Exploring Decomposition for Table-based Fact Verification

Xiaoyu Yang and **Xiaodan Zhu**
Ingenuity Labs Research Institute & ECE, Queen's University, Canada
{xiaoyu.yang, xiaodan.zhu}@queensu.ca

Abstract

Fact verification based on structured data is challenging as it requires models to understand both natural language and symbolic operations performed over tables. Although pre-trained language models have demonstrated a strong capability in verifying simple statements, they struggle with complex statements that involve multiple operations. In this paper, we improve fact verification by decomposing complex statements into simpler subproblems. Leveraging the programs synthesized by a weakly supervised semantic parser, we propose a program-guided approach to constructing a pseudo dataset for decomposition model training. The subproblems, together with their predicted answers, serve as the intermediate evidence to enhance our fact verification model. Experiments show that our proposed approach achieves the new state-of-the-art performance, an 82.7% accuracy, on the TABFACT benchmark.

1 Introduction

Fact verification aims to validate if a statement is entailed or refuted by given evidence. It has become crucial to many applications such as detecting fake news and rumor (Rashkin et al., 2017; Thorne et al., 2018; Goodrich et al., 2019; Vaibhav et al., 2019; Kryscinski et al., 2020). While existing research mainly focuses on verification based on unstructured text (Hanselowski et al., 2018; Yoneda et al., 2018; Liu et al., 2020; Nie et al., 2019), a recent trend is to explore structured data as evidence, which is ubiquitous in our daily life.

Verification performed with structured data presents research challenges of fundamental interests, as it involves both informal inference based on language understanding and symbolic operations such as mathematical operations (e.g., `count` and `max`). While all statements share the same set of operations, complex statements, which involve multiple operations, are more challenging than simple statements. Pre-trained models such as BERT (Devlin et al., 2019) have presented superior performances on verifying simple statements while still struggling with complex ones: a performance gap exists between the simple and complex tracks (Chen et al., 2020).

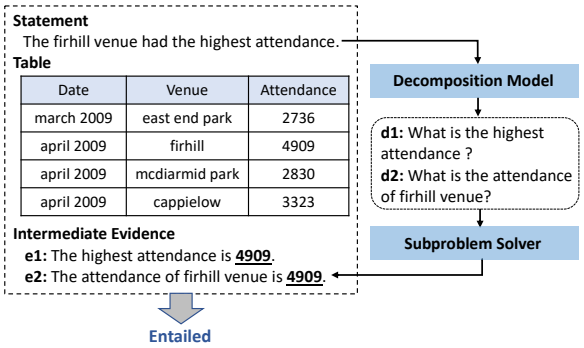

Figure 1: Overview of the proposed approach. An example of executable program parsed from the statement is: $eq\{max\{all\_rows; attendance\}; hop\{filter\_eq\{all\_rows; venue; firhill\}; attendance\}\}$.

In this paper, we propose to decompose complex statements into simpler subproblems to improve table-based fact verification, as shown in a simplified example in Figure 1. To avoid manually annotating gold decompositions, we design a program-guided pipeline to collect pseudo decompositions for training generation models by distinguishing four major decomposition types and designing templates accordingly. The programs we used are parsed from statements with a weakly supervised parser with the training signals from final verification labels. Figure 1 shows a statement-program example. We adapt table-based natural language understanding systems to solve the decomposed subproblems. After obtaining the answers to subproblems, we combine them in a pairwise manner as intermediate evidence to support the final prediction.

We perform experiments on the recently proposed benchmark TABFACT (Chen et al., 2020)

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1045–1052
November 7–11, 2021.

and achieve a new state-of-the-art performance, an 82.7% accuracy. Further studies have been conducted to provide details on how the proposed models work.

2 Method

2.1 Task Formulation and Notations

Given an evidence table T and a statement S, we aim to predict whether T *entails* or *refutes* S, denoted by $y \in \{1, 0\}$. For each statement S, the executable program derived from a semantic parser is denoted as z. An example of program is given in Figure 1. Each program $z = \{op_i\}_{i=1}^{M}$ consists of multiple symbolic operations op_i, and each operation contains an operator (e.g., *max*) and arguments (e.g., *all_rows* and *attendance*). A complex statement S can be decomposed into subproblems $D = \{d_i\}_{i=1}^{N}$, with the answers being $\{a_i\}_{i=1}^{N}$. Using combined problem-answer pairs as intermediate evidence $E = \{e_i\}_{i=1}^{N}$ where $e_i = (d_i, a_i)$, our model maximizes the objective $\log p_\theta(y|T, S, E)$.

2.2 Statement Decomposition

Constructing a high-quality dataset is key to the decomposition model training. Since semantic parsers can map statements into executable programs that not only capture the semantics but also reveal the compositional structures of the statements, we propose a program-guided pipeline to construct a pseudo decomposition dataset.

2.2.1 Constructing Pseudo Decompositions

Program Acquisition. Following Chen et al. (2020), we use latent program algorithm (LPA) to parse each statement S into a set of candidate programs $\mathcal{Z} = \{z_i\}_{i=1}^{K}$. To select the most semantically consistent program z^* among all candidates and mitigate the impact of spurious programs, we follow Yang et al. (2020) to optimize the program selection model with a margin loss, which is detailed in Appendix A.1.

By further removing programs that are label-inconsistent or cannot be split into two isolated sub-programs from the root operator, we obtain the remaining (T, S, z) triples as the source of data construction[1].

Decomposition Templates. Programs are formal, unambiguous meaning representations for the corresponding statements. Designed to support automated inference, the program z encodes the central feature of the statement S and reveals its compositional structures. Our statement decomposition is based on the structure of the program. Specifically, we first extract program skeleton z_s by omitting arguments in the selected program z, then we group the (T, S, z) triples by z_s to identify four major decomposition types: **conjunction**[2], **comparative**, **superlative**, and **uniqueness**.

Some simple templates associated with each decomposition type are designed, which contain instructions on how to decompose the statement, and this manual process only takes a few hours. In this way, we can construct pseudo decompositions, including sub-statements and sub-questions, by filling the slots in templates according to the original statements or program arguments. Templates and decomposition examples can be found in Figure 2. Each sample in our constructed pseudo dataset is denoted as a (S, c, D') triple, where c indicates one of the four types and D' is a sequence of pseudo decompositions.

Data Augmentation. With the (T, S, z) triples, we perform data augmentation. Since some entity mentions in S and z can be linked to cells in T, we can randomly replace the linked entities in S and z with different values in the same column of T. For example, in Figure 1, we can replace the linked

Conjunction	
S	rayo earns 36 points and ferrol earns 41 points
z	*and { eq { hop { filter_eq { all_rows ; club ; rayo } ; points } ; 36 } ; eq { hop { filter_eq { all_rows ; club ; ferrol } ; points } ; 41 } }*
d1	rayo earns 36 points .
d2	ferrol earns 41 points .
Superlative	
S	princes park venue recorded the highest crowd participation
z	*eq { hop { argmax { all_rows ; crowd } ; crowd } ; hop { filter_eq { all_rows ; venue ; princes park } ; crowd } }*
d1	what is the highest crowd ?
d2	what is the crowd of princes park ?
Comparative	
S	daniel had a longer react than felix
z	*greater { hop { filter_eq { all_rows ; athlete ; daniel } ; react } ; hop { filter_eq { all_rows ; athlete ; felix } ; react } }*
d1	what is the react of daniel ?
d2	what is the react of felix ?
Uniqueness	
S	itf 25k was only the tier on may 8th
z	*and { only { filter_eq { all_rows ; date ; may 8th } } ; eq { hop { filter_eq { all_rows ; date ; may 8th } ; tier } ; itf 25k } }*
d1	how many tier on may 8th ?
d2	itf 25k was the tier on may 8th .

Figure 2: Decomposition templates.

[1]These triples do not involve any tables or statements in the dev/test set of the dataset used in this paper.

[2]The conjunction type has overlap with the other three types in the cases that the sub-statements connected by conjunctions can be further decomposed.

entity *"firhill"* with another randomly selected entity *"cappielow"*. Another augmentation strategy is inverting superlative and comparative. For the examples belong to **superlative** and **comparative**, we replace the original superlative or comparative in statements with its antonym, such as *higher* → *lower* and *longest* → *shortest*. In this way, we generate another 3k pseudo statement-decomposition pairs. In total, the final decomposition dataset used for generation model training includes 9,696 samples. More statistics are available in Appendix A.2.

2.2.2 Learning to Decompose

Decomposition Type Detection. Given a statement S, we train a five-way classifier based on BERT to identify whether the statement is decomposable and if yes, which decomposition type it belongs to. In addition to the four types mentioned in the previous section, we add an *atomic* category by involving additional non-decomposable samples. Only the statements not assigned with *atomic* labels can be used for decomposition.

Decomposition Model. We finetune the GPT-2 (Radford et al., 2019) on the pseudo dataset for decomposition generation. Specifically, given the (S, c, D') triple, we train the model by maximizing the likelihood $J = \log p_\theta(D'|S, c)$. We provide the model with gold decomposition type c during training and the predicted type $\hat{c}$ during testing. Only informative and well-formed decompositions are involved in the subsequent process to enhance the downstream verification. In case some sub-statements need further decomposition, it can be implemented by resending them to our pipeline[3].

2.3 Solving Subproblems

We adapt TAPAS (Eisenschlos et al., 2020), a SOTA model on table-based fact verification and QA task, to solve the decomposed subproblems. Verifying sub-statements is formulated as a binary classification with the TAPAS model fine-tuned on the TABFACT (Chen et al., 2020) dataset. To answer each sub-question, we use the TAPAS fine-tuned on WikiTableQuestions (Pasupat and Liang, 2015) dataset. We combine the subproblems and their answers in a pairwise manner to obtain the intermediate evidence $E = \{e_i\}_{i=1}^N = \{(d_i, a_i)\}_{i=1}^N$, an example evidence is shown in Figure 1.

[3]In most cases, there is no need to perform iterative decomposition, and we leave finer-grained decomposition for future research.

2.4 Recombining Intermediate Evidence

Downstream tasks can utilize the intermediate evidence in various ways. In this paper, we train a model to fuse the evidence E together with the statement S and table T for table-based fact verification[4]. Specifically, we jointly encode S and T with TAPAS to obtain the concentrated representation $\boldsymbol{h}_{ST}$. We encode multiple evidence sentences with another TAPAS following the document-level encoder proposed in Liu and Lapata (2019) by inserting [CLS] token at the beginning of every single sentence e_i and taking the corresponding [CLS] embedding $\boldsymbol{h}_{e_i}$ in the final layer to represent e_i.

We employ a gated attention model to obtain aggregated evidence representation $\boldsymbol{h}_{evd}$ and predict the final label as follows:

$$\boldsymbol{h}_{evd} = \sum_{i=0}^{N} a_i \boldsymbol{h}_{e_i}, \quad a_i = \sigma(\boldsymbol{h}_{ST}^T \boldsymbol{h}_{e_i})$$
$$y = \sigma(W([\boldsymbol{h}_{evd} \oplus \boldsymbol{h}_{ST}]))$$

where W are trainable parameters, σ is the sigmoid function, and $\oplus$ indicates concatenation.

3 Experiments

Setup. We conduct our experiments on a large-scale table-based fact verification benchmark TABFACT (Chen et al., 2020). The test set contains a *simple* and *complex* subset according to difficulty. A *small* test set is further annotated with human performance. Following the previous work, we use *accuracy* as the evaluation metric. Details of the data are listed in Appendix A.3.

Implementation Details. During fine-tuning the GPT-2 model to generate decomposition, we run the model with a batch size of 5 for 30 epochs using Adam optimizer (Kingma and Ba, 2015) with a learning rate of 2e-6. We optimize the model for final verification prediction using Adam optimizer with a learning rate of 2e-5 and a batch size of 16. It usually takes 11 to 14 epochs to converge. Our code is available at `https://github.com/arielsho/Decomposition-Table-Reasoning`.

Main Results. We compare our model with different baselines on TABFACT, including LPA (Chen et al., 2020), Table-BERT (Chen et al., 2020), LogicalFactChecker (Zhong et al., 2020), HeterTFV (Shi et al., 2020), SAT (Zhang

[4]For the non-decomposable statements, we put "no evidence" as the placeholder.

Model	Val	Test	Simple	Complex	Small
Human	-	-	-	-	92.1
LPA	57.7	58.2	68.5	53.2	61.5
Table-BERT	66.1	65.1	79.1	58.2	68.1
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
TAPAS-BASE	79.1	79.1	91.4	73.1	81.2
TAPAS-LARGE	81.5	81.2	93.0	75.5	84.1
OURS-BASE	**80.8**	**80.7**	**91.9**	**75.1**	**82.5**
OURS-LARGE	**82.7**	**82.7**	**93.6**	**77.4**	**84.7**

Table 1: The accuracy (%) of models on TABFACT.

Type		TAPAS-BASE	OURS-BASE
Conj.	(15%)	79.9	**82.6**
Sup.	(13%)	81.3	**82.4**
Comp.	(13%)	69.1	**72.1**
Uniq.	(6 %)	70.4	**74.4**
Atomic	(53%)	81.7	**82.5**

Table 2: Decompositions improve the performance on test set over 4 decomposition types.

	BLEU-4 on Dev	Human Val
Our Decomp.	**56.75**	**68%**
w/o data aug	48.42	56%
w/o type info	54.74	63%

Table 3: Evaluation of decomposition quality.

	train	val	test	simple	complex
Our Decomp.	**41.6**	**46.3**	**46.7**	**20.2**	**59.5**
w/o data aug	35.2	39.1	39.4	16.3	50.7

Table 4: Percentage of valid decomposition on all splits in TABFACT.

et al., 2020), ProgVGAT (Yang et al., 2020), and TAPAS (Eisenschlos et al., 2020). Details of the compared systems can be found in Appendix A.4.

Table 1 presents the test accuracy of our BASE model and LARGE model, which are built upon TAPAS-BASE and TAPAS-LARGE, respectively. Results show that our model consistently outperforms the TAPAS baseline (80.7% vs. 79.1% for the base and 82.7% vs. 81.2% for the large model)[5]. We show in Table 2 that our decomposition model decomposes roughly 47% of the total TABFACT test cases, and our model outperforms the TAPAS model over all types of decomposed statements.

Evaluation of Decompositions. We use both an automated metric and human validation to evaluate the decomposition quality. For the automated metric, we randomly sample 1,000 training cases from the pseudo decomposition dataset as the hold-out validation set, based on which we use BLEU-4 (Papineni et al., 2002) to measure the generation quality. We also sample 100 decomposable cases from the TABFACT test set and ask three crowd workers to judge whether the model produces plausible decompositions. The ablation results in Table 3 indicate that data augmentation and the use of type information improve the decomposition quality, and the BLEU-4 score on the pseudo decomposition dataset well reflects the human judgements.

Since we remove the defective decompositions to reduce noise in the verification task, the number of decomposed cases involved by our final verification model varies according to the decomposition quality. We provide the percentages of valid decompositions on all data splits of TABFACT in Table 4. The results show that our decompositions do not completely align with the simple/complex split provided in TABFACT, and data augmentation can improve the number of valid decomposition by around 7%. On the downstream verification task, a lower-quality decomposition (39.4%) yields a 0.4% performance drop compared to our proposed decomposition model (46.7%).

4 Related Work

Existing work on fact verification is mainly based on evidences from unstructured text (Thorne et al., 2018; Hanselowski et al., 2018; Yoneda et al., 2018; Thorne et al., 2019; Nie et al., 2019; Liu et al., 2020). Our work focuses on fact verification based on structured tables (Chen et al., 2020). Unlike the previous work (Chen et al., 2020; Zhong et al., 2020; Shi et al., 2020; Zhang et al., 2020; Yang et al., 2020; Eisenschlos et al., 2020), we propose a framework to verify statements via decomposition.

Sentence decomposition takes the form of Split-and-Rephrase proposed by Narayan et al. (2017) to split a complex sentence into a sequence of shorter sentences while preserving original meanings (Aharoni and Goldberg, 2018; Botha et al., 2018; Guo et al., 2020). In QA task, question decomposition has been applied to help answer multi-hop ques-

[5]We also conduct significance tests over both the base and large models (the proposed model vs. TAPAS), with the one-tail t-test. For the base model, the p-value is 4.7e-6 and for the large model, 3.2e-7.

tions (Iyyer et al., 2016; Talmor and Berant, 2018; Min et al., 2019; Wolfson et al., 2020; Perez et al., 2020). Our work mainly focuses on decomposing statements for table-based fact verification with pseudo supervision from programs.

5 Conclusion

In this paper, we propose a framework to better verify the complex statements via decomposition. Without annotating gold decompositions, we propose a program-guided approach to creating pseudo decompositions on which we finetune the GPT-2 for decomposition generation. By solving the decomposed subproblems, we can integrate useful intermediate evidence for final verification and improve the state-of-the-art performance to an 82.7% accuracy on TABFACT.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We also thank Yufei Feng for his helpful comments and suggestions on the paper writing.

References

Roee Aharoni and Yoav Goldberg. 2018. Split and rephrase: Better evaluation and stronger baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724, Melbourne, Australia. Association for Computational Linguistics.

Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from Wikipedia edit history. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 732–737, Brussels, Belgium. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. 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.

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.

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.

Ben Goodrich, Vinay Rao, Peter J. Liu, and Mohammad Saleh. 2019. Assessing the factual accuracy of generated text. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 166–175. ACM.

Yinuo Guo, Tao Ge, and Furu Wei. 2020. Fact-aware sentence split and rephrase with permutation invariant training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7855–7862. AAAI Press.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. UKP-athene: Multi-sentence textual entailment for claim verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 103–108, Brussels, Belgium. 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.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2016. Answering complicated question intents expressed in decomposed question sequences. *arXiv preprint arXiv:1611.01242*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. 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 3730–3740, Hong Kong, China. 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.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark. Association for Computational Linguistics.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6859–6866. AAAI Press.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, Copenhagen, Denmark. 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 and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. 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 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

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 FEVER2.0 shared task. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 1–6, Hong Kong, China. Association for Computational Linguistics.

Vaibhav Vaibhav, Raghuram Mandyam, and Eduard Hovy. 2019. Do sentence interactions matter? leveraging sentence level representations for fake news classification. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 134–139, Hong Kong. Association for Computational Linguistics.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198.

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.

Takuma Yoneda, Jeff Mitchell, Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. UCL machine reading group: Four factor framework for fact finding (HexaF). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 97–102, Brussels, Belgium. Association for Computational Linguistics.

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. 2020. 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.

A Appendix

A.1 Program Selection

We fine-tune the BERT (Devlin et al., 2019) to model $p_\theta(z|S)$, the probability of program z being semantically consistent with S. Since the gold programs are not available, we use the final verification labels as weak supervision. To mitigate the impact of spurious programs, i.e., programs execute to correct answers with incorrect operation combinations, we follow Yang et al. (2020) to optimize the model with a margin loss:

$$J = \max\left(p(z^-|S) - p(z^+|S) + \gamma, 0\right)$$

where z^- and z^+ denote the label-inconsistent and label-consistent programs with the highest probability, respectively. γ is the parameter to control the margin. The margin loss can encourage selecting one program that is most semantically relevant to the statement while maintaining a margin between the positive (label-consistent) and the negative (label-inconsistent) programs.

A.2 Statistics of Pesudo Dataset

We have 9,696 pseudo statement-decomposition pairs in total, and the number of samples belong to four decomposition types is given in Table 5. To train the decomposition type detection model, we add an additional *atomic* category with 1,739 statements.

Decomp. Type	# of samples
Conjunctive	1,798
Superlative	2,452
Comparative	4,528
Uniqueness	918

Table 5: Statistics of pseudo decomposition dataset.

A.3 Statistics of TABFACT Dataset

The statistics of TABFACT (Chen et al., 2020) can be found in Table 6, a large-scale table-based fact verification benchmark dataset on which we evaluate our method. The test set is further split into a *simple* set and a *complex* set, which include 4,171 and 8,608 sentences, respectively. A small test set with 1,998 samples are provided for human performance evaluation.

Split	Sentence	Table	Row	Col
Train	92,283	13,182	14.1	5.5
Val	12,792	1,696	14.0	5.4
Test	12,779	1,695	14.2	5.4

Table 6: Statistics of TABFACT.

A.4 Compared Systems

- **LPA** (Chen et al., 2020) derives a program for each statement by ranking the synthesized program candidates and takes the program execution results as predictions.
- **Table-BERT** (Chen et al., 2020) takes a linearized table and a statement as the input of BERT for fact verification.
- **LogicalFactChecker** (Zhong et al., 2020) utilizes the structures of programs to prune irrelevant information in tables and modularize symbolic operations with module networks.
- **HeterTFV** (Shi et al., 2020) is a graph-based reasoning approach to combining linguistic information and symbolic information.
- **SAT** (Zhang et al., 2020) is a structure-aware Transformer that encodes structured tables by injecting the structural information into the mask of the self-attention layer.
- **ProgVGAT** (Yang et al., 2020) leverages the symbolic operation information to enhance verification with a verbalization technique and a graph-based network.
- **TAPAS** (Herzig et al., 2020; Eisenschlos et al., 2020) is the previous SOTA model on TABFACT which extends BERT's architecture to encode tables and is jointly pre-trained with text and tables.