Logic-level Evidence Retrieval and Graph-based Verification Network for Table-based Fact Verification

Qi Shi, Yu Zhang*, Qingyu Yin, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, Harbin, China
{qshi,zhangyu,qyyin,tliu}@ir.hit.edu.cn

Abstract

Table-based fact verification task aims to verify whether the given statement is supported by the given semi-structured table. Symbolic reasoning with logical operations plays a crucial role in this task. Existing methods leverage programs that contain rich logical information to enhance the verification process. However, due to the lack of fully supervised signals in the program generation process, spurious programs can be derived and employed, which leads to the inability of the model to catch helpful logical operations. To address the aforementioned problems, in this work, we formulate the table-based fact verification task as an evidence retrieval and reasoning framework, proposing the **L**ogic-level **E**vidence **R**etrieval and **G**raph-based **V**erification network (LERGV). Specifically, we first retrieve logic-level program-like evidence from the given table and statement as supplementary evidence for the table. After that, we construct a logic-level graph to capture the logical relations between entities and functions in the retrieved evidence, and design a graph-based verification network to perform logic-level graph-based reasoning based on the constructed graph to classify the final entailment relation. Experimental results on the large-scale benchmark TABFACT show the effectiveness of the proposed approach[1].

1 Introduction

There are a large number of semi-structured tables on the Internet. How to perform reasoning over semi-structured tables is crucial for people to understand different types of information in the real world. And this direction has spawned many tasks. Among these tasks, table-based fact verification task has recently received a lot of attention, which is important for many applications, such as fake news detection, scientific paper understanding (Wang et al., 2021), etc. This task aims to verify the correctness of the given statement by the given table, which requires both linguistic reasoning and symbolic reasoning (Chen et al., 2020b).

Table

Chassis Manufacturer	Chassis Model	Number in Fleet	Fleet Numbers
man	hocl-nl	20	2101 - 2120
mercedes - benz	o405nh	2	2520 - 2521
mitsubishi	fuso rosa	6	34, 2601 - 2603
scania	k280ub	1	3230
scania	k320ua	6	2831 - 2836

Statement the smallest number in fleet for chassis manufacturer , scania , with a fleet number 3230 is 1

Label ENTAILED

Program-like Evidence

- eq { 1 ; min { all_rows ; number in fleet } } = True
- eq { 1 ; hop { filter_eq { all_rows ; fleet numbers ; 3230 } ; number in fleet } } = True
- eq { scania ; hop { filter_eq { all_rows ; number in fleet ; 1 } ; chassis manufacturer } } = True
- …

Figure 1: Example of TABFACT dataset. Given a table and a statement, the goal is to verify the correctness of the statement by the table. Program-like evidence contains logical operations and can be used as supplementary information to a table.

Symbolic reasoning with logical operations like *"count"* and *"argmax"* plays an important role in the table-based fact verification task. Figure 1 shows an example. Ideally, to verify the correctness of such statements, logical operations provide strong hints to classify the entailment relation. Therefore, how to utilize such logical operations is crucial in this task.

Program is a kind of logic form derived from tables, which contains rich logical operations. Following (Chen et al., 2020b), existing methods (Zhong et al., 2020a; Yang et al., 2020; Shi et al., 2020) mostly use programs to perform symbolic reasoning. Specifically, they derive one or sev-

*Corresponding author.
[1]Our code is available at: https://github.com/qshi95/LERGV

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 175–184
November 7–11, 2021.

eral programs with a semantic parser based on the given table and statement, and then leverage the obtained programs to enhance the verification process. However, these models may select spurious programs (i.e. wrong programs with correct returned labels) because there are weak supervised signals in the semantic parsing process. Consequently, label-consistent programs (the programs whose execution results are consistent with the ground-truth verification labels) tend to be selected instead of semantic-consistent ones. As a result, the obtained programs will not contain helpful logical operations, thus cannot benefit the verification of the correctness of the statement. Ideally, a natural way of leveraging programs is to regard them as supplementary evidence for tables. In other words, some of the programs contain the necessary information with logical operations that summarize or describe the facts observed from the table. So that we can leverage the information conveyed by the programs to help better capture the facts of the table and then classify the entailment relation.

Based on the above considerations, in this work, we formulate table-based fact verification as an evidence retrieval and reasoning pipeline, proposing a logic-level evidence retrieval and graph-based verification method, named LERGV. Firstly, instead of deriving label-consistent programs with a semantic parser, we propose a rule-based method to retrieve valuable logic-level program-like evidence from the table and statement to avoid the issue of spurious programs. Then we leverage the structure of the programs to construct a logic-level graph with the above evidence to catch the logical relations between entities (such as *"number in fleet"*) and functions (such as *"min"*) in the programs. Finally, a graph-based verification network is proposed to reason over the constructed graph to perform logic-level reasoning, which takes advantage of the combination of linguistic reasoning and symbolic reasoning to make the final prediction.

We conduct experiments on a large-scale benchmark dataset TABFACT (Chen et al., 2020b). Experimental results show that our model surpasses all baseline systems with a considerable margin. The main contributions of this paper are three-fold:

- We formulate the table-based fact verification task as an evidence retrieval and graph-based reasoning framework by regarding programs as additional evidence instead of building a weakly supervised semantic parser to avoid the issue of spurious programs.
- We construct a logic-level graph and propose a graph-based verification network to catch logical relations between entities and functions in evidence, which can take advantage of both linguistic reasoning and symbolic reasoning.
- Experimental results on the TABFACT show the effectiveness of our proposed approach that our method outperforms all baseline systems and achieves competitive results.

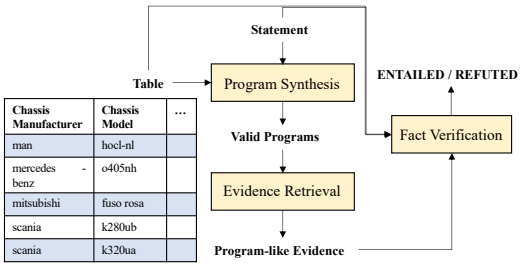

Figure 2: Pipeline for table-based fact verification task. Given a table and a statement, the proposed pipeline can be divided into program synthesis module, evidence retrieval module (§3.1), and fact verification module (§3.2 and §3.3).

2 Task Definition and Overview

In this paper, we study the task of table-based fact verification. Given a table T with R rows and C columns and a statement S, the goal is to verify the correctness of the given statement by the given table with the label ENTAILED or REFUTED.

Table is the only evidence in the original task setting. However, we believe that the additional evidence that contains logical operations is helpful to classify the entailment relation. In this study, we employ such evidence (denote as "evidence" in the rest of the paper) that in the form of programs. In particular, program is a kind of LISP-like logical form that follows a grammar with over 50 pre-defined functions (Chen et al., 2020b). Every program is tree-structured, consisting of functions as parent nodes and their arguments as children nodes, where leaf nodes represent arguments, namely entities linked to the table or the statement, and non-leaf nodes represent functions, such as *"min"*, *"count"*, *"argmax"*, etc. The dotted boxes in Figure 3 show the structure of programs.

To better take advantage of program-like evidence, in this work, we formulate the table-based fact verification task as an evidence retrieval and

reasoning pipeline that consists of three main components, program synthesis, evidence retrieval, and fact verification modules. Figure 2 shows the overview of our proposed approach. Given the table and the statement, the program synthesis module first synthesizes all possible programs with valid combinations. Then the evidence retrieval module selects, decomposes, and filters over the programs to obtain valuable logic-level evidence as supplementary information to the original table. Finally, a fact verification model takes the input of the table, statement, and obtained evidence to predict whether the statement is supported by the given table. Following previous work (Zhong et al., 2020a; Yang et al., 2020; Shi et al., 2020), we perform program synthesis with the latent program search algorithm (LPA) (Chen et al., 2020b), followed by an evidence retrieval module and a fact verification module. We will present above modules in § 3.

3 Methodology

We propose a **L**ogic-level **E**vidence **R**etrieval and **G**raph-based **V**erification network (LERGV) for the table-based fact verification task. Given a table and a statement, LERGV works as follows. First, we start with the latent program algorithm (Chen et al., 2020b) to synthesize programs and then use a rule-based retrieval approach to select, decompose, and filter among all synthesized programs to obtain valuable logic-level evidence (§ 3.1). After that, we construct a graph based on the retrieved evidence and initialize graph node representations from a pre-trained language model (§ 3.2). Finally, we propose a graph-based verification network centered around the obtained evidence to perform graph-based reasoning to predict the final verification result (§ 3.3).

3.1 Logic-level Evidence Retrieval

Program is a kind of logic form with rich logical operations. For the specific task of table-based fact verification, we believe that the program-like evidence can provide valuable information in addition to tables. Following (Zhong et al., 2020a; Shi et al., 2020; Yang et al., 2020), in this work, we follow the latent program search algorithm (LPA) (Chen et al., 2020b) to synthesize valid programs with pre-defined functions. Given a table T and a statement S, LPA first performs entity linking to detect all the entities in the statement and link them to the table, then collects a set of programs by executing sub-programs over the table and store the generated intermediate variables recursively.

After obtaining the program set $P = \{(P_i, A_i)\}_{i=1}^{N}$ for a given statement S (where P_i stands for the i-th program, and A_i refers to the corresponding returned label executing over the table, namely $True$ or $False$), instead of building a semantic parser, we select, decompose, and filter some of them by a series of rules to keep higher quality programs as evidence due to the limitation of input size of the pre-trained language model. Specifically, we retrieve the evidence in the following steps:

- We choose programs with the returned label $A_i = True$ in the program set P as evidence to ensure that the evidence can be correctly observed from the table.
- We decompose the evidence containing function *"and"* into two separate pieces of evidence, where each one is a subtree of the *"and"* node, to simplify and remove duplicate evidence. Since the label of the original evidence $A_i = True$, the two programs connected by the *"and"* node must both be $True$, thus we guarantee the correctness of the obtained evidence.
- For the cases that obtained a large number of programs, we remove evidence that contains functions with negative meanings, including *"not_eq"*, *"filter_not_eq"*, *"not_within"*, etc. This is due to that programs with negative functions tend to be descriptions of the statements that don't explicitly exist in tables (e.g. "Number of teams is not 3"), which are less effective than programs with positive functions. This operation can limit the number of evidence and obtain semantically more relevant evidence.

So far, by aggregating, decomposing, and filtering the information contained in the table, we can use the logic-level evidence to supplement the original table, thereby enhancing the ability of our model to understand semi-structured tables. Just like the motivating example shown in the Figure 1, with the evidence containing *"min"*, *"number in fleet"* and *"1"*, model can easily establish connections between *"smallest"*, *"number in fleet"* and *"1"*, which benefits to the verification of the correctness of the statement. After obtaining the evidence set

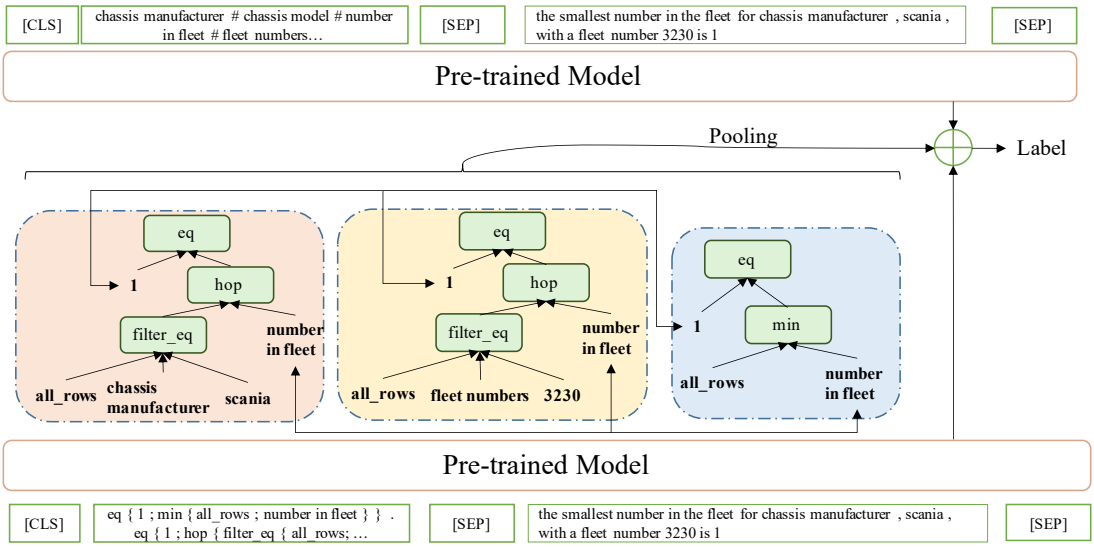

Figure 3: Model structure of LERGV. Logic-level program-like evidence is obtained from the table and statement described in §3.1. Take the table-statement pair and evidence-statement pair as input, we first construct a logic-level graph to catch the logical relations between entities and functions in the programs (§3.2). After that, we design a graph-based verification network to perform reasoning over the constructed graph before making the final verification (§3.3).

E, we use program-like evidence directly instead of converting programs into text (Yang et al., 2020) to preserve the logical connections between entities and functions in the program.

3.2 Graph Construction and Initialization

Given the retrieved evidence, we construct a graph to capture the logical relations among all entities and functions in the evidence programs. Specifically, we denote a graph as $G = \{V, E\}$ and treat each function (such as *"filter_eq"*, *"hop"*) and entity (such as *"all_rows"*, *"chassis manufacturer"*) as a graph node. Besides, to distinguish between nodes of different origins, we further divide the nodes into two types, namely function nodes and entity nodes (shown as nodes with/without frame in Figure 3).

We leverage the structure of evidence to construct edges. In this way, we can explore the connections between entity nodes and function nodes, which will benefit the following verification process. Specifically, we retain the structure of every program by adding edges between each entity node and its corresponding function node to learn the semantic compositionality of the program. Besides, we add edges between entity nodes with the same content across the entire evidence set E to turn the graph into a connected graph to reason over multiple evidence programs, which are shown as arrows in Figure 3.

We feed the table-statement pair and evidence-statement pair into a pre-trained language model separately as shown in Figure 3 and initialize node representations from the top-layer output of the pre-trained language model. For the node with multiple word pieces, we perform average pooling over the representations of their corresponding positions.

3.3 Graph-based Verification Network

We propose a graph-based verification network, which is designed to perform logic-level reasoning over the retrieved evidence along with the given table and statement to combine linguistic reasoning and symbolic reasoning to benefit the final verification decision.

Graph-based Reasoning Process After the graph construction and node initialization, LERGV performs reasoning over the constructed graph. Our network is designed based on the graph attention network (Velickovic et al., 2018), to learn the importance between different nodes and fuse the neighbors to perform graph-based reasoning. Specifically, the node representations are updated as follows:

$$e_{i,j} = \text{LeakyReLU}(a[W_q h_i || W_k h_j]) \quad (1)$$

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k \in N_i} \exp(e_{i,k})} \quad (2)$$

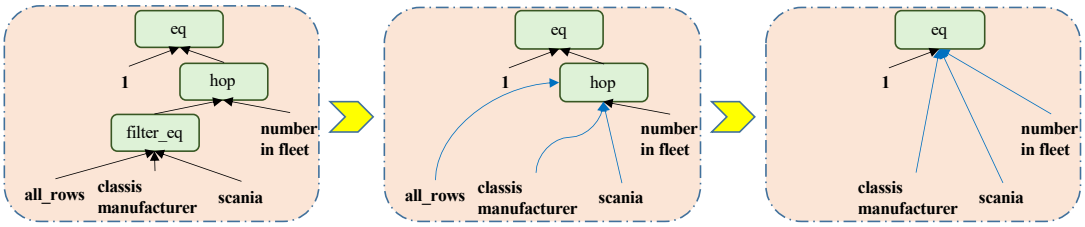

Figure 4: Example of the node pruning process. We take one single program as an example. We remove *"filter_eq"* in Figure (a) and add edges between its children nodes and parent node. After that, we remove nodes *"all_rows"* and *"hop"* in the Figure (b), which is the same as Figure (a).

$$h_i = \sigma(\sum_{j \in N_i} \alpha_{i,j} \mathrm{W_v} h_j) + h_i \quad (3)$$

where $W_q \in R^{F \times F}$, $W_k \in R^{F \times F}$, $W_v \in R^{F \times F}$, $a : R^F \times R^F \rightarrow R$ are trainable parameters. N_i presents the neighbors of node i. $||$ stands for concatenation operation. h_i and h_j mean the representations of node i and node j. We will describe them in details in the following parts.

Node Type Representations Notice that the nodes are either entities linked to the table or the statement, or functions pre-defined in LPA, to this end, we model different types of nodes in the message passing process. First, we combine type embedding for every node in the graph as follows:

$$h_i^t = W_t(o_i) + b_t \quad (4)$$

$$h_i = [h_i^p || h^s || h_i^t] \quad (5)$$

where $o_i \in R^T$ is a one-hot vector indicating the type of the node i. $W_t \in R^T \times R^F$, $b_t \in R^F$ are trainable parameters. T presents the number of node types, in this work, T is set to 2. h_i^p means node representation obtained by the node initialization process. And h^s means the representation of the statement, which is obtained similarly as h_i^p.

Node Pruning Programs are designed for executing over the table T recursively. However, as an evidence program, many nodes are semantically unrelated to the statement, such as *"filter_eq"*, *"hop"*, *"all_rows"*, etc. Here we propose a node pruning approach to prune the nodes automatically to filter such nodes. Figure 4 shows an example. We first calculate the relevance score between the node and statement as follows:

$$s_i = \sigma(W_s[h_i^p || h^s] + b_s) \quad (6)$$

where $W_s \in R^{2F \times F}$, $b_s \in R^F$ are trainable parameters. Then, we remove the nodes with the lowest scores with the probability θ. Finally, take one removed node as an example, we add edges between its children nodes and parent node to connect the new graph to perform graph-based verification.

Label Prediction We adopt an attentive pooling layer to obtain the final representation h. Then, we concatenate h and two [CLS] tokens, that come from the output of the pre-trained language model with table-statement pair and evidence-statement pair as input respectively. Finally, we feed the obtained vector into a classifier to predict the probability of each label. The concatenation operation aims to combine linguistic information containing in the table-statement pair and logic-level information containing in the program-like evidence together to achieve the goal to combine linguistic reasoning and symbolic reasoning.

4 Experiments

4.1 Dataset and Experimental Settings

We evaluate our model on the TABFACT (Chen et al., 2020b), a large-scale dataset for table-based fact verification, which contains 92283, 12792, and 12779 samples in training, validation, and test sets respectively, with one table and one statement in each sample. Each sample is labeled as either ENTAILED or REFUTED, indicates whether the statement is supported by the table. The test set is further divided into the simple channel and complex channel to distinguish the difficulty, with 4171 and 8608 samples for each. Besides, a small test set with $2K$ samples is provided for human evaluation. Samples in the dataset require symbolic reasoning with logical operations, such as *"min"*, *"argmax"*, *"count"*, etc. Following the existing work, we use accuracy as the evaluation metric.

Following (Chen et al., 2020b), we use BERT-base (Devlin et al., 2019) as the backbone to build our model. The maximum sequence length is 512,

Model	Val	Test	Test (simple)	Test (complex)	Small Test
BERT classifier w/o Table	50.9	50.5	51.0	50.1	50.4
Table-BERT-Horizontal-F+T-Concatenate	50.7	50.4	50.8	50.0	50.3
Table-BERT-Vertical-F+T-Template	56.7	56.2	59.8	55.0	56.2
Table-BERT-Vertical-T+F-Template	56.7	57.0	60.6	54.3	55.5
Table-BERT-Horizontal-F+T-Template	66.0	65.1	79.0	58.1	67.9
Table-BERT-Horitonzal-T+F-Template	66.1	65.1	79.1	58.2	68.1
LPA-Voting w/o Discriminator	57.7	58.2	68.5	53.2	61.5
LPA-Weighted-Voting	62.5	63.1	74.6	57.3	66.8
LPA-Ranking w/ Transformer	65.2	65.0	78.4	58.5	68.6
LogicalFactChecker	71.8	71.7	85.4	65.1	74.3
HeterTFV	72.5	72.3	85.9	65.7	74.2
SAT	73.3	73.2	85.5	67.2	-
ProgVGAT	74.9	74.4	**88.3**	67.6	76.2
LERGV	**75.6**	**75.5**	87.9	**69.5**	**77.8**
Human Performance	-	-	-	-	92.1

Table 1: Experimental results on TABFACT. For Table-BERT, *T* and *F* refer to table and statement respectively. *Horizontal* and *Vertical* represent the scanning strategies of linearizing tables. *Concatenate* and *Template* stand for whether use templates to concatenate the table cells. For LPA, *Voting* and *Weighted-Voting* mean voting for the result without/with a weighted-sum score. *Ranking* means using the result of the top-ranked program.

the batch size is set to 8, the learning rate is set to $1e-5$, the warmup step is set to 3000, and the probability of pruning nodes θ is set to 0.3. We set the size of all hidden layers to 768, which is the same as the output of the BERT-base model. Cross entropy loss is adopted to optimize the model.

4.2 Baseline Systems

Latent Program Algorithm (LPA) LPA (Chen et al., 2020b) treats the task in a weakly supervised manner by feature-based entity linking, latent program generation, and candidate program ranking with a Transformer-based encoder (Vaswani et al., 2017).

Table-BERT Table-BERT (Chen et al., 2020b) views the task as a semantic matching problem by encoding linearized table and the statement via BERT to predict the final label.

LogicalFactChecker LogicalFactChecker (Zhong et al., 2020a) derives one program with different semantic parsers and represents it with graph module networks to learn the semantic compositionality of the program.

HeterTFV HeterTFV (Shi et al., 2020) chooses multiple latent programs and proposes a heterogeneous graph-based reasoning network to reason over different types of information.

SAT SAT (Zhang et al., 2020) proposes a structure-aware table representation method by utilizing the mask in self-attention layers.

ProgVGAT ProgVGAT (Yang et al., 2020) improves the semantic parser with a specific loss function and converts the obtained program to natural language sentences with pre-defined templates to perform graph-based reasoning.

4.3 Experimental Results

Table 1 shows the experiment results: our model reaches an accuracy of 75.5% on the test set, which surpasses all baseline systems with remarkable improvements. [2]

From Table 1, we can observe that our model outperforms LPA (Chen et al., 2020b), Table-BERT (Chen et al., 2020b), and SAT (Zhang et al., 2020) with large margins, which illustrates the advantage of the combination of linguistic reasoning and symbolic reasoning. Meanwhile, compared with the semantic parsing-based methods, i.e., LogicalFactChecker (Zhong et al., 2020a), HeterTFV (Shi et al., 2020), and ProgVGAT (Yang et al., 2020), our model gains improvements in performance from 1.1% to 3.8%, which indicates the

[2]We do not compare with the TAPAS-based approach directly (Eisenschlos et al., 2020). Because our approach is centered around evidence rather than table. The pre-trained model with evidence-statement pair as input is not adapted to the TAPAS model.

effectiveness of our proposed model that our model can better understand the semi-structured tables by retrieving logic-level evidence as supplementary information and catching logical relations between entities and functions. In addition, we can also see that our model surpasses ProgVGAT (Yang et al., 2020) by nearly 2 points on the complex test set, which further shows the ability of the proposed graph-based verification network on dealing with complex statements. In sum, all these results demonstrate the utility of the proposed method for fact verification over semi-structured tables.

Notice that there is a narrow gap between our model and ProgVGAT (Yang et al., 2020) on simple test set, which is due to that our method retrieves related programs with rich logical operations, which naturally brings more benefits to complex statements. In comparison, ProgVGAT (Yang et al., 2020) focuses on building a better semantic parser and only choose the most suitable program, so that works better on simple statements. Moreover, our method also gains competitive performance on the simple testset, with only a 0.4% gap compared with ProgVGAT (Yang et al., 2020).

4.4 Ablation Study

We report how each component contributes to LERGV by eliminating each one from the entire model on the validation set. Table 2 shows the results. Specifically, the removals of the node pruning module, node type representation module lead to a drop by $0.4\% - 1.0\%$ on the validation set, which indicates the effectiveness of each component in our proposed graph-based verification network. We then replace the entire graph-based verification module with the concatenation of model inputs and graph node representations only to predict the verification result. This operation causes a 1.5% drop on the validation set, which further shows the effectiveness of our constructed graph: i.e., catching logical relations between entities and functions plays a big part in our model.

Besides, eliminating the evidence retrieval module leads to significant drops in performance, with 4.4% on the validation set. This result demonstrates that logic-level evidence, as supplementary information to the original table, plays a crucial role in our proposed approach, which can help our model better understand semi-structured tables. In addition, by removing this module, our model will be left with just the table and the statement to perform linguistic reasoning only, which also proves the importance of combining linguistic reasoning and symbolic reasoning.

Model	Val
LERGV	75.6
- Node Pruning	75.2
- Node Type Representation	74.6
- Graph-based Verification	74.1
- Evidence Retrieval	71.2

Table 2: Ablation study of the model components.

4.5 Case Study

We provide an example to show the quality of our proposed approach, which is shown in Figure 5. The key point to verify the correctness of such a statement is to obtain evidence about *"only"* and *"more than"*. In our retrieved evidence, the first evidence program contains the function *"only"*, and can establish the logic-level connections between *"26 January 2011"* and *"only"* by graph construction process. Besides, three evidence programs within the function *"filter_greater"* indicate that the score of the game with the date *"26 January 2011"* and the venue *"sai tso wan recreation ground, hong kong"* is greater than 0. Other evidence without the above functions describes the information in the table that is relevant to the statement, which is also helpful to perform verification.

4.6 Error Analysis

We randomly sample 400 examples and categorize the errors into three classes.

In our analysis, the first category of error is the lack of evidence, or evidence is not helpful enough to verify the correctness of the given statement. This may be caused by the following two reasons. Firstly, entities in the statements are not detected and linked correctly to the table cells in the entity linking phase. Secondly, trigger words are applied to shrink the search space in the program synthesis process, which may cause some valuable information to be discarded. For example, the statement states *"ngc 1796 has the largest apparent magnitude of 12.9 followed by ngc 1705 with 12.8"*, the evidence with the function *"second"* is not obtained, which causes the model to fail to get the correct prediction. 363 error examples are caused by this category.

The second category of error appears when the

Table

date	venue	result	scored
20 June 2010	estadio campo desportivo, macau	5 - 1	0
2 November 2010	siu sai wan sports ground, hong kong	0 - 4	0
26 January 2011	sai tso wan recreation ground, hong kong	1 - 0	1
9 February 2011	po kong village park, hong kong	1 - 4	0
3 June 2010	xianghe sports center, beijing	2 - 2	0

Statement only the 26 january 2011 game in sai tso wan recreation ground , hong kong scored more than 0

Label ENTAILED

Program-like Evidence

- only { filter_eq { all_rows ; date ; 26 january 2011 } } = True
- eq { sai tso wan recreation ground , hong kong ; hop { filter_greater { all_rows ; scored ; 0 } ; venue } } = True
- eq { sai tso wan recreation ground , hong kong ; hop { filter_greater { all_rows ; scored ; 0 } ; venue } } = True
- less { 0 ; hop { filter_eq { all_rows ; venue ; sai tso wan recreation ground , hong kong } ; scored } } = True
- eq { sai tso wan recreation ground , hong kong ; hop { filter_greater { filter_eq { all_rows ; date ; 26 january 2011 } ; scored ; 0 } ; venue } } = True
- eq { 26 january 2011 ; hop { filter_eq { filter_greater { all_rows ; scored ; 0 } ; venue ; sai tso wan recreation ground , hong kong } ; date } } = True
- …

Figure 5: Case study of our proposed approach.

statement requires numerical reasoning. For example, for the statement *"jelle van damme scored three times as much as each of the other two players in the uefa champions league tournament"*, the proposed evidence retrieval approach cannot deal with the operation *"three times"*. 32 error examples are caused by this category, and we leave the numerical reasoning for future work.

The third category of error is caused by the program denotation. In other words, the order of the arguments in the program will influence semantic understanding. For example, the statement states *"A is larger than B.* The retrieved evidence is *"less { B; A}"*, which tends to be predicted as "REFUTED" due to the gap between *"larger than"* and *"less"*, although two expressions have the same meanings. 5 error examples are caused by this category.

5 Related Work

5.1 Fact Verification

Fact verification aims to verify the correctness of the input claim by the given evidence. Most of the existing methods on fact verification focus on dealing with the unstructured text as evidence. FEVER (Thorne et al., 2018) is one of the most popular datasets in this direction, which develops automatic fact verification systems to check the veracity of claims by extracting evidence from Wikipedia. After that, FEVER 2.0 share task (Thorne et al., 2019) is built, which is more challenging by the addition of an adversarial attack task. Recently, HOVER (Jiang et al., 2020) is proposed to focus on the many-hop evidence extraction and fact verification task. Previous work mainly follows the pipeline composed of document retrieval, evidence sentence selection, and claim verification. Most of the proposed models focus on the claim verification stage and graph-based reasoning approaches (Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020b) are popular in this stage.

Studies on fact verification over semi-structured evidence achieve much attention recently due to the proposal of the TABFACT dataset (Chen et al., 2020b). Two official baselines are provided along with this dataset named Table-BERT and LPA, which treat the task in a soft linguistic reasoning manner and hard symbolic reasoning manner respectively. Some approaches on this dataset focus on the representation of the semi-structured data (Zhang et al., 2020; Dong and Smith, 2021). And some approaches focus on the combination of linguistic reasoning and symbolic reasoning (Zhong et al., 2020a; Shi et al., 2020; Yang et al., 2020) mainly by building a semantic parser to select programs to serve the verification process. Different from their work, we propose an evidence retrieval module with a rule-based approach to obtain logic-level evidence as supplementary information for the original table to benefit the verification model.

5.2 Reasoning over Semi-Structured Data

Understanding semi-structured data is supposed to understand the structure and the content in every cell simultaneously. There are a lot of approaches in this direction spread over different tasks, such as question answering (Pasupat and Liang, 2015; Nan et al., 2021), natural language inference (Gupta et al., 2020; Neeraja et al., 2021), fact verification (Chen et al., 2020b), etc. And some methods put attention on the pre-training strategies on the semi-structured data along with the textual input (Herzig et al., 2020; Yin et al., 2020; Eisenschlos et al., 2020; Yu et al., 2020). Besides, a range of approaches reason on mixed evidence sources incorporating semi-structured data, such as reasoning over table and text together (Chen et al., 2020c,a) and multi-modal evidence including table, text and image (Talmor et al., 2021). Our work

treats the semi-structured table as a large evidence set and leverages the proposed evidence retrieval approach to aggregate information from tables to obtain logic-level evidence to perform verification.

6 Conclusion

In this study, we focus on the table-based fact verification task and propose LERGV that leverages logic-level program-like evidence to perform fact verification over tables. The main idea is that we formulate the task as an evidence retrieval and graph-based reasoning pipeline, and treat logic-level evidence as supplementary information for tables to avoid the issue of spurious programs. Specifically, we first apply a rule-based evidence retrieval approach to select, decompose, and filter among synthesized programs to obtain valuable logic-level program-like evidence. Then, we construct a graph based on the retrieved evidence to catch logical relations between entities and functions. Finally, we propose a graph-based verification network to perform reasoning over the constructed graph to combine linguistic reasoning and symbolic reasoning effectively. Experimental results on TABFACT illustrate that our model outperforms all the baseline systems and achieves competitive results.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by the Key Development Program of the Ministry of Science and Technology (No. 2019YFF0303003), the National Natural Science Foundation of China (No.61976068) and "Hundreds, Millions" Engineering Science and Technology Major Special Project of Heilongjiang Province (No.2020ZX14A02).

References

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. 2020a. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020c. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rui Dong and David A 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*, pages 2366–2375.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3441–3460, Online. Association for Computational Linguistics.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang,

et al. 2021. Fetaqa: Free-form table question answering. *arXiv preprint arXiv:2104.00369*.

J Neeraja, Vivek Gupta, and Vivek Srikumar. 2021. Incorporating external knowledge to enhance tabular reasoning. *arXiv preprint arXiv:2104.04243*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Qi Shi, Yu Zhang, Qingyu Yin, and Ting Liu. 2020. Learn to combine linguistic and symbolic information for table-based fact verification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5335–5346, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *arXiv preprint arXiv:2104.06039*.

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, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2019. The second fact extraction and verification (fever2. 0) shared task. *EMNLP 2019*, page 1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (semtab-facts). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.

Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. Program enhanced fact verification with verbalization and graph attention network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.

Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020a. LogicalFactChecker: Leveraging logical operations for fact checking with graph module network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020b. 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, Florence, Italy. Association for Computational Linguistics.