Learn to Combine Linguistic and Symbolic Information for Table-based Fact Verification

Qi Shi, Yu Zhang, Qingyu Yin, Ting Liu
Department of Computer Science and Technology,
Harbin Institute of Technology, Harbin, China
{qshi,zhangyu,qyyin,tliu}@ir.hit.edu.cn

Abstract

Table-based fact verification is expected to perform both linguistic reasoning and symbolic reasoning. Existing methods lack attention to take advantage of the combination of linguistic information and symbolic information. In this work, we propose *HeterTFV*, a graph-based reasoning approach, that learns to combine linguistic information and symbolic information effectively. We first construct a program graph to encode programs, a kind of LISP-like logical form, to learn the semantic compositionality of the programs. Then we construct a heterogeneous graph to incorporate both linguistic information and symbolic information by introducing program nodes into the heterogeneous graph. Finally, we propose a graph-based reasoning approach to reason over the multiple types of nodes to make an effective combination of both types of information. Experimental results on a large-scale benchmark dataset TABFACT illustrate the effect of our approach.

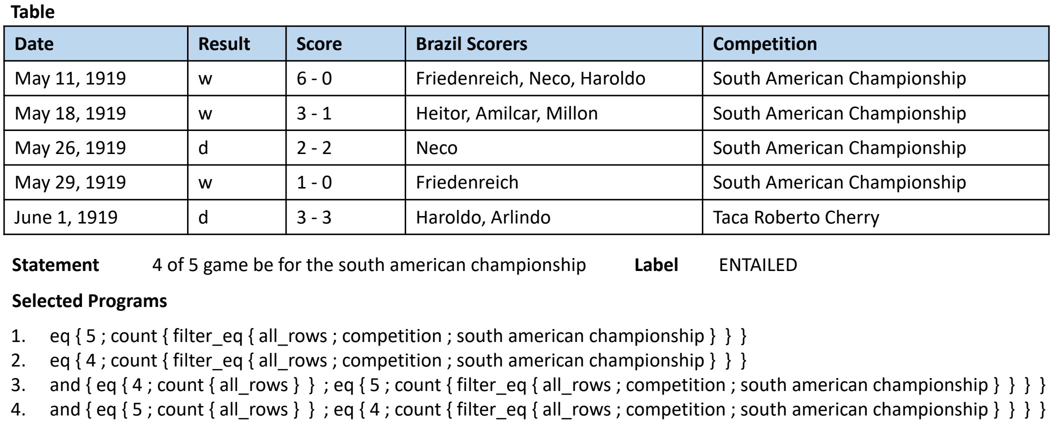
1 Introduction

Fact verification aims to verify whether a fact is entailed or refuted by the given evidence, which has attracted increasing attention. Recent researches mainly focus on the unstructured text as the evidence and ignoring the evidence with the structured or semi-structured format. A recently proposed dataset TABFACT (Wenhu Chen and Wang, 2020) fills this gap, which is designed to deal with the table-based fact verification problem, namely, verifying whether a statement is correct by the given semi-structured table evidence.

It is well accepted that symbolic information (such as *count* and *only*) plays a great role in understanding semi-structured evidence based statements (Wenhu Chen and Wang, 2020). However, most existing approaches for fact verification (Thorne et al., 2018; Nie et al., 2019; Zhou et al., 2019; Liu et al., 2020; Zhong et al., 2020b; Soleimani et al., 2020) focus on the understanding of natural language, namely, linguistic reasoning, but fail to consider symbolic information, which plays an important role in complex reasoning (Liang et al., 2017; Dua et al., 2019; Chen et al., 2019). Due to the diversity of natural language expressions, it is difficult to capture symbolic information effectively from natural language directly. Consequently, how to leverage symbolic information effectively becomes a crucial problem. To alleviate this problem, Zhong et al. (2020a) propose a graph module network that concatenates graph-enhanced linguistic-level representations and program-guided symbolic-level representations together to predict the labels. However, their method focuses on the representation of symbolic information, rather than take advantage of the combination of both types of information. More specifically, we believe that the concatenation operation between two types of representations is not effective enough to leverage the linguistic information and symbolic information to perform reasoning.

In recent studies, graph neural networks show their powerful ability in dealing with semi-structured data (Bogin et al., 2019a; Bogin et al., 2019b). Under this consideration, we propose to use graph neural networks that learn to combine linguistic information and symbolic information in a simultaneous fashion. Since the representations of different types of information fall in different embedding spaces,

Proceedings of the 28th International Conference on Computational Linguistics, pages 5335–5346
Barcelona, Spain (Online), December 8-13, 2020



Table

Date	Result	Score	Brazil Scorers	Competition
May 11, 1919	w	6 - 0	Friedenreich, Neco, Haroldo	South American Championship
May 18, 1919	w	3 - 1	Heitor, Amilcar, Millon	South American Championship
May 26, 1919	d	2 - 2	Neco	South American Championship
May 29, 1919	w	1 - 0	Friedenreich	South American Championship
June 1, 1919	d	3 - 3	Haroldo, Arlindo	Taca Roberto Cherry

Statement 4 of 5 game be for the south american championship **Label** ENTAILED

Selected Programs

1. eq { 5 ; count { filter_eq { all_rows ; competition ; south american championship } } }
2. eq { 4 ; count { filter_eq { all_rows ; competition ; south american championship } } }
3. and { eq { 4 ; count { all_rows } } ; eq { 5 ; count { filter_eq { all_rows ; competition ; south american championship } } } }
4. and { eq { 5 ; count { all_rows } } ; eq { 4 ; count { filter_eq { all_rows ; competition ; south american championship } } } }

Figure 1: Example of the TABFACT dataset, which are expected to combine both linguistic information in the statement and the table and symbolic information in the programs. Given a table and a statement, the goal is to predict whether the label is *ENTAILED* or *REFUTED*. Program is a kind of LISP-like logical form. The program synthesis and selection process are described in Section 3.2.

a heterogeneous graph structure is suitable to reason and aggregate over different types of nodes to combine different types of information.

In this paper, we propose a heterogeneous graph-based neural network for table-based fact verification named *HeterTFV*, to learn to combine linguistic information and symbolic information. Given a statement and a table, we first generate programs with the latent program algorithm (LPA) algorithm proposed by Wenhu Chen and Wang (2020). After that, we construct a program graph to capture the inner structure in the program and use gated graph neural network to encode the programs to learn the semantic compositionality. Then a heterogeneous graph is constructed with statement nodes, table nodes, and program nodes to incorporate both linguistic information and symbolic information, which is expected to exploit the structure in the table and build connections among the statement, table, and programs. Finally, a graph-based neural network is proposed to reason over the constructed heterogeneous graph, which enables the message passing processes of different types of nodes to achieve the purpose to combine linguistic information and symbolic information.

We conduct experiments on the TABFACT (Wenhu Chen and Wang, 2020), a large-scale benchmark dataset for table-based fact verification. Experimental results show that our model outperforms all baselines and achieves state-of-the-art performance.

In summary, the main contributions of this paper are three-fold:

- We construct a heterogeneous graph by introducing program nodes, to incorporate both linguistic information and symbolic information.
- We propose a graph-based approach to reason over the constructed heterogeneous graph to perform different types of message passing processes, which makes an effective combination of linguistic information and symbolic information.
- Experimental results on the TABFACT dataset illustrate the advantage of our proposed heterogeneous graph-based approach: our model outperforms all the baseline systems and achieves a new state-of-the-art performance.

2 Task Definition

Given a statement and a table, our goal is to verify whether a textual statement is entailed or refuted by the given semi-structured table evidence. We evaluate our model on the TABFACT dataset (Wenhu Chen and Wang, 2020), a large-scale benchmark dataset for table-based fact verification. Figure 1 shows an

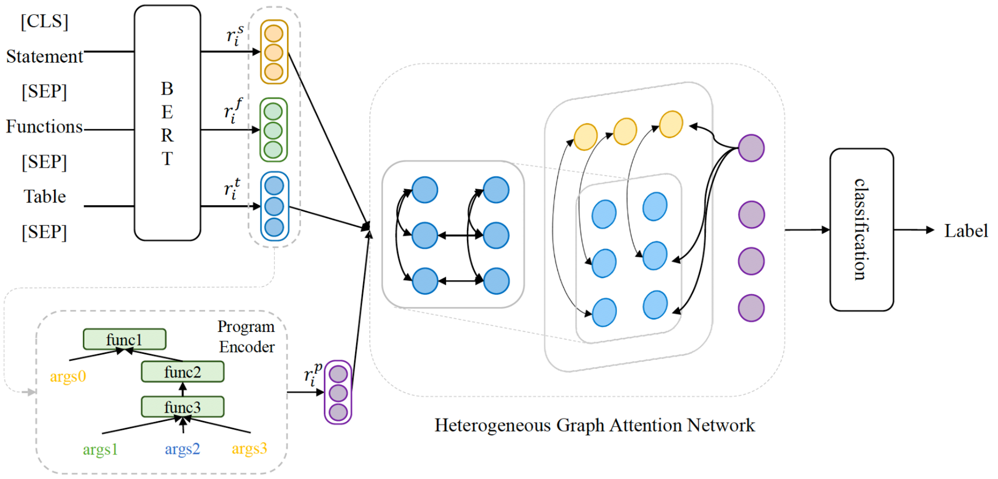


Figure 2: Architecture of the proposed graph-based approach. Taking the statement, function names (such as *eq*, *count*, *filter_eq*, and so on) and the table as input, our approach first constructs a program graph to encode programs, and then constructs a heterogeneous graph among different types of nodes. Finally, a graph-based reasoning network is applied on the constructed heterogeneous graph. Blue, yellow, and purple nodes stand for table nodes, statement nodes, and program nodes, respectively.

example in the TABFACT dataset. Specifically, in our work, as shown in Figure 1, we use programs to represent symbolic information, which is a kind of LISP-like logical form proposed by (Wenhu Chen and Wang, 2020). The program synthesis and selection process are described in Section 3.2. To verify such a statement in the example, our model is expected to combine the linguistic information between the statement and the table and the symbolic information in the programs, such as *count*, *and*, and so on.

3 HeterTFV

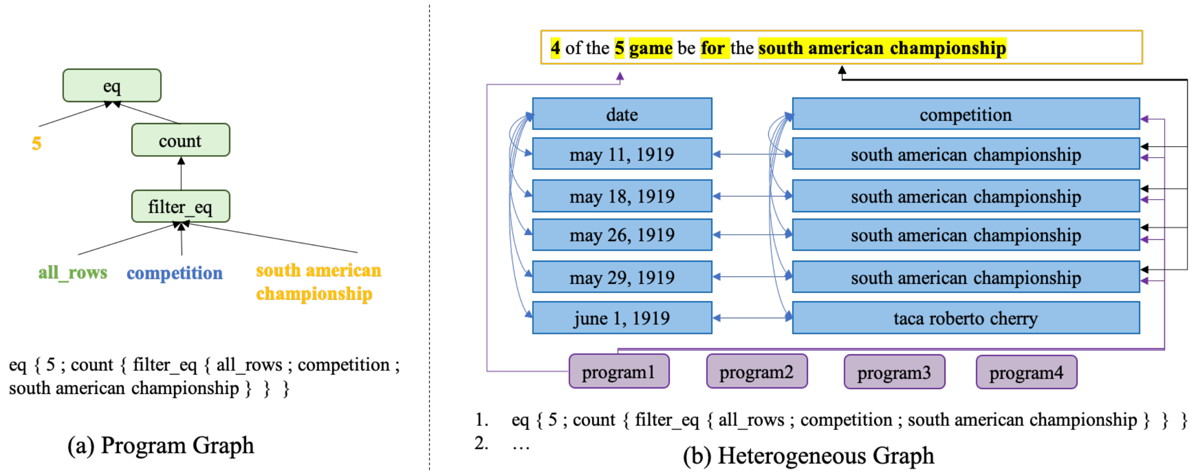
In this paper, we propose a graph-based approach for table-based fact verification task, which takes advantage of both linguistic information and symbolic information. We first generate programs from the statement and table, construct a program graph, and encode each program in a bottom-up manner to learn semantic compositionality (Section 3.2). After that, we construct a heterogeneous graph with the statement, table, and program nodes to enable to reason over multiple types of nodes (Section 3.3). Finally, we propose a graph-based neural network to reason over the constructed heterogeneous graph (Section 3.4). Figure 2 shows the architecture of the proposed graph-based approach.

3.1 Context Encoding

Pre-trained models show their strong text understanding ability across a great many NLP tasks. In this work, we use pre-trained BERT (Devlin et al., 2018) as the backbone of our approach. Given the statement and the table, to make our input suitable for the pre-trained BERT model, following Wenhu Chen and Wang (2020), we first linearize the table into a sequence with a template-based method (denote as table sequence). Then we feed the statement, function names (such as *eq*, *count*, *filter_eq*, and so on, which are pre-defined in LPA algorithm (Wenhu Chen and Wang, 2020)), and the table sequence into pre-trained BERT with the format [[CLS], S, [SEP], F, [SEP], T, [SEP]], where S, F, T represent the statement, function names and table sequence respectively. [CLS] is a special token used for classification and [SEP] is a delimiter.

3.2 Program Graph Construction and Encoding

Program Synthesis and Selection Program is a kind of logical form that conveys symbolic information. In this work, we use LPA proposed by Wenhu Chen and Wang (2020) to perform the program synthesis and selection step. They define the plausible API set to include roughly 50 different functions


eq
5
count
filter_eq
all_rows
competition
south american championship
eq { 5 ; count { filter_eq { all_rows ; competition ; south american championship } } }
4 of the 5 game be for the south american championship
date
may 11, 1919
may 18, 1919
may 26, 1919
may 29, 1919
june 1, 1919
competition
south american championship
south american championship
south american championship
south american championship
taca roberto cherry
program1
program2
program3
program4
1. eq { 5 ; count { filter_eq { all_rows ; competition ; south american championship } } }
2. …

(a) Program Graph

(b) Heterogeneous Graph

Figure 3: Figure (a) shows an example of the constructed program graph and the corresponding program. The nodes in the program graph come from the table, the statement and function names. Figure (b) shows an example of the constructed heterogeneous graph. Program1 - program4 correspond to the 4 programs shown in Figure 1. The directions of the arrows stand for message passing processes, namely *table-to-table*, *statement-to-table*, *table-to-statement*, *program-to-statement*, *program-to-table*. We omit some columns in the table and the edges starts from program2 - program4 for brevity.

like *min, max, count, average, filter, and*, and so on. Each program is tree-structural. Following their setting, each case could synthesize up to 50 programs. Then a transformer-based approach is used to rank the candidate programs. A transformer-based two-way encoder is trained in a weakly supervised manner that all label-consistent programs are regarded as true programs. Due to the issue of spurious programs, namely the programs are wrong but with true returned results, we choose the top-5 predicted programs. The selected programs will be used for the rest of the paper.

Program Graph Construction Given a program, we construct a program graph in a bottom-up manner to learn the semantic compositionality of the program. Specifically, We treat each function name and argument as a node in the graph, then we add edges between each argument and its corresponding function name. Figure 3(a) shows the structure of the program graph.

Program Encoding We use gated graph neural network (Li et al., 2015) to encode programs to learn the structure inside the programs. Since all programs are tree-structural, a bottom-up manner of the edges could learn the semantic compositionality of the programs. Here we refer to a program $p = \{V, E\}$, where $V = \{v_1, v_2, ..., v_n\}$, $E = \{e_1, e_2, ..., e_m\}$. V and E stands for the nodes and edges respectively. The gated graph neural network is designed as follows:

$$h_i^{(0)} = h_i || 0 \tag{1}$$

$$m_i^{(l+1)} = \sum_{e_j \in E_{(i)}} W \cdot h_j^{(l)} \tag{2}$$

$$h_i^{(l+1)} = \mathrm{GRU}(m_i^{(l+1)}, h_i^{(l)}) \tag{3}$$

where h_i means the representations of the i-th node in the program graph obtained by the output of BERT encoder, $E_{(i)}$ represents the edges that contains the node v_i, l means the l-th layer of the gated graph neural network. $||$ stands for cancatenation operation. Finally, we obtain the final representation of the program with a max aggregator:

$$h_p = \max(h_1, h_2, ..., h_n) \tag{4}$$

3.3 Heterogeneous Graph Construction

Given the statement, table, and selected programs, we construct a heterogeneous graph to perform graph-based reasoning. We treat each statement token, table token, and program as a graph node. Our graph exploits structural information in the table, and the connections among the table, statement, and programs. Figure 3(b) shows an example of our constructed graph. Our heterogeneous graph can be constructed in the following steps:

- Each table node is connected to its corresponding column node.
- All table nodes in the same row are fully-connected to each other.
- Statement nodes are connected to the table nodes that have the same content.
- Program nodes are connected to the nodes in the table and statement that appears in the program.

In our heterogeneous graph, table nodes denote the cells in the table, including table heads and table contents. Statement nodes denote the entities in the statement. And program nodes denote the programs selected by LPA algorithm (Wenhu Chen and Wang, 2020). The representations of statement nodes are obtained by program encoding. The first three types of edges are also employed in (Zhong et al., 2020a), which aims to exploit the structure in the table and build connections between the table and statement. The last type of edges is expected to build connections between programs and the table, and the connections between programs and the statement. By constructing such heterogeneous graph, we categorize the edges into five classes, namely *table-to-table*, *statement-to-table*, *table-to-statement*, *program-to-statement*, *program-to-table*. Table-to-table means the edge is from a table node to a table node, others are the same. Different types of edges correspond to different message passing processes, which will be introduced in Section 3.4.

3.4 Reasoning over Heterogeneous Graph

Node initialization We get the initial representation of each node from the output of the pre-trained BERT model. For statement nodes and table nodes, each node representation is initialized by averaging the representations of corresponding tokens generated by the pre-trained BERT model. For program nodes, the representations are generated by the program encoder introduced in Section 3.2.

Heterogeneous Graph Layer Given a constructed heterogeneous graph and representations of the nodes in the graph, we use graph attention networks (GAT) (Veličković et al., 2017) to obtain node representations by encoding graph-structural information, which is designed as follows:

$$z_{i,j} = \mathrm{LeakyReLU}(W_a[W_q h_i || W_k h_j]) \tag{5}$$

$$\alpha_{i,j} = \frac{\exp(z_{i,j})}{\sum\limits_{l \in N_i} \exp(z_{i,l})} \tag{6}$$

$$u_i = \sigma(\sum_{j \in N_i} \alpha_{i,j} \mathrm{W_v} h_j) \tag{7}$$

where W_a, W_q, W_k, W_v are learned parameters, h_i, h_j are the representations of the i-th and j-th node obtained by the output of BERT encoder, $\alpha_{i,j}$ is the attention weight between the i-th node and j-th node. $||$ means concatenation operation.

In this work, we perform *table-to-table*, *statement-to-table*, *table-to-statement*, *program-to-statement* and *program-to-table* message passing processes. The first type aims to exploit the structure in the table. The second type and the third type are expected to build connections between the table and statement. The last two types are excepted to leverage the programs to build connections between programs and the statement and table.

We use GAT network for all the above message passing processes. For the i-th node, we obtain u_i^{tt}, u_i^{st}, u_i^{ts}, u_i^{ps}, u_i^{pt} respectively. Then we aggregate information as follows to map the representations from different sources to the same embedding space:

$$u_i^T = u_i^{tt} \cdot \mathrm{W_t} + u_i^{st} \cdot \mathrm{W_s} + u_i^{pt} \cdot \mathrm{W_p} + \mathrm{b} \tag{8}$$

$$u_i^S = u_i^{ts} \cdot \mathrm{W_t} + u_i^{ps} \cdot \mathrm{W_p} + \mathrm{b} \tag{9}$$

where u_i^T means the representation of the i-th table node, u_i^S means the representation of the i-th statement node, $\mathrm{W_t}$, $\mathrm{W_s}$, $\mathrm{W_p}$, b are learned parameters. tt, st, ts, ps, pt means *table-to-table*, *statement-to-table*, *table-to-statement*, *program-to-statement* and *program-to-table* respectively.

Besides, we add a residual connection for table nodes and statement nodes to avoid gradient vanishing as follows:

$$m_i^{(l)} = u_i^{(l)} + m_i^{(l-1)} \tag{10}$$

where l indicates the l-th heterogeneous graph layer.

After the iteration of several layers, we perform mean pooling over all statement nodes and table nodes to get the final aggregated representation, followed by a one-layer MLP to get the final prediction.

$$u = \mathrm{mean}(m_1^S, m_2^S, ..., m_N^S, m_1^T, m_2^T, ..., m_M^T) \tag{11}$$

$$o = \mathrm{softmax}(\mathrm{ReLU}(\mathrm{W} \cdot m + \mathrm{b})) \tag{12}$$

where W,b are trainable parameters. S and T stand for statement and table. N and M are the numbers of statement nodes and table nodes, respectively.

4 Experiments

4.1 Dataset and Experimental Settings

We evaluate our model on the TABFACT (Wenhu Chen and Wang, 2020), a large-scale table-based fact verification dataset, which is a collection of 16k Wikipedia tables for 118k human-annotated natural language statements (92283 samples in training set, 12792 samples in development set and 12779 samples in test set). Each statement is labeled as either ENTAILED or REFUTED. Samples in this dataset are expected to combine linguistic information and symbolic information. We use accuracy to evaluate our model. The test set is further divided into simple channels and complex channels to distinguish the difficulty, which has 4171 and 8608 samples respectively. Besides, A small test with 2K samples is held out for human evaluation.

We start with BERT-base model (Devlin et al., 2018) in all of our experiments. The maximum sequence length is 512, and the batch size is 6. We set the hidden size to 768, which is the same as the output of the BERT-base model. We adopt cross-entropy loss as our loss function. And the learning rate is 1e-5.

4.2 Baselines

In this section, we describe the baseline systems utilized in our experiments.

Table-BERT Table-BERT (Wenhu Chen and Wang, 2020) views the table verification problem as a two-sequence binary classification program by leveraging pre-trained BERT model to encode the linearized tables and statements.

Latent Program Algorithm (LPA) LPA (Wenhu Chen and Wang, 2020) formulates the table-based fact verification as a weakly supervised learning problem by synthesizing latent programs and ranking the candidate programs with a Transformer-based two-way encoder (Vaswani et al., 2017).

LogicalFactChecker LogicalFactChecker (Zhong et al., 2020a) derives a program of the statement in a semantic parsing manner and propose a program-driven module network to exploit the hierarchical structure of the program.

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 (program from LPA)	71.7	71.6	85.5	64.8	74.2
LogicalFactChecker (program from Seq2Action)	71.8	71.7	85.4	65.1	**74.3**
HeterTFV	**72.5**	**72.3**	**85.9**	**65.7**	74.2
Human Performance	-	-	-	-	92.1

Table 1: Results on the TABFACT dataset. *T+F* means table followed by fact, *F+T* means fact followed by table. *Horizontal* and *Vertical* stand for the order to scan the table cells when linearizing tables. *Concatenate* and *Template* stand for the strategies to concatenate the table cells without or with templates respectively. *Voting* means assigning each program with equal weight and vote for the result. *Weighted-Voting* is to compute a weighted-sum score for each program and vote for the result. *Ranking* means using the result of the top-ranked program. *Seq2Action* means the programs are generated by the sequence-to-action approach (Chen et al., 2018; Iyer et al., 2018; Guo et al., 2018; Zhong et al., 2020a).

4.3 Experimental Results

Table 1 shows our results on the TABFACT dataset (Wenhu Chen and Wang, 2020). Our model achieves state-of-the-art performance, 72.3% accuracy on the test set. In particular, compared with Table-BERT, our model gains an improvement of 7.2% in accuracy on the test set, which shows the effect of symbolic information and the reasoning ability over multiple types of information. Besides, our model outperforms LogicalFactChecker by 0.7% on the test set by employing the same program synthesis method (program from LPA), which indicates that the proposed heterogeneous graph-based model has a better ability to combine information of different types than concatenation operation only.

4.4 Ablation Study

Effect of model components We prove the effect of our model by removing each model component sequentially. Table 2 shows the effect of our model components. We conduct ablation experiments on the development set. We first remove the program encoding process, and performing mean pooling over function names and arguments in the programs as program representations instead. The result drops by 0.7%, which proves that the proposed program encoder has a great effect. Then we remove all program information, namely perform reasoning only on the table nodes and statement nodes. The performance of our model drops by 1.6%. This proves that the combination of linguistic information and symbolic information performs better than a single one. Next, we remove the residual connection. The result drops slightly, by 0.5%. And we remove graph-based reasoning, namely concatenate representations of all types of nodes directly. our model's result drops by 1.2%, which indicates that the proposed heterogeneous graph and graph-based reasoning neural network make sense in our model. Finally, we remove all modules above, namely, we simply concatenate statement tokens and linearized table tokens into the pre-trained BERT model. Under this circumstance, The result drops by 2.2%. All ablation studies above illustrate that both program information and proposed graph-based reasoning approach are effective to combine linguistic information and symbolic information on the table-based fact verification task.

Model	Val
HeterTFV	72.5
w/o Program Encoding	71.8
w/o Program Information	70.9
w/o Residual Connection	72.0
w/o Graph-based Reasoning	71.3
w/o All Above components	70.3

Table 2: Effect of the model components.

Model	Val
HeterTFV	72.5
w/o Table-to-Table	71.5
w/o Statement-to-Table	71.9
w/o Table-to-Statement	72.3
w/o Program-to-Statement	71.7
w/o Program-to-Table	72.0

Table 3: Effect of the different types of message passing processes.

Effect of message passing We conduct ablation experiments on the development set to better understand different types of message passing processes. The experimental results are shown in Table 3. We first remove the *table-to-table* message passing process, the result drops by 1.0%, which proves the importance of the structure in the table. Then we remove the *statement-to-table* message passing process, the result of our model drops by 0.6%. And we remove the *table-to-statement* message passing process, the result drops slightly, by 0.2%, which indicates that the connections between the table and the statement also have some effect. Next, we remove the *program-to-statement* message passing process, the result drops by 0.8%. Finally, we remove *program-to-table* message passing process, the result drops by 0.5%. The removal of *program-to-statement* and *program-to-table* message passing processes indicates that the combination of linguistic information and symbolic information plays a great role in our model.

Besides, the results above show that all types of message passing play a great role, which prove the rationality of our constructed heterogeneous graph and the effect of our proposed graph-based reasoning approach to combine different types of information.

4.5 Case Study

Figure 4 shows an example in our experiments. We omit some rows, columns in the table and some programs for simplicity. To verify the statement *there be more than 5 number of week for 50073 attendance*, the key issue is to understand *more than* in the statement. In our selected programs, the first program contains the *filter_greater* function, which could help the model to understand *more than* in the statement. Besides, the first selected program can represent the meaning of the statement according to the definition of the latent program algorithm (LPA). The program encoding process could catch the structure and learn the semantic compositionality of the program, which is beneficial to verify the statement. And although the second selected program couldn't stand for the meaning of the statement completely, some keywords in the program could be also helpful to verify the statement.

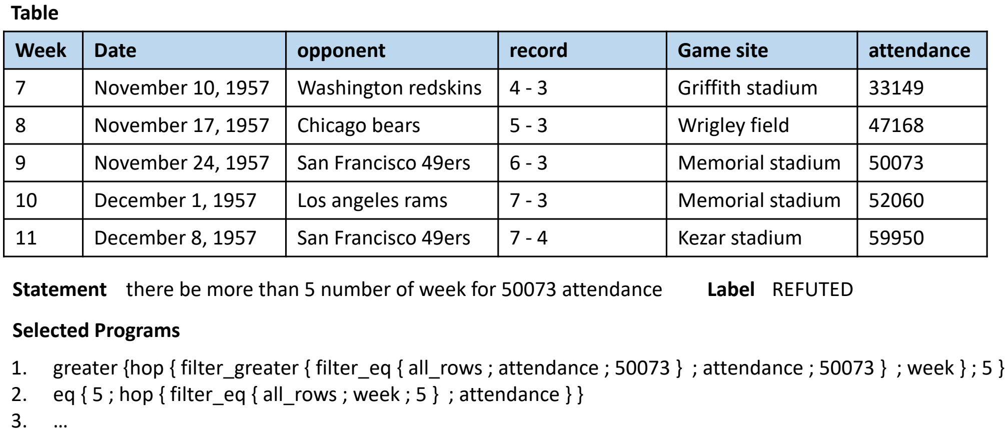
4.6 Error Analysis

We analyze the error examples in the TABFACT dataset (Wenhu Chen and Wang, 2020), which can be mainly categorized into three classes.

In our analysis, the first error type of our framework is due to the failure to synthesize programs. For the statement *the shortest name of any competitor belongs to the skip for spain*, we failed to synthesize the corresponding programs, which may be because that some entities in the statement are not detected in the entity linking phase. It's hard to incorporate symbolic information without any program.

The second type of error is due to spurious programs, which is especially serious in binary classification tasks such as table-based fact verification. For the statement *there be over 15 million us viewer in season 2*, we selected four programs. The key issue to verify this fact is to compare the number of the us viewer in season 2 and 15 million. However, for example, the program *eq* { *count* { *filter_eq* { *all_rows ; no in season ; 15* } } *; count* { *filter_eq* { *all_rows ; no in season ; 2* } } } aims to judge whether the number of instances in the table whose season is 15 and the number of instances in the table whose season is 2 are equal, which leads to a misunderstanding of the model because there is no enough information to perform a comparison operation.

The third type of error is due to the diversity of program expression. For example, the program formats



Table

Week	Date	opponent	record	Game site	attendance
7	November 10, 1957	Washington redskins	4 - 3	Griffith stadium	33149
8	November 17, 1957	Chicago bears	5 - 3	Wrigley field	47168
9	November 24, 1957	San Francisco 49ers	6 - 3	Memorial stadium	50073
10	December 1, 1957	Los angeles rams	7 - 3	Memorial stadium	52060
11	December 8, 1957	San Francisco 49ers	7 - 4	Kezar stadium	59950

Statement there be more than 5 number of week for 50073 attendance **Label** REFUTED

Selected Programs

1. greater {hop { filter_greater { filter_eq { all_rows ; attendance ; 50073 } ; attendance ; 50073 } ; week } ; 5 }
2. eq { 5 ; hop { filter_eq { all_rows ; week ; 5 } ; attendance } }
3. …

Figure 4: Case study of our approach.

filter_greater { *A* ; *B* } and *filter_less* { *B* ; *A* } have the same meanings. But these two programs make the model have a different understanding because they have different function names.

5 Related Work

5.1 Fact Verification

The fact verification task aims to verify whether a textual statement is entailed or refuted by the given evidence. Most of the existing methods of fact verification focus on dealing with the unstructured text as evidence. Thorne et al. (2018) release a dataset named FEVER. There are several works on this dataset. Nie et al. (2019) modify ESIM (Chen et al., 2017) and design a neural semantic matching network for fact verification. Zhou et al. (2019) propose to reason and aggregate over claim-evidence pair with graph attention network(Veličković et al., 2017). Liu et al. (2020) propose a kernel graph attention network that conducts more fine-grained evidence selection and reasoning. Zhong et al. (2020b) construct a semantic-level graph and present two graph-driven representation learning mechanisms to perform reasoning over the claim-evidence graph.

Recently, there have been some studies on reasoning over semi-structured evidence. Wenhu Chen and Wang (2020) first release the TABFACT dataset, which is a large-scale dataset for table-based fact verification. Besides, they release two baselines: Table-BERT and latent program algorithm (LPA). In Table-BERT, they propose an approach to convert a table into the natural language and apply the pre-trained BERT model to predict the result. In LPA, they first parse the statement into the LISP-like program format to represent its semantics, then rank all candidates with a discriminator that with a Transformer-based two-way encoder (Vaswani et al., 2017). Zhong et al. (2020a) propose a graph-based neural module network to utilize logical operations for fact checking by presenting a graph-based mask matrix for self-attention mechanism. Different from their work, they focus on how to represent symbolic information, and we focus on how to combine different types of information. In our work, we propose a heterogeneous graph-based neural network that aims to make a effective combination of linguistic information and symbolic information.

5.2 Heterogeneous Graph for NLP

Graph neural networks show their superior performance when dealing with structured data. Most of the existing works focus on the homogeneous graph that all nodes in the graph have the same type. However, most of the graphs in the real world have different types, namely heterogeneous graph. A lot of works have emerged based on heterogeneous graphs. Wang et al. (2019) propose a novel heterogeneous graph neural network based on the hierarchical attention, including node-level and semantic-level attentions. Linmei et al. (2019) establish a heterogeneous graph among documents, topics, and entities to perform reasoning on the semi-supervised short text classification task. Tu et al. (2019) introduce a heterogeneous graph constructed with documents, entities, and candidates for multi-hop reading comprehension. Wu et al. (2019) propose a novel relation-aware dual-graph convolutional network to incorporate relation information for heterogeneous knowledge graphs. Wang et al. (2020) propose a heterogeneous graph network for extractive summarization. Shangwen Lv and Hu (2020) propose a heterogeneous graph from heterogeneous external knowledge source for commonsense question answering. Inspired by the success of the heterogeneous graph-based neural networks above, in this work, we construct a heterogeneous graph among the statement, table, and programs to enable the combination of multiple types of information for the table-based fact verification task and propose a graph-based neural network to reason over the proposed heterogeneous graph.

6 Conclusion

In this work, we focus on the table-based fact verification task, propose *HeterTFV*, a graph-based approach that learns to combine linguistic information and symbolic information effectively by reasoning over the constructed heterogeneous graph. Specifically, we first construct a program graph to encode programs. Then we construct a heterogeneous graph among tables, statements, and programs to incorporate linguistic information and symbolic information. Finally, we propose a heterogeneous graph-based neural network to perform reasoning to learn to combine both types of information. We conduct experiments on the TABFACT dataset. Experimental results illustrate that our model outperforms all the baseline systems with considerable increment and achieves state-of-the-art performance. In the future, we will do more attempt to combine linguistic information and symbolic information on the other tasks, such as knowledge graph reasoning or text reasoning.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61976068) and the Key Development Program of the Ministry of Science and Technology (No. 2019YFF0303003). According to the meaning by Harbin Institute of Technology, the corresponding author of this paper is Ting Liu.

References

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-sql parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. Global reasoning over database structures for text-to-sql parsing. 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 3650–3655.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Bo Chen, Le Sun, and Xianpei Han. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V Le. 2019. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2942–2951.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33.

Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. 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 4823–4832.

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, July. Association for Computational Linguistics.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866.

Jingjing Xu Duyu Tang Nan Duan Ming Gong Linjun Shou Daxin Jiang Guihong Cao Shangwen Lv, Daya Guo and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, USA*, pages 8449–8456. AAAI Press.

Amir Soleimani, Christof Monz, and Marcel Worring. 2020. Bert for evidence retrieval and claim verification. In *European Conference on Information Retrieval*, pages 359–366. Springer.

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.

Ming Tu, Guangtao Wang, Jing Huang, Yun Tang, Xiaodong He, and Bowen Zhou. 2019. Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2704–2713.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online, July. Association for Computational Linguistics.

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

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. *arXiv preprint arXiv:1908.08210*.

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, July. 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, July. 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.