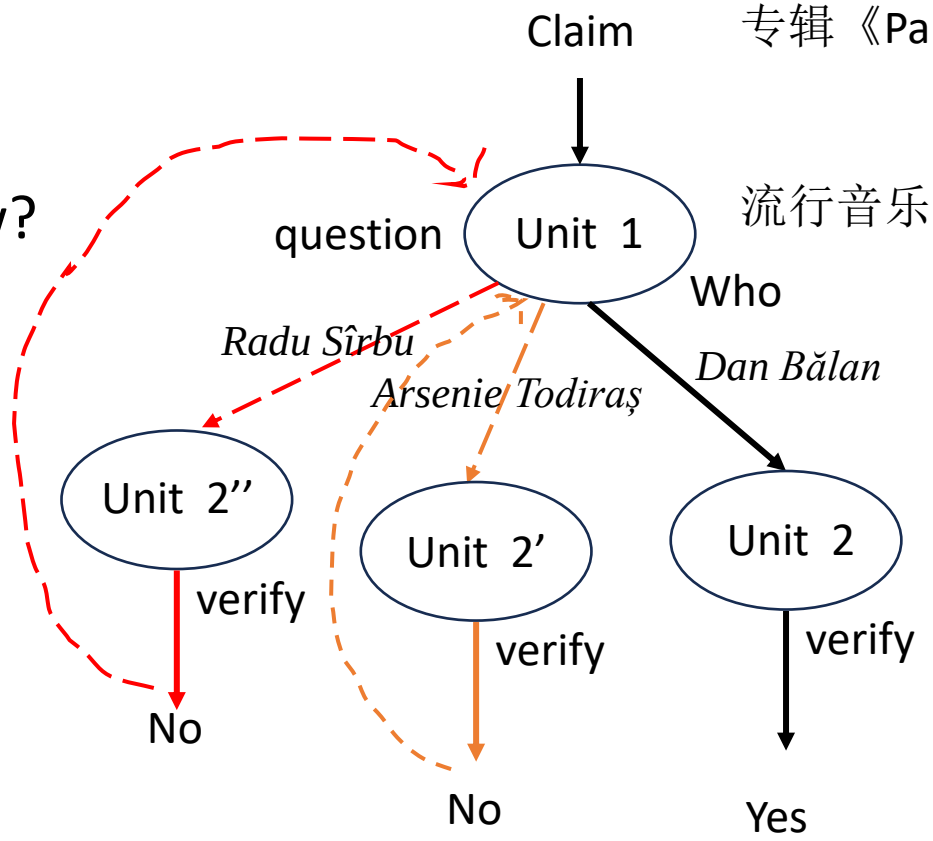
| Method | macF1 | Accuracy | $F1_{true}$ | $F1_{false}$ |
|---|---|---|---|---|
| **Zero-Shot** | 0.588 | 0.590 | 0.559 | 0.617 |
| **COT [48]** | 0.587 | 0.590 | 0.550 | 0.624 |
| **FOLK [37]** | 0.542 | 0.550 | 0.602 | 0.483 |
| **Our method** | **0.627** | **0.630** | **0.593** | **0.661** |

Table 6: Comparison of Model Performance on HOVER dataset

In this experiment, all methods are implemented using the Llama-3.1-405b model[4]. The model is not fine-tuned; instead, it relies solely on in-context learning. In both the FOLK method and our proposed approach, to generate knowledge-grounded answers for the intermediate questions, we employ a retriever based on Google Search via the SerpAPI service[5].

# Future

- Muti-agents? RAG?
- Fine-tuning LLMs using verification trace?
- Real-world claim?
- Time-consuming?
- Backtrack?
- Automatic workflow?




A member of a pop music trio O-Zone co-wrote the song from the T.I. album Paper Trail.

“一位流行音乐三人组O-Zone的成员共同创作了T.I.专辑《Paper Trail》中的一首歌曲。”

流行音乐三人组O-Zone的成员是谁？

*Dan Bălan* 共同创作了T.I.专辑《Paper Trail》中的一首歌曲。