TRACE: Tree-Structured Reasoning for Multi-Table Multi-Hop Fact-Checking

Anonymous ACL submission

Abstract

Multi-table fact-checking verifies claims whose evidence is distributed across multiple tables and connected through multi-hop reasoning. This setting is challenging because claims may contain chained or nested dependencies, table retrieval is noisy, and retrieved evidence must be selected and composed before a final verdict can be made. We introduce two multi-table fact verification testbeds, **OTT-FV** and **Hybrid-FV**, and propose **TRACE**, a tree-structured framework that connects claim decomposition, evidence retrieval, and evidence composition. TRACE represents each claim as a dependency-aware semantic tree, retrieves locally relevant table evidence for each node, and composes evidence-aware node representations bottom-up to produce the final verdict. Experiments on TabFact, OTT-FV, and Hybrid-FV show that TRACE consistently improves over strong baselines across multiple backbones under both gold- and open-evidence settings. Further analyses show that tree construction, node-level retrieval, and bottom-up evidence composition all contribute to multi-table verification.[1]

1 Introduction

Many real-world claims require evidence from more than one table. For example, verifying the claim in Figure 1 requires checking a president's term in one table, retrieving his residence information from another, and combining these pieces of evidence to assess the full claim. Multi-table multi-hop fact-checking studies this setting: the system must use evidence distributed across multiple tables and reason over their connections to determine whether a claim is supported or refuted.

Recent multi-hop fact-checking methods have made progress by combining claim decomposition, evidence retrieval, and reasoning over retrieved evidence (Wang et al., 2024; Pan et al., 2023b; Chen et al., 2022). However, directly applying this paradigm to multi-table fact-checking remains challenging. One difficulty lies in claim decomposition. Multi-table claims often contain chained or nested dependencies that cannot be easily flattened into a set of independent subclaims. For example, the claim "the capital of the European country with the highest GDP is Paris" depends on an intermediate entity, and treating it as several independent subclaims can either miss this dependency or introduce redundant verification steps.

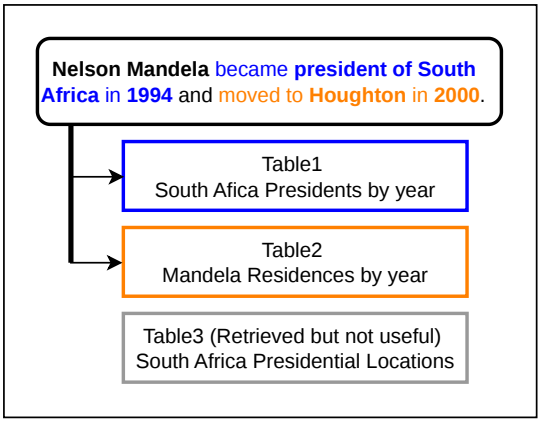

Figure 1: **A motivating example.** Verifying one claim may require combining evidence from multiple tables.

The difficulty continues after decomposition. Retrieving tabular evidence is fragile because tables provide limited textual context and distribute information across titles, headers, rows, columns, and cells (Chen et al., 2020a, 2021). A query may therefore retrieve tables that are topically related but insufficient for verifying a specific part of the claim. Increasing the number of retrieved tables can improve coverage, but it also introduces more noisy or weakly relevant evidence. Even after retrieval, the evidence must still be selected, combined, and propagated across reasoning steps. Simply passing a large set of tables to a verifier increases the burden on the model, while aggressive filtering may discard local evidence that is crucial for the final

[1] Code and datasets are included in the supplementary material and will be released upon publication.

judgment. An effective system for multi-table fact-checking therefore needs to preserve the structure of the claim, retrieve evidence for its local components, and compose the resulting evidence in a controlled way.

Thus, we introduce TRACE, a multi-table fact-checking framework that uses tree structure to explicitly connect claim decomposition, evidence retrieval, and evidence composition. TRACE first represents a claim as a dependency-aware semantic tree, so that chained or nested claims can be decomposed without losing their intermediate dependencies. It then performs node-level retrieval to gather locally relevant table evidence for each semantic unit, improving evidence coverage beyond a single global query. Finally, TRACE composes evidence-aware node representations bottom-up along the tree, allowing the system to filter, organize, and combine local evidence signals into a final verdict.

To systematically study this problem, we construct two new multi-table fact verification testbeds, OTT-FV and Hybrid-FV, derived from OTT-QA (Chen et al., 2021) and HybridQA (Chen et al., 2020b). OTT-FV and Hybrid-FV require systems to verify claims whose evidence is distributed across multiple tables and connected through 2/3/4-hop reasoning. They cover seven types of cross-table logical patterns, including conjunction, chaining, comparison, and aggregation, and provide both gold-evidence and open-evidence settings for evaluating controlled reasoning and end-to-end retrieval-plus-verification.

Experiments on TabFact, OTT-FV, and Hybrid-FV show that TRACE consistently improves over strong baselines across multiple backbones under both gold- and open-evidence settings. The gains are especially meaningful in open-evidence settings, where retrieval noise and missing evidence make flat retrieve-then-verify pipelines less reliable. Further analyses show that stronger tree construction improves both evidence retrieval and verification, node-level retrieval expands evidence coverage, and bottom-up composition helps convert locally retrieved evidence into accurate final decisions. Our contributions are summarized as follows. **(1)** We construct two multi-table multi-hop fact-checking testbeds, **OTT-FV** and **Hybrid-FV**, with gold- and open-evidence settings for evaluating controlled reasoning and retrieval-based verification. **(2)** We propose **TRACE**, a tree-structured framework that connects claim decomposition, node-level table retrieval, and bottom-up evidence composition. **(3)** We show that tree construction, node-level retrieval, and explicit evidence composition consistently benefit multi-table verification across datasets, backbones, and evidence settings.

2 Related Work

Fact-checking and claim decomposition. Fact-checking studies whether natural-language claims are supported by evidence, often requiring both retrieval and reasoning (Thorne et al., 2018). Recent systems improve verification by decomposing complex claims into atomic facts, questions, reasoning steps, or executable programs (Pan et al., 2023a; Xie et al., 2025; Pan et al., 2023b; Chen et al., 2022). These methods show that decomposition can guide search and make intermediate reasoning more explicit. However, they are mainly designed for textual evidence, web search, or tool-based verification. Our work focuses on multi-table fact-checking, where decomposition must also define table-level evidence-seeking units and preserve dependencies for cross-table evidence composition.

Table and heterogeneous-evidence fact-checking. Prior table fact-checking work mainly focuses on *single-table* verification. TabFact (Chen et al., 2020a) studies claim verification against a given table, inspiring methods that linearize tables for joint encoding (Dong and Smith, 2021; Herzig et al., 2020) or model latent operations and structured execution (Chen et al., 2020a; Wang et al., 2021). More recent work improves table verification with stronger table-aware pretraining and backbones (Gu et al., 2022). FEVEROUS (Aly et al., 2021) studies verification over heterogeneous Wikipedia evidence, including textual sentences and table cells. These works highlight the importance of structured evidence, but do not directly isolate the multi-table multi-hop setting where evidence must be retrieved from multiple tables and composed through structured reasoning.

Reasoning over structured and unstructured evidence. Multi-hop QA benchmarks such as HybridQA (Chen et al., 2020b) and OTT-QA (Chen et al., 2021) require combining evidence across structured and unstructured sources, but target answer extraction rather than claim verification. Other work introduces graph or tree structures for verification and multi-hop reasoning (Zhong et al., 2020; Zhou et al., 2019; Chen et al., 2025; Zou et al., 2023; Barik et al., 2025). In particular, Tri-

fication (Barik et al., 2025) uses tree-structured planning to organize LLM verification actions. TRACE uses tree structure differently: each semantic node drives table/row-level retrieval, obtains an evidence-aware representation, and is composed bottom-up for the final verdict.

3 Methodology

3.1 Task Definition and Notations

Following TabFact (Chen et al., 2020a), we study multi-table multi-hop fact-checking as a binary classification problem, though our model is generalizable to more fine-grained classification when other labels, e.g., not enough information, are available. Given a natural-language claim x and a corpus of tables $\mathcal{T}$, the goal is to predict a verdict $y \in \mathcal{Y} = \{\text{SUPPORTED}, \text{REFUTED}\}$. We consider two evaluation settings: **Gold** evidence, where relevant evidence is provided; and **Open** evidence, where a system must retrieve evidence from $\mathcal{T}$.

Our framework constructs a semantic tree structure for each claim and performs node-level retrieval, node encoding, and bottom-up compositional reasoning. Let the semantic tree be $G = (V, E)$, where V is the set of nodes and E is the set of tree edges. Each node $v \in V$ is associated with a node query q_v produced during tree construction. For each node, we first retrieve a small set of relevant tables E_v and then rank rows within them to obtain row-level evidence R_v. Based on the retrieved evidence, we compute an evidence-aware node representation. Finally, we compose node representations from leaves to the root to predict y. Figure 2 provides an overview of the model architecture.

3.2 Semantic Tree Construction

Since a claim that needs multiple tables for verification usually contains complex information, we decompose it for fine-grained modeling. Given a claim x, we first obtain a semantic parse and normalize it into a *tree-structured* semantic representation $G = (V, E)$. In this work, we mainly use Abstract Meaning Representation (AMR)-style semantic parses (Banarescu et al., 2013) for their robustness and efficiency. Alternative tree constructors, including dependency trees, simple rule-based trees, and LLM-generated trees, are investigated in Section 4.3.

Each node $v \in V$ corresponds to a semantically coherent subclaim. We then derive the node query q_v directly from the normalized tree node by lightly canonicalizing its surface form: we remove AMR-specific markers and relation artifacts, and retain only the core lexical content (typically a word or short phrase) that best represents the node semantics for retrieval.

3.3 Node-level Evidence Retrieval

In the **Open** setting, the system must retrieve evidence from a large table corpus $\mathcal{T}$, where relevant evidence may be scattered across multiple tables. After constructing the semantic tree, we perform retrieval *independently for each node* using its derived query q_v. Since node queries are typically short and focused, while evidence is represented as structured tables, we employ a task-specific SentenceBERT-based retriever (Reimers and Gurevych, 2019), which is fine-tuned to better align such semantic queries with serialized tabular evidence.

Concretely, we linearize each candidate table into a text sequence that preserves basic structural cues, including the title, headers, and row content. Formally, we represent each table as:

```
Title: <table title> || Header: <h1 | h2
| ...> || Rows: <row1> || <row2> || ...
```

Each row is linearized by concatenating its cell values with a vertical bar (|). We truncate long titles, headers, and row content when necessary so that the serialized table fits within the retriever's maximum input length. For example, a table with title *NBA Finals*, headers *Year, Champion, Runner-up*, and two rows is serialized as:

```
Title: NBA Finals || Header: Year |
Champion | Runner-up || Rows: 2020 |
Lakers | Heat || 2021 | Bucks | Suns.
```

This linearization preserves key table structure while remaining simple and efficient for dense retrieval.

To train the retriever, we use node queries derived from the semantic tree as supervision on the query side, paired with annotated evidence tables as positives. To improve discrimination among closely related tables, we further sample hard negatives from the same table family as the positive tables. The retriever is then optimized to assign higher similarity to relevant query–table pairs than to irrelevant but confusable ones.

At inference time, we first retrieve tables by scoring the node query against serialized tables:

$$s(q_v, t) = \text{sim}(\phi(q_v), \phi(\text{lin}(t))), \qquad (1)$$

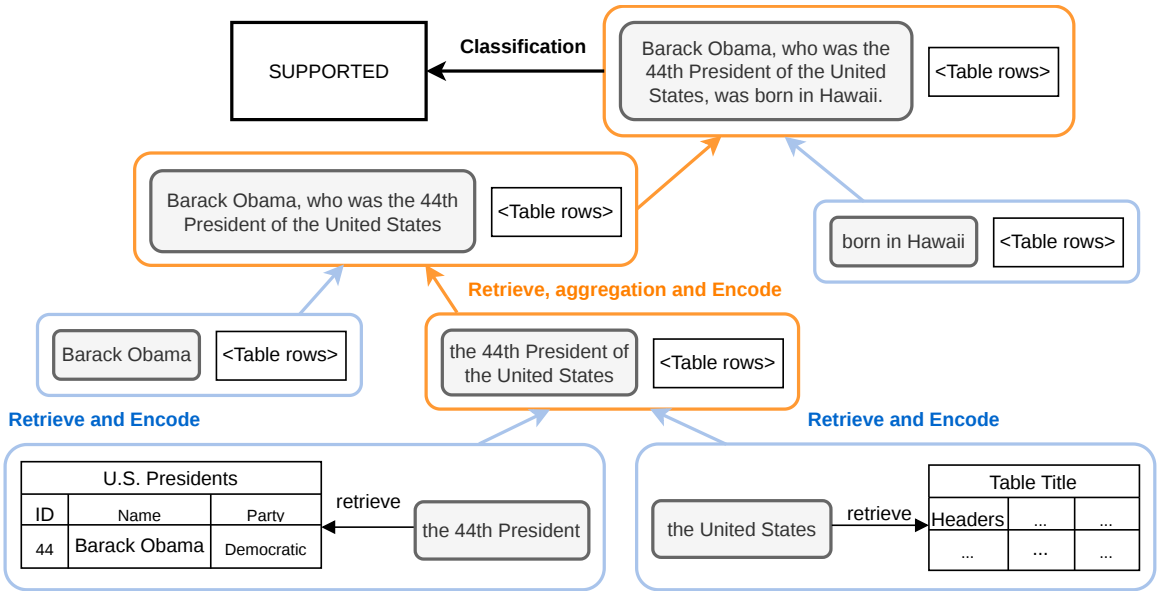

Figure 2: Overview of TRACE. We construct a semantic tree with node queries, retrieve evidence per node, encode node queries and evidence, and compose representations bottom-up to make the final decision at the root.

where $\phi(\cdot)$ denotes the fine-tuned SentenceBERT encoder, $\mathrm{lin}(\cdot)$ is the table linearization function, and $\mathrm{sim}(\cdot,\cdot)$ is a similarity measure (e.g., cosine). For each node $v \in V$, we select the top-k scoring tables:

$$E_v = f_{\text{top-}k}\{s(q_v, t) \mid t \in \mathcal{T}\}. \tag{2}$$

k is a hyperparameter; empirically, we set $k = 10$.

We then perform a row-level reranking stage to refine evidence at a finer granularity. For each node v, we collect rows from the retrieved tables in E_v, linearize each row together with its table context, and score it against the same node query using the same retriever. We retain the top-n rows as the final node-level evidence:

$$R_v = f_{\text{top-}n}\{s(q_v, r) \mid r \in \mathcal{R}(E_v)\}, \tag{3}$$

where $\mathcal{R}(E_v)$ denotes the set of candidate rows from the retrieved tables. These top-ranked rows are passed to the downstream node encoder. Empirically, we set $n = 10$.

3.4 Evidence-aware Node Encoding

We encode each node together with its row-level evidence using a pretrained LLM backbone encoder $f_\theta(\cdot)$. For node v, we first build an evidence context $\mathrm{ctx}(R_v)$ by linearizing the reranked rows in R_v and concatenating them in rank order. We then form the encoder input with a fixed template that combines the node query and its evidence context:

$$\mathbf{z}_v = \mathrm{Format}(q_v, \mathrm{ctx}(R_v)), \tag{4}$$

where $\mathrm{ctx}(R_v)$ denotes the serialized evidence context constructed from the row-level evidence R_v, and $\mathrm{Format}(\cdot,\cdot)$ denotes a simple input templating function that places the query and evidence into a single sequence. We provide the exact serialization and formatting template in Appendix C.2.

3.5 Bottom-up Compositional Reasoning

To perform multi-hop reasoning across nodes, we aggregate node representations following a post-order traversal from leaves to root. For a leaf node v, we directly set its composed representation to its evidence-aware representation, i.e., $\tilde{\mathbf{h}}_v = \mathbf{h}_v$. For a non-leaf node v with children $\mathcal{C}(v)$, we compose its own evidence-aware representation $\mathbf{h}_v$ with the set of composed child representations $\{\tilde{\mathbf{h}}_c\}_{c\in\mathcal{C}(v)}$. We instantiate composition using **PMA + FiLM** (Lee et al., 2019; Perez et al., 2018; de Vries et al., 2017), where PMA provides a permutation-invariant pooling operator for a variable-size set of child nodes, and FiLM conditionally modulates the parent representation based on the aggregated child summary.

PMA over child nodes. We apply Pooling by Multihead Attention (PMA) (Lee et al., 2019) to summarize the child representations into a fixed-size vector. Since different nodes may have different numbers of children, PMA provides a permutation-invariant way to aggregate a variable-size set of subclaims. Concretely, given the child set $\{\tilde{\mathbf{h}}_c\}_{c\in\mathcal{C}(v)}$, we use a learnable seed vector $\mathbf{s}$ to

attend over the child representations:

$$\mathbf{u}_v = \mathrm{MHA}\Big(\mathbf{s}, \{\tilde{\mathbf{h}}_c\}_{c\in\mathcal{C}(v)}, \{\tilde{\mathbf{h}}_c\}_{c\in\mathcal{C}(v)}\Big), \quad (5)$$

where $\mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ denotes multi-head attention with query $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$. We use the resulting vector $\mathbf{u}_v \in \mathbb{R}^d$ as the aggregated summary of the child nodes.

FiLM modulation. We use Feature-wise Linear Modulation (FiLM) (Perez et al., 2018; de Vries et al., 2017) to condition the parent node on aggregated child information. Specifically, given the child summary $\mathbf{u}_v$, we predict a feature-wise scale and shift:

$$\Delta\boldsymbol{\gamma}_v,\ \boldsymbol{\beta}_v = g(\mathbf{u}_v), \qquad \boldsymbol{\gamma}_v = 1 + \tanh(\Delta\boldsymbol{\gamma}_v), \quad (6)$$

where $g(\cdot)$ is an MLP. We then modulate the parent representation as

$$\tilde{\mathbf{h}}_v = \boldsymbol{\gamma}_v \odot \mathbf{h}_v + \boldsymbol{\beta}_v, \quad (7)$$

where $\odot$ denotes element-wise product. The reparameterization $\boldsymbol{\gamma}_v = 1 + \tanh(\Delta\boldsymbol{\gamma}_v)$ keeps the scaling centered around 1.

3.6 Root-level Prediction and Training

Let r be the root node. After bottom-up composition, we obtain $\tilde{\mathbf{h}}_r$ and predict the veracity by

$$\hat{\mathbf{p}} = \mathrm{softmax}(\mathbf{W}\tilde{\mathbf{h}}_r + \mathbf{b}), \qquad \hat{y} = \arg\max_{y\in\mathcal{Y}} \hat{\mathbf{p}}_y, \quad (8)$$

where $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters. Given retrieved evidence, we jointly train the encoder, composition modules, and classifier using cross-entropy loss:

$$\mathcal{L} = -\sum_{y\in\mathcal{Y}} p_y \log \hat{p}_y. \quad (9)$$

4 Experiments

4.1 Experimental Setup

Task settings and datasets. We evaluate binary tabular fact-checking with SUPPORTED and REFUTED labels under two evidence settings. In the **Gold** setting, the required tabular evidence is provided; in the **Open** setting, systems must first retrieve relevant tables from a large table corpus. We report results on **OTT-FV** and **Hybrid-FV**, two multi-table multi-hop fact-checking testbeds introduced in this work, as well as **TabFact** (Chen et al., 2020a), a widely used single-table benchmark whose open-evidence setting still requires corpus-level table retrieval. Detailed dataset construction and statistics are provided in Appendix A.

Baselines. We compare TRACE with representative baselines, including general backbones (**DeBERTa-v3** (He et al., 2021), **Qwen3-8B** (Yang et al., 2025)), table-oriented methods (**PASTA** (Gu et al., 2022), **Table-llama** (Zhang et al., 2024)), LLM-based fact-checkers (**Factcheck-GPT** (Wang et al., 2024), **DCUF** (Hu et al., 2022)), and industrial LLMs (**GPT-5.4** (OpenAI, 2026), **Gemini-2.5-pro** (Comanici et al., 2025)). We further instantiate TRACE with different backbones to evaluate its adaptability. Additional discussion on baseline selection is provided in Appendix E.1.

Metrics. Following prior work on table-based and evidence-grounded fact verification (Chen et al., 2020a), we report **Accuracy** and **Macro-F1**. For evidence retrieval, we additionally report evidence coverage metrics (Zheng et al., 2024).

4.2 Main Results and Analyses

Overall results. Table 1 reports verification results. In the more realistic **Open** setting, the best-performing TRACE variants achieve the strongest results across all three benchmarks. The gains are especially clear on **OTT-FV** and **Hybrid-FV**, where evidence is distributed across multiple tables and verification requires multi-hop evidence composition. These results support our central motivation that multi-table fact-checking benefits from explicitly structuring both evidence retrieval and evidence composition.

In the **Gold** setting, where the evidence space is constrained, TRACE remains competitive and achieves the best results on TabFact and OTT-FV with suitable backbones, while industrial LLM verifiers remain advantageous on Hybrid-FV. This pattern suggests that when evidence is already provided, model capacity and table-specific pretraining play a larger role, especially on Hybrid-FV Gold; when evidence must be retrieved from a corpus, structured retrieval and composition become more important differentiators.

TRACE is backbone-agnostic. TRACE improves a range of verifier backbones in both settings, including general neural backbones, open-source LLMs, and table-oriented models. This suggests that the framework is not tied to a specific verifier architecture, but can provide a general structured retrieval-and-composition layer for different backbones in multi-table fact-checking.

Why Open evidence benefits more. The larger gains in the Open setting align with the design

Model	TabFact		OTT-FV		Hybrid-FV	
	Gold	Open	Gold	Open	Gold	Open
DeBERTa-v3	86.1/85.4	78.8/77.9	67.4/66.2	62.5/61.7	56.2/56.3	53.7/53.1
Qwen3-8B	86.7/86.0	80.5/79.6	68.2/67.1	64.6/63.8	60.1/60.1	53.3/54.2
PASTA	89.4/87.7	78.5/77.6	70.1/69.0	62.8/61.9	58.2/56.2	54.1/56.8
Table-llama	87.5/86.6	80.8/79.9	69.0/67.9	65.0/64.1	51.2/50.3	53.5/52.6
DCUF	86.1/85.2	82.7/81.9	64.7/63.6	63.1/62.2	56.1/55.0	50.8/49.9
Factcheck-GPT	82.0/81.3	80.6/79.8	74.6/73.8	68.3/67.5	63.3/64.2	56.4/55.5
GPT-5.4	87.4/86.4	81.0/81.3	74.2/73.5	70.8/67.6	**64.0/64.8**	55.7/55.9
Gemini-2.5-pro	76.3/75.4	74.2/73.4	70.9/69.8	66.9/66.0	62.8/61.9	55.2/54.3
Ours (DeBERTa)	88.8/88.1	83.1/82.3	72.8/71.9	72.1/71.2	61.8/60.9	59.2/**59.1**
Ours (Qwen3-8B)	**89.4/88.6**	**83.5/82.7**	74.2/73.3	**72.4/71.6**	61.0/60.1	**59.7**/58.9
Ours (PASTA)	87.1/86.5	81.8/81.3	**76.3/74.2**	66.7/67.2	60.8/59.3	55.7/56.4
Ours (Table-llama)	87.8/86.3	81.2/80.3	71.0/68.4	69.5/65.7	52.4/50.6	55.5/54.4

Table 1: Fact-checking results (Accuracy (%) / Macro-F1 (%)) on three benchmarks under Gold and Open settings.

goal of TRACE. Node-level retrieval turns a single global retrieval problem into a set of focused evidence-seeking steps, improving coverage of locally relevant table evidence. The bottom-up composition module then provides a structured path for filtering, organizing, and combining these local evidence signals into a final verdict. This structure is particularly useful when retrieval noise and missing evidence make flat retrieve-then-verify pipelines less reliable.

Discussion on strong table-centric backbones. The gains are smaller in some settings with strong table-centric backbones such as Table-llama, suggesting that table-specific pretraining can already capture part of the required reasoning behavior. However, TRACE still improves Open-setting performance in most cases, indicating that explicit structure remains useful even when the verifier has strong table modeling ability. We further analyze the roles of tree construction, retrievers, and composition modules in the following sections.

4.3 Tree Construction Analyses

We further analyze how tree construction affects retrieval and verification. We keep the retriever, the number of retained evidence rows, and the composition module fixed, and vary only the way a claim is converted into a tree. We report **Macro-F1** for end-to-end verification and **Evidence Recall@U**, computed over the union of retrieved rows across all tree nodes after deduplication.

Table 2 shows that tree structure has a substantial impact on both evidence recall and verification performance. The root-only variant performs worst on both benchmarks, indicating that treating the entire claim as a single retrieval unit fails to cover the evidence needed for multi-table multi-hop verification. In contrast, all tree-based variants improve evidence recall and Macro-F1, showing that decomposition directly affects the evidence pathway.

The results also show a clear trend with tree quality. Simple rule-based segmentation provides limited gains, while dependency-based and AMR-based trees achieve stronger performance. This suggests that more semantically aligned structures produce better node-level retrieval queries and provide more useful evidence for downstream composition. Although LLM-generated trees achieve the best results, we use AMR-based trees as the main implementation for their stability, reproducibility, model independence, and lower cost. The LLM-tree results therefore serve as a headroom analysis, showing the potential of TRACE with higher-quality claim structures.

4.4 Retrieval Method Study

We study the effect of the node-level retriever by keeping the rest of TRACE fixed. We compare our task-specific retriever with several generic retrieval pipelines, including BM25 (Robertson and Zaragoza, 2009), SentenceBERT, and BM25 with an additional reranking stage. We report both end-to-end **Macro-F1** and **Evidence Recall@U**.

Table 3 shows that the task-specific retriever consistently outperforms generic retrievers on both datasets. Replacing it with BM25 or Sentence-

Tree Construction	OTT-FV		Hybrid-FV	
	Macro-F1 (%)	Evid. Recall@U (%)	Macro-F1 (%)	Evid. Recall@U (%)
AMR-tools tree	71.2	80.4	59.1	46.8
Dependency tree	68.9	70.6	57.2	40.5
Simple rule-based segmentation tree	65.0	52.5	55.9	34.4
LLM-generated tree	75.6	84.6	63.5	56.8
No tree (root-only)	61.3	40.7	53.7	18.7

Table 2: **Tree construction analyses.** Macro-F1 and evidence recall across five tree construction strategies.

Retriever	OTT-FV		Hybrid-FV	
	Macro-F1 (%)	Evid. Recall@U (%)	Macro-F1 (%)	Evid. Recall@U (%)
Ours (task-specific)	71.2	80.4	59.1	46.8
BM25	64.5	52.3	52.5	21.9
SentenceBERT	66.1	63.7	53.9	30.5
BM25 + reranking	65.7	69.2	52.8	24.6

Table 3: **Retrieval method study.** Macro-F1 and evidence recall with different node-level retrievers on OTT-FV and Hybrid-FV.

BERT substantially reduces evidence recall, especially on Hybrid-FV, where evidence is more dispersed across tables and precise subclaim-to-evidence alignment is more important. The drop in recall is accompanied by lower Macro-F1, indicating that retrieval errors directly limit downstream composition and verification. These results support our design choice of using node-level table retrieval tailored to semantic tree nodes, rather than relying on off-the-shelf retrievers to match decomposed claims to structured table evidence.

4.5 Component Ablations

We ablate two key components in the bottom-up composition module: **PMA**, which pools variable-size child node representations, and **FiLM**, which conditionally fuses aggregated child information into the parent representation. All experiments are conducted under the **Open** setting while keeping tree construction, node-level retrieval, and verifier backbones fixed.

Table 4 shows that removing either component consistently reduces performance across datasets and backbones. This indicates that the gains of TRACE do not come only from retrieving node-level evidence; the model also benefits from explicitly composing these local evidence signals along the tree structure. In most settings, removing PMA causes larger drops than removing FiLM, suggesting that stable aggregation over child nodes is especially important for bottom-up reasoning. FiLM further improves performance by allowing the parent representation to be conditioned on the composed child information.

The effect is more pronounced on the multi-table benchmarks, especially OTT-FV and Hybrid-FV, where verification often requires combining evidence from multiple subclaims. This supports our motivation that multi-table fact-checking requires not only better evidence coverage, but also an explicit mechanism for filtering, organizing, and integrating evidence across nodes. The smaller drops observed for Table-llama in some cases suggest that strong table-centric backbones may already capture part of this compositional behavior, or that performance in these settings is partly constrained by retrieval quality.

4.6 Tree Retrieval vs. Bottom-up Composition

To further separate the effect of tree-structured retrieval from bottom-up evidence composition, we add an intermediate diagnostic baseline under the Open setting. This baseline uses the same semantic tree and node-level retriever as TRACE, but directly concatenates all retrieved node evidence and feeds the flattened evidence to the verifier, without the PMA+FiLM composition module. This setting tests whether the gains of TRACE can be explained mainly by retrieving better node-level evidence, or whether explicitly composing the retrieved evidence along the tree provides additional benefits.

Table 5 shows that tree retrieval with flattened evidence provides only limited improvement over the backbone-only baselines and remains substantially below full TRACE. This suggests that higher evidence coverage does not automatically translate into stronger verification. Once node-level evidence is flattened into a single context, the veri-

Model	TabFact (Open)			OTT-FV (Open)			Hybrid-FV (Open)		
	Full	w/o FiLM	w/o PMA	Full	w/o FiLM	w/o PMA	Full	w/o FiLM	w/o PMA
Ours (DeBERTa)	**82.3**	79.6	77.8	**71.2**	66.9	64.1	**59.1**	55.4	52.6
Ours (Qwen3-8B)	**82.7**	80.1	78.3	**71.6**	67.8	65.0	**58.9**	55.9	53.1
Ours (PASTA)	**81.3**	78.7	76.9	**67.2**	63.1	60.4	**56.4**	53.0	51.7
Ours (Table-llama)	**80.3**	78.2	74.7	**65.7**	60.9	60.4	**54.4**	53.8	54.1

Table 4: **Component ablation study.** Macro-F1 scores under the Open setting after removing FiLM or PMA from the bottom-up composition module.

Backbone	Method	OTT-FV (Open)	Hybrid-FV (Open)
DeBERTa-v3	Backbone only	62.5 / 61.7	53.7 / 53.1
	Tree retrieval + flattened evidence	63.1 / 64.0	52.5 / 51.1
	TRACE	**72.1 / 71.2**	**59.2 / 59.1**
Qwen3-8B	Backbone only	64.6 / 63.8	53.3 / 54.2
	Tree retrieval + flattened evidence	65.2 / 65.1	54.4 / 53.9
	TRACE	**72.4 / 71.6**	**59.7 / 58.9**

Table 5: **Tree retrieval vs. bottom-up composition.** Results are reported as Accuracy / Macro-F1 under the Open setting. The diagnostic baseline uses the same tree-structured node-level retrieval as TRACE, but flattens all retrieved evidence before verification, without PMA+FiLM bottom-up composition.

Model	OTT-FV Accuracy (%)			Hybrid-FV Accuracy (%)		
	2-hop	3-hop	4-hop	2-hop	3-hop	4-hop
Ours (DeBERTa)	72.8	63.9	56.1	61.8	57.3	54.8
Ours (Qwen3-8B)	74.2	65.1	57.3	61.0	57.9	55.6
Ours (PASTA)	76.3	66.4	58.2	60.8	56.8	54.3
Ours (Table-llama)	71.6	67.0	59.5	57.2	55.6	52.8

Table 6: **Hop breakdown on OTT-FV and Hybrid-FV.** Accuracy grouped by hop count. As hop count increases, performance converges toward the random baseline (50%). Our models remain consistently above chance, indicating stronger robustness under high-hop multi-table reasoning.

fier must implicitly identify useful evidence, filter noisy rows, and recover cross-node dependencies from an unordered evidence pool. This is especially challenging in multi-table settings, where evidence retrieved for different nodes can play different roles in the reasoning process.

In contrast, TRACE preserves the semantic tree after retrieval and composes evidence-aware node representations bottom-up. This allows the model to organize local evidence signals according to the claim structure, rather than asking the verifier to infer the entire reasoning path from a flattened context. These results show that tree-structured retrieval and bottom-up composition are complementary: node-level retrieval improves the availability of locally relevant evidence, while PMA+FiLM composition helps convert these structured evidence signals into stronger final verification.

4.7 Hop Breakdown

We analyze performance by reasoning depth, grouping examples by hop count (2/3/4-hop) and reporting **Accuracy**. Since the task is binary, random guessing corresponds to 50% accuracy.

Table 6 shows that accuracy generally decreases as hop count increases, confirming that verification becomes harder when more evidence links must be retrieved and composed. The degradation is most visible in 4-hop examples, where missing evidence and error accumulation make the task approach a harder binary regime. Differences among backbones also become smaller in these cases, suggesting that high-hop verification exposes shared bottlenecks in evidence coverage and composition. Nevertheless, models remain above chance in most 4-hop settings, indicating non-trivial robustness under deeper multi-hop reasoning.

Case study. We present a case study to visually illustrate our framework. See Appendix F.

5 Conclusion

We study multi-table multi-hop fact-checking and present **TRACE**, a semantic tree-based framework that explicitly structures both retrieval and fact-checking reasoning. TRACE constructs a semantic tree with node queries, performs node-level evidence retrieval, and composes evidence-aware node representations bottom-up to make a final verification decision. We introduce two multi-table fact verification testbeds, **OTT-FV** and **Hybrid-FV**. TRACE achieves consistent improvements, with particularly strong benefits under open evidence. Our analyses further show that tree construction quality and task-specific node-level retrieval are critical, and that the composition modules contribute substantially to performance.

Limitations

Our study has several limitations. First, although TRACE improves multi-table multi-hop fact-checking by introducing tree-structured decomposition and bottom-up reasoning, its performance still depends on the quality of the constructed semantic tree. Errors in parsing or subclaim decomposition may propagate to downstream retrieval and composition, especially in high-hop settings where reasoning chains are longer and intermediate mistakes accumulate more easily.

Second, our retrieval framework is still constrained by evidence coverage. Even with node-level retrieval, the correct evidence may be missed when relevant rows are lexically sparse, semantically implicit, or distributed across tables in ways that are difficult to capture with the current retrieval setup. As a result, some failures may stem from retrieval bottlenecks rather than reasoning errors alone.

Finally, stronger tree construction methods, such as LLM-based decomposition, may improve performance but also introduce additional computational cost and variability. This creates a practical trade-off between accuracy, stability, and efficiency that is not fully explored in the current work.

Potential Risks

Our work studies fact-checking over tabular evidence and is intended to improve the reliability of automated information systems. However, such systems may still make incorrect verification decisions when the retrieved evidence is incomplete, misleading, or only partially relevant. If deployed in real-world settings without human oversight, these errors could lead users to place undue trust in incorrect judgments. In addition, although our datasets are constructed from public table corpora and are designed for research use, automated claim generation and verification pipelines may inherit biases or factual inaccuracies from the underlying source tables. We therefore view TRACE as a decision-support tool for research rather than a fully reliable replacement for human fact-checking in high-stakes settings.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. FEVEROUS: Fact extraction and VERification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Anab Maulana Barik, Shou Ziyi, Yang Kaiwen, Yang Qi, and Shen Xin. 2025. Trification: A comprehensive tree-based strategy planner and structural verification for fact-checking. *Preprint*, arXiv:2512.00267.

Jifan Chen, Aniruddh Sriram, Eunsol Choi, and Greg Durrett. 2022. Generating literal and implied subquestions to fact-check complex claims. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3495–3516, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. 2021. Open question answering over tables and text. In *International Conference on Learning Representations*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Yingjian Chen, Haoran Liu, Yinhong Liu, Jinxiang Xie, Rui Yang, Han Yuan, Yanran Fu, Peng Yuan Zhou, Qingyu Chen, James Caverlee, and Irene Li. 2025. GraphCheck: Breaking long-term text barriers with extracted knowledge graph-powered fact-checking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14976–14995, Vienna, Austria. Association for Computational Linguistics.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville.

2017. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Rui Dong and David Smith. 2021. Structural encoding and pre-training matter: Adapting BERT for table-based fact verification. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. PASTA: Table-operations aware fact verification via sentence-table cloze pre-training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Nan Hu, Zirui Wu, Yuxuan Lai, Xiao Liu, and Yansong Feng. 2022. Dual-channel evidence fusion for fact verification over texts and tables. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

OpenAI. 2026. Introducing gpt-5.4. `https://openai.com/index/introducing-gpt-5-4/`. Accessed: 2026-05-24.

Liangming Pan, Xinyuan Lu, Min-Yen Kan, and Preslav Nakov. 2023a. QACheck: A demonstration system for question-guided multi-hop fact-checking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 264–273, Singapore. Association for Computational Linguistics.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023b. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6981–7004, Toronto, Canada. Association for Computational Linguistics.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. 2018. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 809–819.

Nancy X. R. Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 317–326, Online. Association for Computational Linguistics.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. 2024. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14199–14230, Miami, Florida, USA. Association for Computational Linguistics.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. 2025. FIRE: Fact-checking with iterative retrieval and verification. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2901–2914. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. TableLlama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (Volume 1: Long Papers), pages 6024–6044, Mexico City, Mexico. Association for Computational Linguistics.

Liwen Zheng, Chaozhuo Li, Xi Zhang, Yu-Ming Shang, Feiran Huang, and Haoran Jia. 2024. Evidence retrieval is almost all you need for fact verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9274–9281.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180, Online. Association for Computational Linguistics.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. Gear: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901. Association for Computational Linguistics.

Anni Zou, Zhuosheng Zhang, and Hai Zhao. 2023. Decker: Double check with heterogeneous knowledge for commonsense fact verification. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11891–11904.

A Dataset Construction and Statistics

A.1 Source Table Corpora

To construct multi-table multi-hop fact verification benchmarks, we build on the source table corpora underlying two existing QA resources. For **OTT-FV**, we use the official table corpus released with OTT-QA. For **Hybrid-FV**, we use the Wikipedia table pool underlying HybridQA, i.e., the original source corpus rather than only the subset of tables explicitly appearing in HybridQA examples. This setup allows our Open evidence evaluation to more faithfully reflect large-scale retrieval over a realistic table pool.

A.2 Claim Construction

We construct claims by first sampling groups of related tables from the source corpus and then prompting `gpt-4o` to identify table combinations that support a fact requiring information from multiple tables. The goal is to create claims that cannot be verified from a single table alone, and instead require cross-table composition. This design encourages multi-table and multi-hop reasoning patterns rather than single-table lookup.

More specifically, given a set of candidate tables, we ask `gpt-4o` to determine whether there exist supporting tables with overlapping or related information that can be combined into a true factual claim. For each valid instance, the model is instructed to generate (1) a concise claim, (2) the identifiers of the supporting tables, and (3) a relation type describing how the evidence is combined. The allowed relation types are CONJUNCTIVE, CHAINED, COMPARATIVE, CONDITIONAL, CAUSAL, AGGREGATION, and TAXONOMIC. If no suitable table combination can be found, the model outputs `no`. We provide the full prompt in Appendix A.3 for reproducibility.

For **supported** claims, the generated statement must be fully grounded in the selected tables. For **refuted** claims, we minimally perturb a supported claim so that the incorrect part directly conflicts with the table evidence, rather than introducing content that is merely unsupported or unverifiable. This construction ensures that refuted examples remain evidence-based and require genuine verification against the tables.

All generated claim–label pairs are manually verified by 20 graduate students in natural language processing. Annotators are shown each generated claim together with the designated evidence tables, and are asked to determine whether the claim is SUPPORTED or REFUTED by the provided evidence. They also correct noisy labels produced during automatic generation. The agreement between the `gpt-4o`-proposed labels and the final human-verified labels is **88.10%**. This number measures the consistency between automatic candidate labels and final human judgments, rather than inter-annotator agreement, and shows the necessity of human verification in our construction pipeline.

A.3 Prompt for Claim Construction

We use the following prompt to guide `gpt-4o` in constructing multi-table claims:

> You are given several tables from a series. Your task is to check whether there are TWO tables that contain overlapping or related information that can be combined to construct a TRUE fact-based claim. A valid claim must be verifiable using information from both tables (neither table alone is sufficient).
>
> Each table has a unique ID (uid). Your output must have exactly four lines:
>
> 1. A single concise factual claim (similar in style to FEVER/HoVer). 2. The uid of the first supporting table. 3. The uid of the second supporting table. 4. One logic-relation label, chosen from the

year	country	president	term_note
1994	South Africa	Nelson Mandela	Took office on 1994-05-10
1995	South Africa	Nelson Mandela	In office
1996	South Africa	Nelson Mandela	In office
1997	South Africa	Nelson Mandela	In office
1998	South Africa	Nelson Mandela	In office
1999	South Africa	Nelson Mandela / Thabo Mbeki	Mandela served until 1999-06-14; Mbeki succeeded him in June 1999
2000	South Africa	Thabo Mbeki	In office

Table 7: South Africa presidents by year.

year	person	main_role
1993	Nelson Mandela	ANC leader / anti-apartheid statesman
1994	Nelson Mandela	President of South Africa
1995	Nelson Mandela	President of South Africa
1996	Nelson Mandela	President of South Africa
1997	Nelson Mandela	President of South Africa
1998	Nelson Mandela	President of South Africa
1999	Nelson Mandela	President of South Africa / retired statesman
2000	Nelson Mandela	Former president; public humanitarian work

Table 8: Nelson Mandela roles by year.

> list: CONJUNCTIVE, CHAINED, COMPARATIVE, CONDITIONAL, CAUSAL, AGGREGATION, TAXONOMIC.
>
> Do not quote or paraphrase the table contents inside the claim. If no suitable pair of tables can be found, output exactly one line: no. Output nothing else (no explanations, no blank lines).

For 3-hop and 4-hop examples, we use the same prompt template with only minimal modifications: the required number of supporting tables and the number of returned table IDs are adjusted accordingly.

A.4 Annotation Quality Audit

To further assess annotation quality, we conduct an additional double-annotation audit. For each dataset, we sample 200 examples balanced by label, including 100 originally labeled SUPPORTED and 100 originally labeled REFUTED. A second annotator independently re-labels each example using the claim and the designated evidence tables, without access to the original label. Disagreements in the audited subset are resolved by a third annotator before finalizing the revised dataset.

Table 13 reports the audit results. The second annotation achieves high agreement with the original human-verified labels on both datasets, with an overall raw agreement of 95.25% and Cohen's κ of 0.905. This suggests that the constructed labels are reliable despite the use of automatic candidate generation.

A.5 Dataset Statistics

The final dataset statistics are as follows.

OTT-FV. OTT-FV contains **951** training examples and **435** test examples, corresponding to **68.61% / 31.39%** of the full dataset, respectively. The underlying table pool contains approximately **400K** tables. The label distribution is **55.47%** supported and **44.53%** refuted. Its hop-count distribution is: **2-hop: 1087**, **3-hop: 152**, and **4-hop: 147**. Its relation-type distribution is: **Conjunctive: 649**, **Chained: 333**, **Aggregation: 183**, **Comparative: 132**, **Taxonomic: 69**, **Conditional: 18**, and **Causal: 2**.

Hybrid-FV. Hybrid-FV contains **2400** training examples and **800** test examples, corresponding to **75.00% / 25.00%** of the full dataset, respectively. The underlying table pool contains approximately **1M** tables. The label distribution is **53.15%** supported and **46.85%** refuted. Its hop-count distribution is: **2-hop: 2260**, **3-hop: 680**, and **4-hop: 260**. Its relation-type distribution is: **Conjunctive: 1624**, **Chained: 845**, **Aggregation: 257**, **Comparative: 239**, **Conditional: 109**, **Taxonomic: 107**, and **Causal: 19**.

A.6 Data Privacy, Human Annotation and Sensitive Content

Our datasets are constructed from public table corpora derived from Wikipedia and prior QA resources. As a result, some tables may contain names of real people or other publicly available entity information. We do not collect private personal data, and we do not add new personally identifying information beyond what already appears in the

period	person	location	residence_type
late 1992–1999	Nelson Mandela	Houghton, Johannesburg	
1994–1999	Nelson Mandela	Pretoria	
1994–1999	Nelson Mandela	Cape Town	
from retirement in 1999 (incl. 2000)	Nelson Mandela	Houghton, Johannesburg	Post-presidential home/office base

Table 9: Mandela residence-related evidence by period.

capital	type	country
Pretoria	administrative capital	South Africa
Cape Town	legislative capital	South Africa
Bloemfontein	judicial capital	South Africa

Table 10: South Africa capitals.

site	city
Union Buildings	Pretoria
Mahlamba Ndlopfu	Pretoria
Tuynhuys	Cape Town
Genadendal	Cape Town

Table 11: South African official government sites.

public source tables. During dataset construction, we manually verify generated claims and labels, and we do not intentionally include offensive, hateful, or sexual content. Because the source corpora are large and open-domain, we cannot completely rule out the presence of isolated sensitive or offensive entries inherited from the original tables; however, the benchmark is intended for research use on public factual data rather than for profiling individuals.

To verify the quality of generated claim–label pairs, we recruited 20 graduate students from our university as volunteer annotators. Annotators were shown a generated claim together with the corresponding gold table(s), and were asked to judge whether the claim was factually supported or refuted by the provided evidence. They were also asked to correct annotation mistakes when necessary. Participation was voluntary and unpaid. Annotators were informed that their annotations would be used for research and paper writing, and all participation was based on consent. To further assess label reliability, we additionally conducted the double-annotation audit described in Appendix A.4. The audit was used only for quality control and disagreement resolution, and all annotations were performed over public tabular evidence.

Because the annotation task involved only verification over public tabular data and did not involve private personal data, intervention, or high-risk human-subject protocols, we did not seek separate ethics review board approval.

A.7 Use of AI Assistants

We used GPT-4o during dataset construction to generate candidate fact-checking claims from related tables. The model was used only as a data generation aid: all generated claim–label pairs were subsequently manually verified and corrected by human annotators before inclusion in the dataset. We did not rely on AI-generated outputs without human verification.

A.8 Licenses, terms of use, and intended use.

We use publicly available datasets, table corpora, pretrained models, software tools, and commercial LLM APIs for research purposes, and cite their original sources. We follow the corresponding licenses or terms of use, and our use is consistent with their intended research use. The datasets constructed in this work, OTT-FV and Hybrid-FV, and our code will be released for research use, with license and usage terms included in the released repository. These artifacts are intended for research on table-based fact-checking and should not be used as fully reliable fact-checking systems in high-stakes real-world settings without human oversight.

B AMR-guided Tree Construction

B.1 AMR Parser

We use `transition-amr-parser` to obtain an AMR-style semantic graph for each claim. The parser output serves as the starting point for our semantic tree construction.

B.2 Tree Normalization

Our semantic tree should be understood as a structured decomposition of the sentence rather than a direct one-to-one copy of the AMR graph. In practice, we extract entity nodes and attribute/relation structures from the AMR graph and normalize them into a tree suitable for bottom-up composition.

At a high level, the normalization process groups words into the same subtree based on their distances in the AMR graph and in the generated subtrees associated with entities. Intuitively, words that are semantically closer in the AMR structure

year	person	event
1993	Nelson Mandela	Won Nobel Peace Prize with F. W. de Klerk
1994	Nelson Mandela	Inaugurated as President of South Africa
1998	Nelson Mandela	Married Graça Machel
1999	Nelson Mandela	Stepped down after one term as President

Table 12: Nelson Mandela notable milestones.

Dataset	Sample Size	Raw Agreement	Cohen's κ
OTT-FV	200	97.5%	0.950
Hybrid-FV	200	93.0%	0.860
Overall	400	95.25%	0.905

Table 13: **Annotation quality audit.** We report human-human agreement between the original human-verified labels and an independent second annotation.

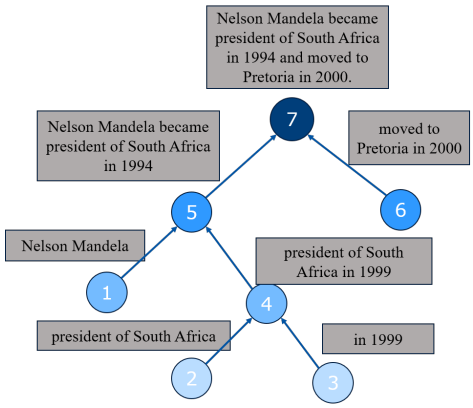

Figure 3: Illustrative example of AMR-guided semantic tree construction.

are more likely to be assigned to the same branch of the tree. This produces a hierarchical decomposition in which lower-level nodes correspond to smaller semantic units and higher-level nodes compose them into larger meaning-bearing subclaims.

Figure 3 shows an illustrative example. For the claim *"Nelson Mandela became president of South Africa in 1994 and moved to Pretoria in 2000."*, the tree first composes *Nelson Mandela* with *president of South Africa*, and then with *in 1994* to form the left branch. In parallel, it combines *moved to Pretoria* with *in 2000* to form the right branch. The root node finally merges these two branches into the full claim.

C Retrieval and Evidence Formatting

C.1 Retrieval Granularity

In the Open setting, we use a two-stage node-level retrieval process. For each semantic node, we first retrieve the top-k candidate tables by scoring the node query against serialized table representations. We then perform row-level reranking within the retrieved tables and retain the top-ranked rows as the final node-level evidence. Thus, table retrieval serves as the first-stage recall step, while row-level reranking provides finer-grained evidence selection for downstream composition and verification.

C.2 Evidence Serialization for Node Encoding

For node encoding, we serialize the node query together with the retrieved evidence rows into a single text sequence. Concretely, given a node query q_v and a ranked list of retrieved rows $\mathcal{R}_v = \{r_1, r_2, \ldots, r_n\}$, we construct the input as:

```
[Query]: <node query>
[Evidence]
[Row1]  <header1>:  <cell1>
<header2>: <cell2> ...
[Row2]  <header1>:  <cell1>
<header2>: <cell2> ...
⋮
[Rown]  <header1>:  <cell1>
<header2>: <cell2> ...
```

That is, each evidence row is linearized as a sequence of `header: cell` pairs, preserving the original column names so that the encoder can access both cell values and their semantic roles. Rows are concatenated in retrieval order, with special markers (`[Query]`, `[Evidence]`, `[Row`i`]`) used to distinguish the query from different evidence rows.

For example, if the node query is *"Nelson Mandela presidency in 1994"*, one retrieved row may be serialized as:

```
[Row1] Year: 1994 President:
Nelson Mandela Country: South
Africa
```

This serialized sequence is then fed into the node encoder to obtain the evidence-aware representation of the node.

C.3 Task-specific Retriever

Our task-specific retriever is based on Sentence-BERT, which we fine-tune to better match short node queries with serialized tables. Positive examples are constructed from a node query paired

Dataset	Original label	Second: SUPPORTED	Second: REFUTED
OTT-FV	SUPPORTED	96	4
	REFUTED	1	99
Hybrid-FV	SUPPORTED	90	10
	REFUTED	4	96

Table 14: **Confusion matrices for the annotation audit.** Rows correspond to the original human-verified labels, and columns correspond to labels assigned by the second annotator.

with its corresponding gold table(s), while negative examples are constructed by pairing the query with other tables from the same table series as the gold table. At inference time, we use the simplest dense retrieval setup: cosine similarity between the query embedding and serialized-table embedding over the full table pool.

The retriever is trained with the following settings:

- backbone: SentenceBERT
- batch size: 64
- learning rate: 1×10^{-5}
- number of epochs: 3
- warmup ratio: 0.05
- mixed precision: fp16
- max input length: 2048

D Implementation Details

D.1 Training Hyperparameters

For **DeBERTa** and **PASTA**, we use:

- optimizer: AdamW
- learning rate: 2×10^{-5}
- number of epochs: 3
- batch size: 16
- seed: 42
- max input length: 512

For **Qwen** and **Table-llama**, we use:

- optimizer: AdamW
- learning rate: 2×10^{-5}
- number of epochs: 3
- batch size: 12
- seed: 42
- max input length: 2048

D.2 Complexity

For a tree with $|N|$ nodes, TRACE makes one retrieval call and one node encoding per node, giving an overall complexity of $O(|N|)$ retrieval and encoding steps. The bottom-up composition is also linear in the tree size, i.e., $O(|N|)$ for a tree. In practice, the semantic trees are small and the evidence per node is capped at Top-k, which keeps the overall computation manageable in open-evidence settings.

D.3 Hardware

All experiments are conducted on the university's NVIDIA A100 GPU cluster.

E Baseline Details

Local backbones. DeBERTa-v3 and Qwen3-8B are deployed locally.

Official reproductions. PASTA, Table-llama, DCUF, and Factcheck-GPT are reproduced from their official repositories.

API-based LLMs. GPT-5.4 and Gemini-2.5-pro are evaluated via their official APIs. We use zero-shot prompting and otherwise keep the default API settings.

E.1 Baseline Selection

We choose baselines to cover the main comparison groups relevant to multi-table fact-checking: general verifier backbones, table-oriented verification models, LLM-based fact-checking systems, and industrial LLMs. This allows us to evaluate whether TRACE improves over both general-purpose verifiers and methods with stronger table or fact-checking capabilities.

We also discuss several related decomposition-based fact-checking systems, including ProgramFC (Pan et al., 2023b) and ClaimDecomp (Chen et al., 2022). These methods are important related work because they decompose complex claims into programs, subquestions, or intermediate reasoning units. However, they are primarily designed for textual evidence, web search, or tool-based verification. Directly adapting them to our setting would require additional design choices for table retrieval, row selection, and aggregation of table evidence into a final verdict. Since these components are not part of their original formulations,

such adaptations would introduce implementation-specific factors that make direct comparison less controlled.

Similarly, FEVEROUS (Aly et al., 2021) is highly relevant as a benchmark for verification over heterogeneous Wikipedia evidence, including both textual sentences and table cells. Our work instead isolates a table-only multi-table setting where evidence is distributed across multiple tables and must be composed through structured reasoning. Evaluating on the full FEVEROUS benchmark would require extending TRACE to mixed text-table evidence and FEVEROUS-specific evidence-verdict evaluation, which we leave to future work.

E.2 Computational Cost Analysis

We analyze the computational cost introduced by TRACE under the Open setting. Since wall-clock inference time and GPU-hour measurements can vary substantially with hardware, batching, caching, and implementation details, we report device-independent cost proxies: the average number of tree nodes, retrieval queries, retrieved rows, and verifier input tokens per claim. Flat backbone baselines use a single retrieval query per claim, while TRACE performs retrieval for each semantic tree node.

Table 15 shows that TRACE introduces additional computation compared with flat baselines. With AMR-based trees, each claim contains 7.3 nodes on average, leading to 7.3 retrieval queries and 59.6 retrieved rows per claim. This also increases the average verifier input length, from 358.7 to 2287.1 tokens for DeBERTa-v3 and from 452.5 to 3201.8 tokens for Qwen3-8B.

This additional cost reflects the intended trade-off of TRACE. Rather than retrieving a small flat evidence set with a single global query, TRACE retrieves locally relevant evidence for each semantic node and composes these evidence signals bottom-up. The increased computation is accompanied by clear improvements in Open-setting performance: for example, DeBERTa-v3 improves from 61.7 to 71.2 Macro-F1 on OTT-FV and from 53.1 to 59.1 on Hybrid-FV. These results make explicit the accuracy-efficiency trade-off introduced by structured retrieval and composition.

F Case Study: Multi-table Retrieval and Compositional Reasoning

We present a real example to illustrate how tree-structured retrieval helps TRACE recover evidence distributed across multiple tables, while a flat retriever is more easily distracted by only part of the claim. Tables 7–12 show a set of potentially retrievable related tables.

Claim. *"Nelson Mandela became president of South Africa in 1994 and moved to Pretoria in 2000."*

Gold label. REFUTED

Why this example is challenging. This claim contains two subclaims: (i) Nelson Mandela became president of South Africa in 1994, and (ii) he moved to Pretoria in 2000. The first subclaim is correct, while the second is false. As shown in Figure 3, the semantic tree decomposes the claim into these two parts, enabling the system to retrieve and verify them separately before composing a final decision at the root.

Relevant evidence. The first subclaim is supported by multiple tables. In `south_africa_presidents_by_year`, the entry for 1994 lists Nelson Mandela as president, with a note indicating that he took office on 1994-05-10. In `mandela_roles_by_year`, the entry for 1994 states that Nelson Mandela's main role was *President of South Africa*, again confirming the first half of the claim.

The second subclaim requires more careful reasoning. A flat retriever is easily attracted by tables mentioning Pretoria, such as `south_africa_capitals` and `south_africa_government_sites`, because Pretoria is strongly associated with South African government and presidential offices. However, these tables do not establish that Mandela *moved to Pretoria in 2000*. Instead, the relevant evidence comes from `mandela_residence_by_period`, which shows that Mandela had an official presidential residence in Pretoria during 1994–1999, but from retirement in 1999 (including 2000), his post-presidential home/office base was in *Houghton, Johannesburg*. Therefore, the claim that he *moved to Pretoria in 2000* is contradicted by the evidence.

Why tree-structured retrieval helps. The semantic tree separates the claim into a "presidency"

Method	Avg. #Nodes	Avg. #Queries	Avg. Rows	Avg. Tokens	OTT-FV	Hybrid-FV
DeBERTa-v3	1.0	1.0	10.0	358.7	62.5 / 61.7	53.7 / 53.1
TRACE (DeBERTa-v3)	7.3	7.3	59.6	2287.1	72.1 / 71.2	59.2 / 59.1
Qwen3-8B	1.0	1.0	10.0	452.5	64.6 / 63.8	53.3 / 54.2
TRACE (Qwen3-8B)	7.3	7.3	59.6	3201.8	72.4 / 71.6	59.7 / 58.9

Table 15: **Computational cost analysis under the Open setting.** We report device-independent cost proxies and verification results as Accuracy / Macro-F1. Avg. Tokens denotes the average verifier input tokens per claim.

branch and a "moved to Pretoria in 2000" branch. This decomposition allows node-level retrieval to focus on different evidence needs: one branch retrieves tables about Mandela's role in 1994, while the other retrieves tables about residence/location around 2000. In contrast, a flat root-level retriever tends to prioritize the more salient first half of the claim—*Mandela*, *president*, and *1994*—and may only retrieve partially relevant Pretoria-related tables for the second half. As a result, plain retrieval often captures only the supported portion of the claim, while missing the crucial contradictory evidence needed for the full verification decision.

Bottom-up reasoning. After node-level retrieval, TRACE verifies the two branches separately. The left branch establishes that Mandela indeed became president in 1994. The right branch, however, finds that although Pretoria is associated with presidential residence during Mandela's term, the evidence for the period after his retirement places him in Houghton, Johannesburg rather than Pretoria in 2000. When these two branch-level results are composed at the root, the final verdict is REFUTED.

Takeaway. This example highlights the main advantage of TRACE: by decomposing a claim into semantically coherent subclaims, the framework can retrieve evidence for each part separately and avoid being dominated by the most salient supported fragment of the claim. In multi-table fact-checking, this tree-structured retrieval-and-composition process is especially important when one part of the claim is correct but another part is contradicted by less obvious evidence.